K-Means and Spherical Clusters with Large Variance in Sizes

[1] Fahim A.M. [2] Saake G. [3] Salem A. M. [4]Torkey F. A. [5]Ramadan M. A.

[1]Faculty of Education, Suez Canal University, ahmmedfahim@yahoo.com
[2]Faculty of information, Otto-von-Guericke-University Magdeburg, saake@iti.cs.uni-magdeburg.de
[3]Faculty of Computers & Information, Ain Shams University, absalem@asunet.shams.edu.eg
[4]Faculty of Computers & Information, Minufiya University, fatorkey@Yahoo.com
[5]Faculty of Science, Minufiya University, mramadan@mailer.eun.eg

Abstract

Data clustering is an important data exploration technique with many applications in data mining. The k-means algorithm is well known for its efficiency in clustering large data sets. However, this algorithm is suitable for spherical shaped clusters of similar sizes. The quality of the resulting clusters decreases when the data set contains spherical shaped clusters with large variance in sizes. In this paper, we introduce a simple idea to overcome this problem. Our experimental results reveal that our proposed algorithm produces satisfactory results.

Keywords*: K-Means, Data Clustering, Cluster Analysis.*

1. Introduction

Clustering is a popular approach to implement the partitioning operation. Clustering can be defined as the process of organizing objects in a database into clusters (groups) such that objects within the same cluster have a high similarity, while objects belonging to different clusters have a high dissimilarity [1], [5] and [6]. K-means clustering is one of the most popular data clustering methods because of its simplicity and computational efficiency.

Although the K-means method has a number of advantages over other data clustering techniques, it also has drawbacks; it often converges at a local optimum. The final result depends on the initial starting centroids. Many researchers introduce some methods to select good initial starting centroids [2]. Other researchers try to find the best value for the parameter k that determines the number of clusters. The value of k must be supplied by the user [8]. In recent years many improvements have been proposed and implemented in the K-means method [3].

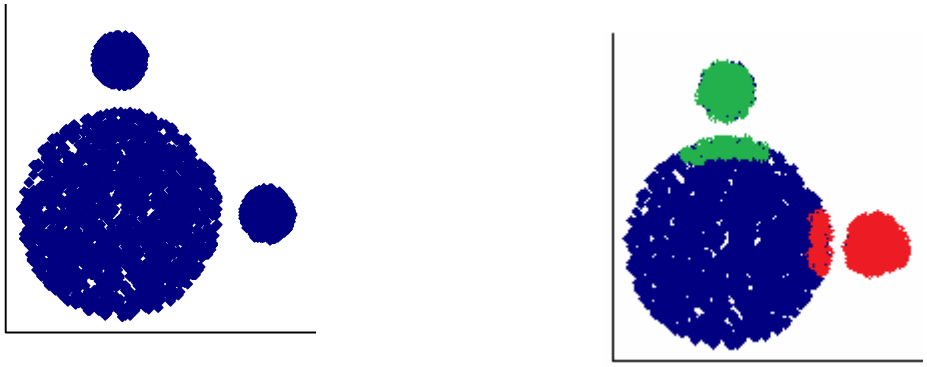
Existing clustering algorithms can be broadly classified into partitional and hierarchical ones [5]. Partitional clustering algorithms attempt to determine k partitions that optimize a certain criterion function. The square-error criterion, defined in (1), is the most commonly used (m_i is the mean of cluster C_i)

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} \left\| p - m_i \right\|^2 \qquad (1)$$

The square-error is a good measure of the within-cluster variation across all the partitions. The objective is to find k partitions that minimize the square-error. Thus, square error clustering tries to make the k clusters as compact and separated as possible, and works well when clusters are compact clouds rather well separated from one another[4]. However, when there are large differences in the sizes or geometries of different clusters, as illustrated in Figure 1, the square-error method could split large clusters to minimize the square-error. In this figure the square-error is larger for the three separate clusters in (a) than for the three clusters in (b), where the big cluster is split into three portions, one of which is merged with the two smaller clusters. The reduction in

square-error for (b) is due to the fact that the slight reduction in square error due to splitting the large cluster is weighted by many data points in the large cluster. We propose a simple idea to solve this problem.

