International Journal of Grid and Distributed Computing
Vol. 9, No. 11 (2016), pp.119-132
http://dx.doi.org/10.14257/ijgdc.2016.9.11.10

An Efficient K-Means Algorithm and its Benchmarking against other Algorithms

Anupama Chadha[*] and Suresh Kumar[**]

[*]*Faculty of Computer Applications, MRIU, Faridabad, India*
anupamaluthra@gmail.com
[**]*Faculty of Engineering and Technology, MRIU, Faridabad, India*
suresh.fet@mriu.edu.in

Abstract

K-Means is a widely used partition based clustering algorithm famous for its simplicity and speed. It organizes input dataset into predefined number of clusters. K-Means has a major limitation -- the number of clusters, K, need to be pre-specified as an input. Pre-specifying K in the K-Means algorithm sometimes becomes difficult in absence of thorough domain knowledge, or for a new and unknown dataset. This limitation of advance specification of cluster number can lead to "forced" clustering of data and proper classification does not emerge.

In this paper, a new algorithm based on the K-Means is developed. This algorithm has advance features of intelligent data analysis and automatic generation of appropriate number of clusters. The clusters generated by the new algorithm are compared against results obtained with the original K-Means and various other famous clustering algorithms. This comparative analysis is done using sets of real data.

Keywords*: Clustering, K-Means, Automatic generation of clusters*

1. Introduction

Clustering is an unsupervised technique for segregating a given input dataset into groups called clusters. This distribution is done in such a way that the objects in the same cluster are more similar as compared to objects in another cluster.

Many clustering algorithms have been proposed in the literature [1, 2, 3, 4]. These clustering algorithms are broadly classified into two categories, Hierarchical and Partitional. The hierarchical algorithms find clusters by arranging them into hierarchy (top-down or bottom-up), as a result they are not suitable for large datasets. On the other hand, the partition based clustering algorithms find the clusters independently. So they can easily partition large datasets.

However, K-Means has a major limitation -- the number of clusters, 'K', need to be pre-specified as an input to the algorithm. An inefficient grouping can occur if the person performing the clustering is not domain expert and inputs an incorrect number of clusters required. To overcome this limitation, researchers are still exploring ways.

In this paper, we have introduced a new algorithm based on the K-Means that takes only the numerical dataset as an input and generates appropriate number of clusters on the run. The algorithm has been implemented in C# and the results are compared with results of the original K-Means using software RapidMiner [5]. Further, the results of the proposed algorithm are compared with Hierarchical clustering algorithm with Single Linkage, Centroid Linkage, Complete Linkage and Average Linkage, Fuzzy C-Means, Hierarchical K-Means, K-Means using CCIA, K-Means presented by Ahmad *et al*. [6].

The analysis has been conducted on real datasets: Iris, Wine, Breast Tissue and Yeast from UCI Machine Repository [7].

ISSN: 2005-4262 IJGDC

The paper is organized as follows: Section 2 discusses some of the previous solutions to overcome the limitation of providing K as input. Section 3 introduces the new algorithm. Section 4 explains the algorithm using two small datasets. Section 5 discusses the results of the new algorithm on some real datasets. Section 6 gives the conclusion.

2. Related Work

This section discusses the attempts that have been made in the literature to remove the limitation of giving the number of clusters required as an input. Contribution of some authors is discussed below:

Dan Pelleg *et al*. [8] presented X Means algorithm as an extension of K Means which overcomes the limitation of inputting the value of K and performs faster than the original K Means. The algorithm starts with K as the lower bound of the given range and continues adding centroids until the upper bound is reached. During this process the centroid set that scores the best is recorded using the data structure kd-tree and is produced as output. The user has to input a range suggesting the lower and upper bound of K.

Kiris Wagstaff *et al*. [9] utilized information about the problem domain in performing clustering. This information about the problem domain is used in specifying the constraints on the data set. While assigning the data points to a cluster it is ensured that none of the constraint is violated.

V.Leela *et al*. [10] presented Y-Means algorithm based on K-Means algorithm. Initially, it runs K-Means algorithm on the data set and then follows the sequence of splitting, deleting and merging the clusters. The algorithm depends on K-Means algorithm to find the clusters initially.