There is a very large number of clustering algorithms but we focus on the k-means algorithm proposed in this paper. A simple function is added to the K-means algorithm making it able to discover clusters with large variance in size with small separation between clusters. So we review the k-means in section 2, discuss our idea in section 3, present some experimental results in section 4 and conclude with section 5.



(a) (b)

Figure 1: Splitting a large cluster by k-means algorithm

2. k-Means Clustering

The general algorithm introduced in [7] is referred to as k-means. Since then it has become widely popular and is classified as a partitional or non-hierarchical clustering method [5]. It is defined as follows: given a dataset D = $\{x_1, \ldots, x_n\}$ of n numerical data points, a natural number k < n, and a distance measure d which is L_2 distance (Euclidean distance). The k-means algorithm aims at finding a partition C of D into k non-empty disjoint clusters $C_1, \ldots, C_k$ with $C_1 \cap C_2 = \varphi$ and $\bigcup_{i=1}^{k} C_i = D$ such that the overall sum of the squared distances between data points and their cluster centers is minimized. This algorithm works as follows:

```
MSE=largenumber
Select initial cluster centroids {m_j}_j^k=1
Do
    OldMSE=MSE;
    MSE1=0;
    For j=1 to k
        m_j=0; n_j=0;
    endfor
    For i=1 to n
        For j=1 to k
            Compute squared Euclidean
            distance d^2(x_i, m_j);
        endfor
        Find the closest centroid m_j to x_i;
        m_j=m_j+x_i; n_j=n_j+1;
        MSE1=MSE1+d^2(x_i, m_j);
    endfor
    For j=1 to k
        n_j=max(n_j, 1); m_j=m_j/n_j;
    endfor
    MSE=MSE1;
while (MSE<OldMSE)
```

Figure 2: *k*-means algorithm

It randomly selects k points (centroids) from the dataset as initial solution for the clustering problem, and assigns each point to the nearest centroid according to the distance measure. After distributing points over the centroids, the algorithm locates the new position of the centroids by taking the average mean of points in each cluster. Then the algorithm goes to the next iteration until there is no change at the centroids or maximum number of iteration is performed. Figure 2 shows the k-means algorithm.

3. Refinement the Final Clusters

The k-means algorithm is a popular clustering algorithm and has its application in data mining, image segmentation [8], bioinformatics and many other fields. This algorithm has many advantages (simplicity, speed convergence to final means, scalable and easily implemented). Its disadvantages are the following: the final means depend on the initial starting means, is sensitive to outliers, requires the number of clusters (the value of k) in advance and it works well with spherical shaped cluster of similar size. In this section we present how to make this algorithm work well with spherical shaped cluster of any size. According to our proposed method, first we find the distances between the means as a result from the k-means algorithm. For each cluster we calculate its average radius by dividing the sum of squared error of its points from its representative by the number of points assigned to it. We search for the largest cluster (having the largest average radius) and test whether this cluster has some portion merged with other clusters. At the first time you can say if the summation of the two radiuses is larger than the distance between the two clusters. Then there is a portion of the larger cluster merged with the other cluster. So we can redistribute the points in the smaller cluster only over the two clusters. But this formula is not suitable at all. Since the mean of cluster is the center of gravity of the points. So for the large cluster in figure 1 the average radius is larger than the actual radius. On the other hand, the average radius of a smaller cluster is larger than the actual radius. Why does it happen? This occurs because the means of the smaller cluster is attracted toward the large cluster, since the objective of the k-means is to get the smallest value for the squared error function in equation 1. So, we use the following formula to test the partition of the large cluster. Where *ml* and *ms* are the means of the large and the small cluster respectively, *d* is the dimensionality of the data, *L* and *S* refer to the large and small cluster respectively.

$$\text{Means\_distance} = \sqrt{\sum_{i=1}^{d} (ml_i - ms_i)^2} \qquad (2)$$

$$\text{Sum\_of\_radius} = (\text{radius}(L) + \text{radius}(S))*0.80 \qquad (3)$$

If the sum_of_radii is larger than or equal to the means_distance and at the same time the ratio between the two radii is smaller than 0.90, this condition is used to insure that there is large difference in size, then some portion of the larger cluster is merged with the smaller cluster. In this case we must redistribute the points in the smaller cluster to return the misclassified points to the large cluster. How can we redistribute the points in the smaller cluster? To do this operation, we take the average of the two means as new means (*Av_mean* = (*ml*+*ms*)/2). This new mean is located in the large cluster, and at the mid distance between the two clusters centers. We redistribute the points in the small cluster over the new mean (*Av_mean*) and its original mean (*ms*). The means of the two clusters are shifted. So, we use the next formula to redistribute the points of the small cluster. We add a small value to the right hand side of relation 4, since we expect a small separation between the two clusters and the mean of the smaller cluster is attracted to the large cluster. We multiply the ratio between the two radiuses by 0.80 since the radius of the small cluster has error percent larger than the other cluster

$$\text{If } (\text{Dis}(p_i, Av\_mean) <= \text{Dis}(p_i, ms) + \text{radius}(L)*0.80 / \text{radius}(S)) \qquad (4)$$

If the formula 4 is true then the point p_i is moved to the large cluster, otherwise it remains in the smaller cluster. After redistribution of all points in the small cluster, we recalculate the new means for the two clusters. All these processes are repeated for all clusters. So the final means of our proposed method are better than those produced first from the k-means algorithm. The following figure 3 shows the proposed function added to the k-means algorithm to improve the final results.

```
For i=1 to k
    For j=i+1 to k
        Compute Euclidean distance d (m_i, m_j);
     Next j
 Next i

For j=1 to k/2
 Find the largest cluster L
 For i=1 to k
   If (radius(L)> radius(I)
     S=I
         Sum_of_radius = (radius(L) + radius(S) )*0.80

      If (Sum_of_radius >= d (m_L, m_S)&&( radius(S) / radius(L)) < 0.90)
          Av_mean=(m_L+m_S)/2
          Redistribute points represented by m_S over the  two means m_S and
            Av_mean.
          All points assigned to Av_mean are moved to the cluster m_L.
           Find the new mean for the cluster m_S.
       Endif
     Endif
  Next i
 Find the new mean for the cluster m_L.
Next j
```

Figure 3: Refinement process of the final results of the *k*-means algorithm

4. Experimental Results

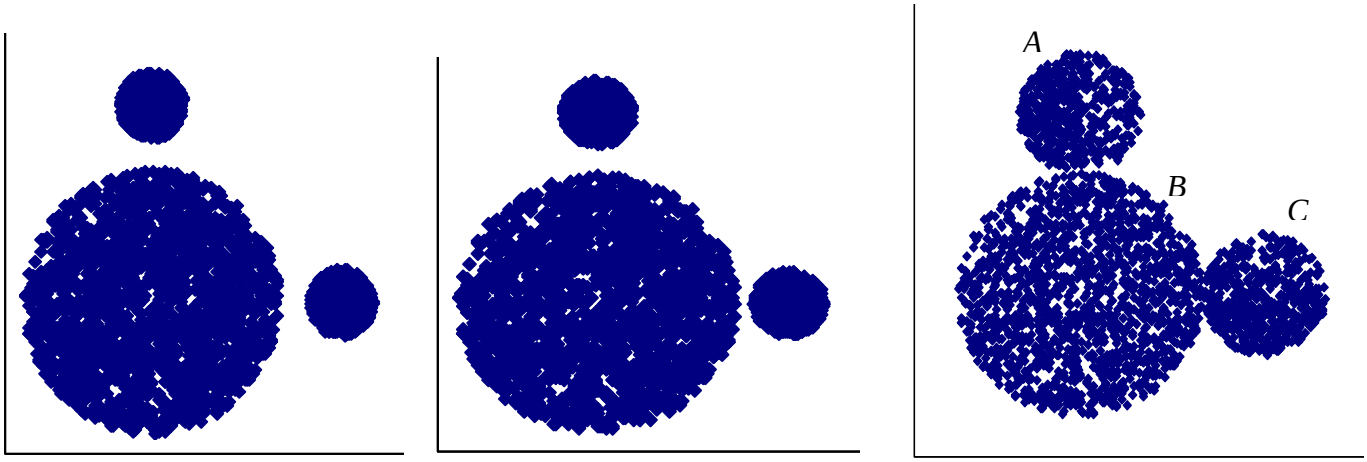
In this section, we present some experimental evaluation of our proposed algorithm, which reveals a great improvement in the k-means algorithm when the dataset contains spherical shaped clusters with large variance in their size.