B M Ahamed Shafeeq *et al*. [11] introduced an algorithm to overcome the problem of inputting the required number of clusters by finding the optimal number of clusters on the run. The main drawback of this approach is that it takes more computational time than the K-Means for larger data sets. Also the user has to input the number of clusters 'K' as 2 in the first run.

Mohamed Abubaker *et al*. [12] presented a new approach to overcome two of the major limitations of K-Means algorithm: Firstly, to select efficiently the initial centroids and second to remove the need of giving the number of clusters required as input to the algorithm. The proposed algorithm is based on the k-nearest neighbor method. They have suggested two versions of the algorithm. The first version takes kn (the number of nearest neighbors), and the number of clusters k in the data set as input parameters. Then the sorted list of data points is investigated until the number of obtained prototypes reached the specified k and the algorithm is aborted. The second version takes kn as the only input parameter and obtains both the number of clusters and the prototypes in parallel. The user has to input the number of nearest neighbors kn.

All the methods discussed above for removing the limitation of providing the value of K initially in K-Means algorithm require some parameter other than K to be input. However, the method suggested by Tibshirani *et al*. [13] overcomes this limitation by using the technique of Gap Statistic. This method utilizes the output generated by any clustering algorithm to compare the change in within cluster dispersion to that expected under an appropriate reference null distribution. This Gap method works well when the clusters are well separated. Also, the method is computationally complex. The new clustering technique named STep-wise Automatic Rival penalized (STAR) suggested by Cheung [14] overcomes two major limitationsof K-Means: One to select efficiently the initial centroids and second to remove the need of giving the number of clusters required as input to the algorithm. The algorithm consists of two separate steps. The first step provides each cluster a center. The next step then adjust the units adaptively by a learning rule. The limitation of this algorithm is the complex computation involved in it.

3. An Extended K-Means Algorithm

The extended K-Means algorithm which does not require an initial estimation and specification of number of clusters (K) as input is proposed below. This algorithm works best with datasets in which all the attributes have numeric values measured on the same scale.

In comparison to other methods discussed in section 2, no additional input except the dataset itself is needed. This algorithm starts by choosing two objects from the dataset as initial centroids. The objects whose sum of attribute values represent the minimum and maximum values are chosen as initial centroids. The rest of the objects in the dataset are distributed in two clusters depending on their Euclidean distance from these two centroids. The algorithm proceeds further by breaking these two initial clusters if the objects in the clusters do not satisfy the objective function representing the intra cluster similarity. This objective function is based on Euclidean distance. The pseudo code of the modified algorithm is discussed below.

Input:

D: The set of n objects with attributes A1, A2, . . . , Am where m = Number of attributes and all the attributes are numeric.

Method:

1) Compute sum of the attribute values of each object.
2) Take objects with minimum and maximum values of the sum as initial centroids.
3) Create two initial partitions (clusters) based on the Euclidean distance between every object and the initial centroids.
4) Find the average of Euclidean distances of the objects from their centroid in each cluster. The Euclidean distance is calculated as:

$$d(x,c) = \sqrt{\sum_{i=1}^{n}(c_i - x_i)^2} \qquad (1)$$

Where $x=(x_1, x_2, x_3,\ldots\ldots x_m)$, $c=(c_1, c_2, c_3,\ldots\ldots c_m)$

Calculate the average of euclidean distances for every cluster as:

$$d_{avg}(C_j) = \frac{1}{n}\sum_{i=1}^{n} d_i \qquad (2)$$

Where d_i=distance of ith object from its centroid in cluster j, n=number of objects in cluster j.

The minimum of the two average distances is taken as d_m.

5) Compute new means (centroids) for the partitions created in step 3.
6) Compute Euclidean distance of every object from the new means (cluster centers) and find the outliers depending on the following objective function:

If $d(x, c) < d_m$ then not an Outlier.

Obtain the transformed clusters.

7) Compute centroids of the transformed clusters.
8) Calculate Euclidean distance of every outlier from the cluster centroids and find the outliers not satisfying the objective function mentioned in step 6. Put back the outliers

satisfying the objective function and obtain transformed clusters. This step is performed to check whether any of the outliers can be adjusted in the existing clusters. Remaining outliers are added to set B.

9) Let B={Y_1,Y_2,…..Y_p} be the set of outliers obtained in step 8.

10) If set B contains one object then

Create a new cluster containing that object

Else

Repeat

a. Assume this set B as a new dataset (D)

b. Perform steps 1 to 10

until (B==Φ)

4. Illustrative Examples

The proposed algorithm is explained using two small datasets. The first dataset is a sample taken from Iris dataset from UCI Machine Repository. The algorithm segregates this dataset of flowers into groups depending upon the sepal length, sepal width, petal length and petal width. The second dataset is a sample taken from Ruspini dataset from UCI Machine Repository which groups the set of points based on the value of their X and Y coordinates.

IRIS Dataset

Table 1. Sample of IRIS Dataset

Sepal length	Sepal width	Petal length	Petal width
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
5.0	3.6	1.4	0.2
4.8	3.0	1.4	0.1
7.0	3.2	4.7	1.4
6.4	3.2	4.5	1.5
6.3	3.3	4.7	1.6
6.7	3.1	4.4	1.4

As per our algorithm:

Step 1. Find the sum of all the attribute values as shown in Table 2. The records with minimum and maximum value of the sum are shaded.

Table 2. The Sum of Attribute Values of Table 1

Sepal length	Sepal width	Petal length	Petal width	Sum
5.1	3.5	1.4	0.2	10.2
4.9	3.0	1.4	0.2	9.5
5.0	3.6	1.4	0.2	10.2
4.8	3.0	1.4	0.1	9.3
7.0	3.2	4.7	1.4	16.3
6.4	3.2	4.5	1.5	15.6
6.3	3.3	4.7	1.6	15.9
6.7	3.1	4.4	1.4	15.6

Step 2: The shaded objects are chosen as initial centroids.

Step 3: After calculating the Euclidean distance of all the objects from the two initial centroids, initial clusters are formed as shown in Table 3. The objects chosen as centroids are shown in bold.

Table 3. Initial Clusters

Cluster1	{(**4.8, 3.0, 1.4, 0.1**), (5.1, 3.5, 1.4, 0.2), (4.9, 3.0, 1.4, 0.2), (5.0, 3.6, 1.4, 0.2)}
Cluster2	{(**7.0, 3.2, 4.7, 1.4**), (6.4, 3.2, 4.5, 1.5), (6.3, 3.3, 4.7, 1.6), (6.7, 3.1, 4.4, 1.4)}

Step 4: Calculate Euclidean distance of every object in cluster1 from its centroid (4.8, 3.0, 1.4, 0.2). The distance of the object (5.1, 3.5, 1.4, 0.2) from the centroid is calculated as:

$$\sqrt{(4.8-5.1)^2+(3.0-3.5)^2+(1.4-1.4)^2+(0.2-0.2)^2} = 0.34$$

The average of all the distance values comes out to be 0.46.

Similarly, the average of all the object distances in cluster2 from its centroid (7.0, 3.2, 4.7, 1.4) is 0.60.
The minimum of the two is taken as d_m. d_m=0.46.

Step 5: Compute new mean (centroid) for cluster1 as:
Centroid cluster1: (4.95, 3.28, 1.4, 0.18)

Similarly, Compute new mean (centroid) for cluster2:
Centroid cluster2: (6.6, 3.2, 4.58, 1.48)

Step 6. No outlier is found in cluster1 and cluster2 according to the objective function mentioned in step 6 of the algorithm. So B= Φ. The algorithm stops here and finally two clusters are obtained:

Cluster1	{(4.8, 3.0, 1.4, 0.2), (5.1, 3.5, 1.4, 0.2), (4.9, 3.0, 1.4, 0.2), (5.0, 3.6, 1.4, 0.2)}
Cluster2	{(7.0, 3.2, 4.7, 1.4), (6.4, 3.2, 4.5, 1.5), (6.3, 3.3, 4.7, 1.6), (6.7, 3.1, 4.4, 1.4)}

These results are compared with the original K-Means results obtained using software RapidMiner with value of K as 2 in Table 4. The results obtained with the proposed algorithm are exactly the same as of original K-Means.