We have created many 2-dimensional datasets that contain spherical clusters of different sizes. We present here three of them. All these datasets contain three clusters as shown in Figure 4. Table 1 presents the exact number of points in each cluster. Also we present the exact number of points in each cluster found by the k-means and by our proposed algorithm.

Table1: comparison between the results from the k-means and our proposed method

Data set	Exact clusters	k-mean clusters	k-means error	Proposed clusters	proposed error
Dataset1	1815	1210	605 points 19.158 %	1815	0 points 0.0 %
	683	973		683	
	660	975		660	
Dataset2	1582	1025	557 points 19.030 %	1582	0 points 0.0 %
	703	1015		703	
	642	887		642	
Dataset3	1582	1043	539 points 20.256 %	1585	7 points 0.263 %
	557	816		552	
	522	802		524	

The experimental results in this table show the great improvement at the final clusters discovered by our proposed algorithm. When we examine the percent of error in our proposed algorithm, we find that, our method produces the exact clusters in dataset1 and dataset2, because there is a separation between clusters. But there is a small error at clusters discovered from dataset3, note that, there is no separation between clusters. From table1 and figure 4 you can see that there are 7 points misclassified: cluster *B* takes 5 points from cluster *C*, cluster *A* takes 2 points from cluster *B*.



Dataset1 Dataset2 Dataset3

Figure 4: data set we used to test our proposed method

4. 1 Time Complexity

As we know, the time complexity of the k-means algorithm is $O(nki)$; where n is the number of data points in the dataset, k is the number of clusters and i is the number of iteration. Since we use the k-means and then we apply our procedure, the time complexity is equal to the summation of the two times. At first, our method finds the distances between the pair of k clusters so this operation requires $O(k^2)$. Then we search for the largest cluster, that requires $O(k)$, at most the points of 3 or 4 clusters will be redistributed over their means and the average means, this operation requires $O(2mh)$; 2 is the two means, m is the number of points in the cluster and h is the number of clusters, we redistribute their points. Since we redistribute the points in the smaller cluster so $m < n/2k$, and h is very small, we can say h=4 at most. So the time complexity added to the k-means is very small compared with the time complexity of the k-means itself. So, the time is $O(k^2+ mh)$, $k < n$. At the end the time complexity is $O(nki +k^2 +mh)$.

5. Conclusion

This paper represents a new procedure added to the end of the k-means clustering algorithm. The objective of this procedure is to refine the results of the k-means. The procedure is optional. It is strongly recommended to use after the k-means, especially when the dataset contains spherical clusters with large difference in their sizes. Our experimental results show that the proposed method improves the quality of the resulting clusters.

References

[1] Anderberg M R "Cluster Analysis for Applications", Academic Press, 1973.

[2] Bradley P. S., Fayyad U. M. "Refining Initial Points for K-Means Clustering", Proc. of the 15th International Conference on Machine Learning (ICML98), J. Shavlik (ed.), pp. 91-99. Morgan Kaufmann, San Francisco, 1998.

[3] Fahim A. M., Salem A. M., Torkey F. A. and Ramadan M. "An efficient enhanced k-means clustering algorithm", Journal of Zhejiang University SCIENCE A, vol 7(10), pp. 1626-1633, 2006.

[4] Guha, S., Rastogi, R., Shim, K., "CURE: An Efficient Clustering Algorithms for Large Databases". Proc. ACM SIGMOD Int. Conf. on Management of Data. Seattle, WA, pp.73-84, 1998.

[5] Jain A. K. and, Dubes R. C., "Algorithms for Clustering Data", Prentice Hall, 1988.

[6] Kaufman L. and Rousseeuw P., "Finding Groups in Data: An Introduction to Cluster Analysis": Wiley, 1990.

[7] MacQueen J.B. "Some methods for classification and analysis of multivariate observations". Proc. 5th Symp. Mathematical Statistics and Probability, Berkelely, CA, Vol. 1, pp. 281–297, 1967.

[8] Ray S. and Turi R. H., "Determination of number of clusters in k-means clustering and application in colour image segmentation.", in Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques, pp. 137-143, 1999.

Article received: 2008-03-13

This paper contains 4 Figures and 1 table