Table 4. Results for Iris Dataset

	K-Means Algorithm	**Extended Algorithm**
Cluster1	{(4.8, 3.0, 1.4, 0.2), (5.1, 3.5, 1.4, 0.2), (4.9, 3.0, 1.4, 0.2), (5.0, 3.6, 1.4, 0.2)}	{(4.8, 3.0, 1.4, 0.2), (5.1, 3.5, 1.4, 0.2), (4.9, 3.0, 1.4, 0.2), (5.0, 3.6, 1.4, 0.2)}
Cluster2	{(7.0, 3.2, 4.7, 1.4), (6.4, 3.2, 4.5, 1.5), (6.3, 3.3, 4.7, 1.6), (6.7, 3.1, 4.4, 1.4)}	{(7.0, 3.2, 4.7, 1.4), (6.4, 3.2, 4.5, 1.5), (6.3, 3.3, 4.7, 1.6), (6.7, 3.1, 4.4, 1.4)}

Ruspini Dataset

The extended algorithm is explained further using a sample of Ruspini dataset.

Table 5. Sample of Ruspini Dataset

X	Y	Sum
10	59	69
24	58	82
27	55	82
28	60	88
30	52	82
31	60	91
32	61	93
36	72	108
32	149	181
35	153	188
33	154	187
78	94	172
97	122	219
98	116	214
98	124	222
99	119	218
99	128	227
101	115	216
108	111	219
110	111	221
108	116	224
111	126	237
115	117	232
117	115	232

As per our algorithm:

Step 1. The sum of the attribute values of each object is shown in the Sum column in Table 5.

Step 2. The shaded objects are chosen as initial centroids (minimum and maximum sum value).

Step 3. After calculating the distance of all the objects from the two initial centroids following clusters were formed:

Cluster1	{**(10, 59),** (24, 58), (27, 55), (28, 60), (30, 52), (31, 60), (32, 61), (36, 72), (78,94)}
Cluster2	{**(111, 126**), (32, 149), (35, 153), (33, 154), (97, 122), (98, 116), (98, 124), (99, 119), (99, 128), (101, 115), (108, 111), (110, 111), (108, 116), (115, 117), (117, 115)}

Step 4. Calculate Euclidean distance of every object in cluster1 from its centroid (10, 59) and calculate the average distance which comes out to be 27.51.

Similarly, the average of all the object distances in cluster2 from its centroid (111, 126) is 27.93.

The minimum of the two is taken as dm. dm=27.51.

Step 5. Compute new mean (centroid) for cluster1 as:
Centroid cluster 1: (32.89, 63.44)

Similarly, compute new mean (centroid) for cluster2:

Centroid cluster2: (90.73, 125.07)

Step 6. In cluster1 there is one outlier: (78, 94)

So cluster1 transforms into:
{(10, 59), (24, 58), (27, 55), (28, 60), (30, 52), (31, 60), (32, 61), (36, 72)}

In cluster2 outliers are: (32, 149), (35, 153), (33, 154)
Transformed cluster2 is:

{(111, 126), (97, 122), (98, 116), (98, 124), (99, 119), (99, 128), (101, 115), (108, 111), (110, 111), (108, 116), (115, 117), (117, 115)}

Step 7. Calculate new centroid of cluster1 and cluster2 after removing outliers as:

Centroid cluster1: (27.25, 59.62)
Centroid cluster2: (105.08, 118.33)

Step 8. Calculate distance of the outliers from the centroids of cluster1 and cluster2. No outlier satisfies the objective function mentioned in step 6 of the algorithm.

Step 9. So set B={(78, 94), (32, 149), (35, 153), (33, 154)}

X	Y	Sum
78	94	172
32	149	181
35	153	188
33	154	187

Step 10. Repeat steps 1-9 on the above dataset taking the shaded objects as initial centroids. Two clusters are formed after repeating the whole process. Finally four clusters are obtained as shown below:

Cluster1	{(10, 59), (24, 58), (27, 55), (28, 60), (30, 52), (31, 60), (32, 61), (36, 72)}
Cluster2	{(111, 126), (97, 122), (98, 116), (98, 124), (99, 119), (99, 128), (101, 115), (108, 111), (110, 111), (108, 116), (115, 117), (117, 115)}
Cluster3	{(78, 94)}
Cluster4	{(35, 153), (32, 149), (33, 154)}

These results are compared with the original K-Means algorithm using software RapidMiner with value of K=4 and the results are shown in Table 6. It is observed that results of both the algorithms are exactly same.

Table 6. Results of Ruspini Dataset

	K-Means Algorithm	Extended Algorithm
Cluster1	{(10, 59), (24, 58), (27, 55), (28, 60), (30, 52), (31, 60), (32, 61), (36, 72)}	{(10, 59), (24, 58), (27, 55), (28, 60), (30, 52), (31, 60), (32, 61), (36, 72)}
Cluster2	{(111, 126), (97, 122), (98, 116), (98, 124), (99, 119), (99, 128), (101, 115), (108, 111), (110, 111), (108, 116), (115, 117), (117, 115)}	{(111, 126), (97, 122), (98, 116), (98, 124), (99, 119), (99, 128), (101, 115), (108, 111), (110, 111), (108, 116), (115, 117), (117, 115)}
Cluster3	{(78, 94)}	{(78, 94)}
Cluster4	{(35, 153), (32, 149), (33, 154)}	{(35, 153), (32, 149), (33, 154)}

5. Results and Discussion

To explore the practical application of our algorithm, we implemented it in C# and tested its performance on real world datasets of different sizes and dimensions. Four standard data sets are tested, namely the Iris, Wine, Breast Tissue and Yeast from UCI Machine Repository. Also a comparative analysis of the performance of the proposed algorithm is done with the original K-Means algorithm, Hierarchical clustering algorithm with Single Linkage Centroid Linkage Complete Linkage and Average Linkage, Fuzzy C-Means, Hierarchical K-means, K-means using CCIA, K-Means suggested by Ahmad and Dey.

The results of our algorithm are compared with the results of the original K-Means using software RapidMiner. These results are discussed in detail in the following paragraphs.

The Deviation in the results is calculated as the ratio percent of differently classified points to total number of points in dataset.

$$\text{Deviation in \%age} = \frac{\text{Number of differently classified points}}{\text{Total number of points}} * 100$$

Comparison on IRIS Data

This dataset contains information about Iris flowers. There are three classes of Iris flowers, namely Iris Setosa, Iris Versicolor and Iris Virginica. The dataset contains 150 objects with 4 attributes. Each class contains 50 objects. The experiment is performed taking K=2 in original K-Means algorithm, as the two clusters out of three predefined clusters are not linearly separable. Figure 1 and Figure 2 show the clusters generated by the original K-Means algorithm and the extended algorithm respectively.

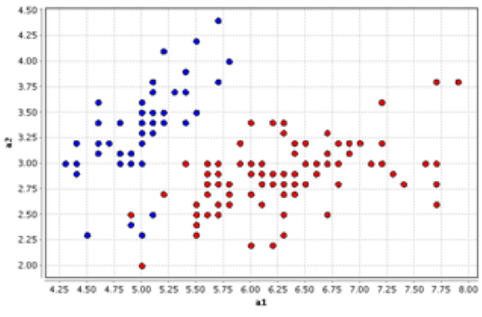

Figure 1. Clusters of Iris Dataset using RapidMiner (K=2)

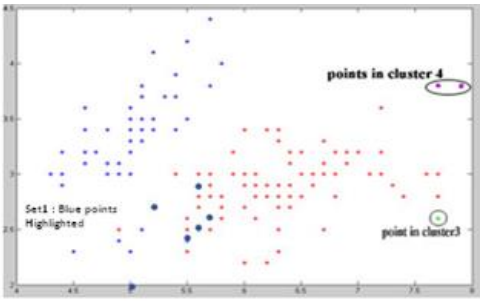

Figure 2. Clusters of Iris Dataset using Extended Algorithm

As can be seen from the figure above, our algorithm generates 4 clusters instead of two generated by RapidMiner, with 3 points in clusters 3 and 4. This is due to the fact that these points lie far from the centroids of the two major clusters generated. Also there are 6 points of cluster plotted in red which are differently classified and are included in the cluster plotted in blue (Set I)**.**

Taking together the 3 plus 6 differently classified points,

$$\text{Deviation} = 9/150*100 = 6\%$$

Table 7 shows the error % of our algorithm and other algorithms for Iris dataset. The error percentage is calculated by taking K=3. For our algorithm the value of K is not provided as input. As two classes out of three have a large overlap the error% increases for the proposed algorithm if we compare the results with the original classification of 50 records in each class.

Table 7. Percentage Error of various algorithms for Iris Dataset (see Arai *et al*. [15])

Algorithm	Error (%)
Single Linkage	32
Centroid Linkage	9.3
Complete Linkage	16
Average Linkage	9.3
Fuzzy C-Means	13.52
K-Means using random initialization	17.70
Hierarchical K-Means	10.67
K-Means using CCIA	11.33
K-Means suggested by Ahmad *et al*.	5.3
Abubaker *et al*. method	29.3
Extended Algorithm	32

Comparison on Wine Data

This dataset contains the chemical analysis of wines grown in a region in Italy but derived from three different cultivars. There are 3 classes and 13 attributes. The dataset consists of 178 objects with 59 objects in class1, 71 objects in class2 and 48 objects in class3.

The data points in Wine dataset are separable based on the value of the 13th attribute in the dataset (Proline). The values in this attribute are measured on a scale which is far higher than the other attributes' scale. The results of classification with K=3 on this dataset show that the third cluster is linearly separable from the other two clusters which merge. So in our experiment we have compared the results of the extended algorithm with original K-Means with K=2. Figure 3 and Figure 4 show the clusters generated by the original K-Means algorithm and the extended algorithm respectively.

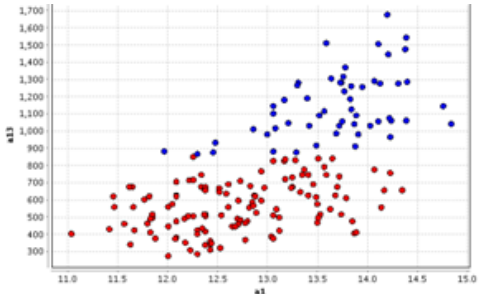

Figure 3. Clusters of Wine Dataset using RapidMiner (K=2)

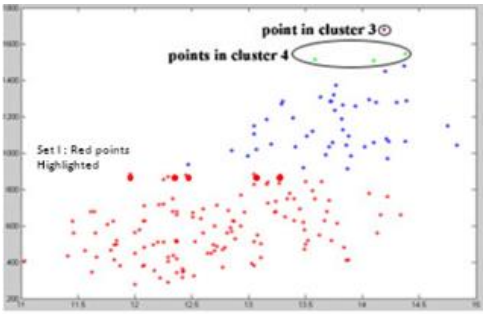

Figure 4. Clusters of Wine Dataset using Extended Algorithm

As can be seen in Fig. 4, 4 clusters are generated in Wine dataset using our algorithm. The 4 points in cluster 3 and cluster 4 are located far from the mean values of two major clusters generated, that is the reason for their going to other clusters.

If we compare these results with those generated in RapidMiner taking K=2 as input, there are another 5 points of cluster plotted in blue which have been differently classified and are included in cluster plotted in red (Set I). These 5 points are better classified in red cluster by our proposed algorithm than the K-Means.

Considering these better classified points in Set I also as differently classified,

$$\text{Deviation} = 9/178*100 = 5.05\%$$

Table 8 shows the comparison of our algorithm with other algorithms for Wine dataset in terms of percentage of error for K=3. The value of error% is high for the proposed algorithm in this case because the two classes out of three have a large overlap.

Table 8: Percentage Error of Various Algorithms for Wine Dataset (see Arai *et al*. [15])

Algorithm	Error (%)
Single Linkage	57.30
Centroid Linkage	38.76
Complete Linkage	32.58
Average Linkage	38.76
Fuzzy C-Means	30.34
K-Means using random initialization	32.52
Hierarchical K-Means	29.78
K-Means using CCIA	5.05
JiMing *et al*. [46] method	30
Extended Algorithm	30.33

Comparison on Breast Tissue Data

This dataset has predefined 6 classes 106 objects. All these six classes are not linearly separable so we have done experiments taking values of K as 2 and 3. Figure 5, 6 and 7 show the results generated by the original K-Means algorithm and the extended algorithm.

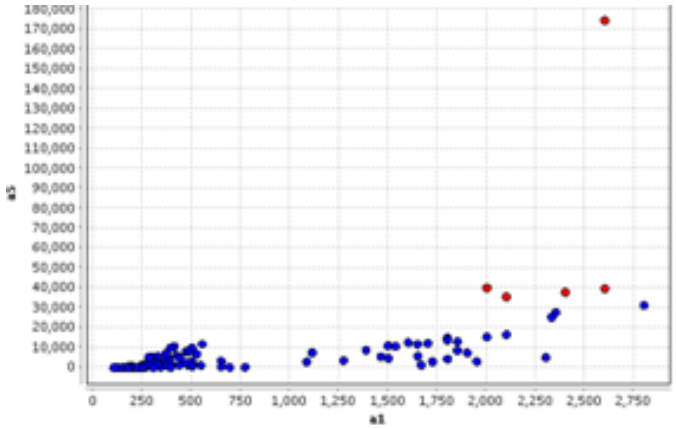

Figure 5. Clusters of Breast Tissue Dataset using RapidMiner (K=2)

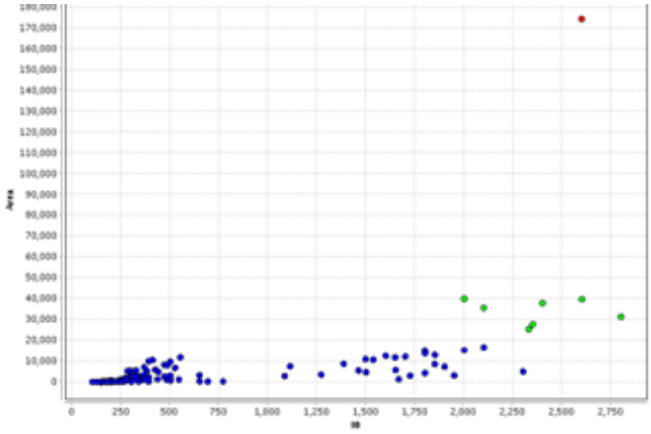

Figure 6. Clusters of Breast Tissue Dataset using RapidMiner (K=3)

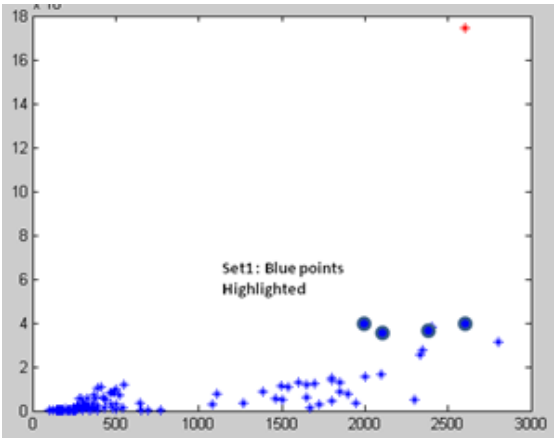

Figure 7. Clusters of Breast Tissue Dataset using Extended Algorithm

As shown in Figure 7, the extended algorithm generates two clusters. If we compare these results with original K-Means taking K=2, there are 4 points plotted in red which are differently classified and are included in the cluster plotted in blue (Set I).

Deviation= 4/106*100 = 3.77%

Comparing the results with original K-Means with K=3 there are 7 points plotted in green which are differently classified and are included in the cluster plotted in blue.

Deviation= 7/106*100 = 6.6%

Comparison on Yeast Data

The Yeast dataset has 1484 records and 10 predefined classes. These 10 classes are not linearly separable so we have done experiments taking value of K=2. The results are shown in Figure 8 and Figure 9.

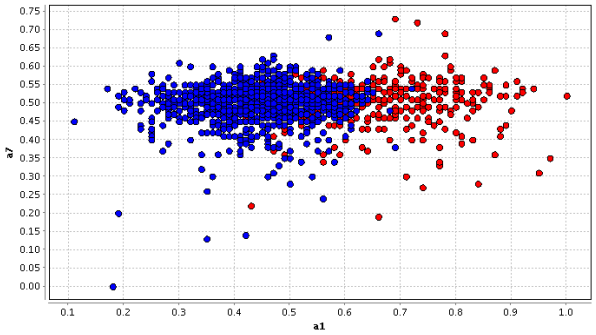

Figure 8. Clusters of Yeast Dataset using RapidMiner (K=2)

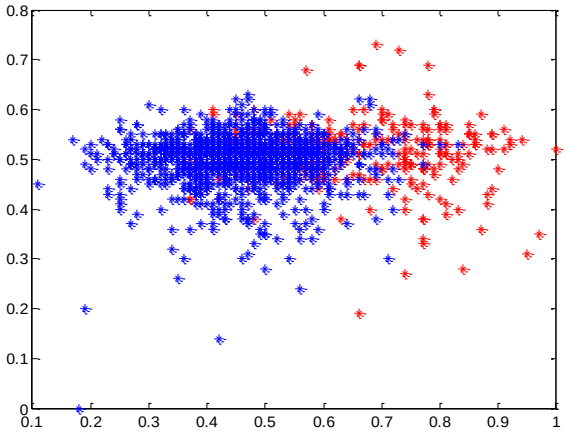

Figure 9. Clusters of Yeast Dataset using Extended Algorithm

The extended algorithm generates seven clusters, with majority of the points (1465) falling in cluster1 and cluster2, putting only the 19 points in rest of the clusters numbered from 3 to 7. As shown in Figure 9 cluster1 plotted in blue has 1118 records and cluster2 plotted in red has 347 records. If we compare the results with RapidMiner results with K=2, there are 97 points of cluster plotted in red which are differently classified and are included in the cluster plotted in blue. Taking these 97 points and 19 points in other four clusters the deviation is calculated as:

$$\text{Deviation} = (116/1484)*100 = 7.81\%$$

6. Conclusion

In this paper a new algorithm based on the basic K-Means algorithm is proposed. The new algorithm overcomes one of the major limitations of basic K-Means algorithm --- providing number of clusters K as input. The algorithm is coded in C# and tested on many real datasets of different sizes and dimensions from UCI machine repository. The results of the experiments show the quality of clusters produced by our algorithm is better than the basic K-Means algorithm and various other algorithms even with no prior-estimated K input and hence we get better clustering under weaker assumption.

References

[1] Rui Xu, Donald Wunsch, "Survey of Clustering Algorithms", IEEE Transactions on Neural Networks, vol. 16, no. 3, **(2005)**, pp. 645-678.
[2] Guojun Gan, Chaoqun Ma, Chaoqun Ma, Jianhong Wu, "Data clustering: theory, algorithms, and applications" , SIAM: Society for Industrial and Applied Mathematics, **(2007)**.
[3] Anil K. Jain, Richard C. Dubes, "Algorithms for clustering data", Prentice-Hall Inc., Upper Saddle River, NJ, USA, **(1988).**
[4] Xindong Wu *et al*., "Top 10 Algorithms in Data Mining", Journal of Knowledge and Information Systems, vol. 14, no. 1, **(2007)**, pp. 1-37.
[5] http://rapidminer.com/
[6] A. Ahmad, and L. Dey, "A K-Mean Clustering Algorithm for Mixed Numeric and Categorical Data", Data & Knowledge Engineering, vol. 63, **(2007)**, pp. 503–527.
[7] http://archive.ics.uci.edu/ml/
[8] Dan Pelleg, Andrew Moore, " Accelerating exact k-means algorithms with geometric reasoning*",* KDD '99 Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, **(1999)**, pp. 277-281.
[9] Kiris Wagstaff, Claire Cardie , Seth Rogers, Stefan Schroedl, "Constrained K-means Clustering with Background Knowledge", Proceedings of the Eighteenth International Conference on Machine Learning, **(2001)**, pp. 577-584.
[10] V.Leela, K.Sakthipriya, R.Manikandan, "A comparative analysis between k-mean and y-means Algorithms in Fisher's Iris data sets" , International Journal of Engineering and Technology, vol. 5, no. 1, **(2013)**, pp. 245-249.
[11] B M Ahamed Shafeeq, K S Hareesha, " Dynamic Clustering of Data with Modified K-Means Algorithm", International Conference on Information and Computer Networks, vol. 27, **(2012)**, pp. 221-225.
[12] Mohamed Abubaker, WesamAshour, "Efficient Data Clustering Algorithms: Improvements over Kmeans" , International Journal of Intelligent Systems and Applications, vol. 5, no. 3, **(2013)**, pp. 37-49.
[13] R. Tibshirani, G. Walther, T. Hastie, " Estimating the number of clusters in a dataset via the gap statistic", Technical Report 208, Department of Statistics, Stanford University, California, **(2000)**.
[14] Yiu-Ming Cheung, "k*-Means: A new generalized k-means clustering algorithm", Pattern Recognition Letters, vol. 24, no. 15, **(2003)**, pp. 2883-2893.
[15] Kohei Arai, Ali Ridho Barakha, 'Hierarchical K-means: An Algorithm for Centroids Initialization for K-Means', Reports of the Faculty of Science and Engineering, Saga University, vol.36, no.1, **(2007)**, pp. 25-31.

