International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:1, No:11, 2007

Enhancing K-Means Algorithm with Initial Cluster Centers Derived from Data Partitioning along the Data Axis with the Highest Variance

S. Deelers, and S. Auwatanamongkol

Abstract**—**In this paper, we propose an algorithm to compute initial cluster centers for *K*-means clustering. Data in a cell is partitioned using a cutting plane that divides cell in two smaller cells. The plane is perpendicular to the data axis with the highest variance and is designed to reduce the sum squared errors of the two cells as much as possible, while at the same time keep the two cells far apart as possible. Cells are partitioned one at a time until the number of cells equals to the predefined number of clusters, *K*. The centers of the *K* cells become the initial cluster centers for *K*-means. The experimental results suggest that the proposed algorithm is effective, converge to better clustering results than those of the random initialization method. The research also indicated the proposed algorithm would greatly improve the likelihood of every cluster containing some data in it.

Keywords**—**Clustering algorithm, *K*-means algorithm, Data partitioning, Initial cluster centers.

I. Introduction

CLUSTERING is an important tool for a variety of applications in data mining, statistical data analysis, data compression, and vector quantization. The goal of clustering is to group data into clusters such that the similarities among data members within the same cluster are maximal while similarities among data members from different clusters are minimal.

K-means is a well known prototype-based, partitioning clustering technique that attempts to find a user-specified number of clusters (*K*), which are represented by their centroids. The *K*-means algorithm is as follows:

1. Select initial centers of the *K* clusters. Repeat steps 2 through 3 until the cluster membership stabilizes.
2. Generate a new partition by assigning each data to its closest cluster centers.
3. Compute new cluster centers as the centroids of the clusters.

S. Deelers was with the School of Applied Statistics at the National Institute of Development Administration, Thailand. He is now with the Faculty of Management Science, Silpakorn University, Phetchaburi Information Technology Campus, 76120, Thailand (e-mail: sirichai@su.ac.th).

S. Auwatanamongkol is with the School of Applied Statistics, National Institute of Development Administration, Khong-chan, Bangkapi, Bangkok 10240, Thailand (e-mail: surapong@as.nida.ac.th).

Although *K*-means is simple and can be used for a wide variety of data types, it is quite sensitive to initial positions of cluster centers. The final cluster centroids may not be optimal ones as the algorithm can converge to local optimal solutions. An empty cluster can be obtained if no points are allocated to the cluster during the assignment step. Therefore, it is quite important for *K*-means to have good initial cluster centers.

The algorithms for initializing the cluster centers for *K*-means have been proposed in the past. In this paper, some of the more recent proposals are reviewed [2, 3, and 7].

Bradley and Fayyad (1998) proposed the refinement algorithm that builds a set of small random sub-samples of the data, then clusters data in each sub-samples by *K*-means. All centroids of all sub-samples are then clustered together by *K*-means using the *K* centroids of each sub-sample as initial centers. The centers of the final clusters that give minimum clustering error are to be used as the initial centers for clustering the original set of data using *K*-means algorithm.

Likas *et al.* (2003) proposed the global *k*-means clustering algorithm that constructs initial centers by recursively partitioning data space into disjointes subspaces using a *k-d* trees method. The cutting hyperplane used in the method is defined as the plane that is perpendicular to the highest variance axis derived by principal component analysis. The partitioning is performed until each of the leaf node (bucket) contains less than a predefined number of data instances (bucket size) or the predefined number of buckets have been created. The centroids of data in the final buckets are then used as initial centers for *K*-means

S.S. Khan and A. Ahmad (2004) proposed cluster center initialization algorithm (CCIA) based on considering values for each attribute of the given data set. This can provide some information leading to a good initial cluster center. The algorithm can be described as follows.

1. Compute mean (μ_j) and standard deviation (σ_j) for every j^{th} attribute values.
2. Compute percentile Z_1, Z_2,…, Z_k corresponding to area under the normal curve from $-\infty$ to $(2s-1)/2k$, $s=1, 2, \ldots, k$ (clusters).
3. Compute attribute values $x_s = z_s * \sigma_j + \mu_j$ corresponding to these percentiles using mean and standard deviation of the attribute.
4. Perform *K*-means to cluster data based on j^{th} attribute values

using x_s as initial centers and assign cluster labels to every data.

5. Repeat the steps of 3-4 for all attributes (l).
6. For every data item t create the string of the class labels $P_t = (P_1, P_2,\ldots, P_l)$ where P_j is the class label of t when using the ^{jth} attribute values for step 4 clustering.
7. Merge data item which have the same pattern string P_t yielding K' clusters. The centroids of the K' clusters are computed. If $K' > K$, apply *Merge- DBMSDC* (Density-based Multi Scale Data Condensation) algorithm [6] to merge these K' clusters into K clusters.
8. Find the centroids of the K clusters and use the centroid as initial centers for clustering the original dataset using K-Means.

Although the mentioned initialization algorithms can help finding good initial centers for some extent, they are quite complex and some use the K-Means algorithm as part of their algorithms, which still need to use the random method for cluster center initialization.

II. Proposed Algorithm

This story focus on the initialization of cluster centers for K-means. The proposed algorithm follows a novel approach that performs data partitioning along the data axis with the highest variance. The approach has been used successfully for color quantization [8]. The data partitioning tries to divide data space into small cells or clusters where intercluster distances are large as possible and intracluster distances are small as possible.

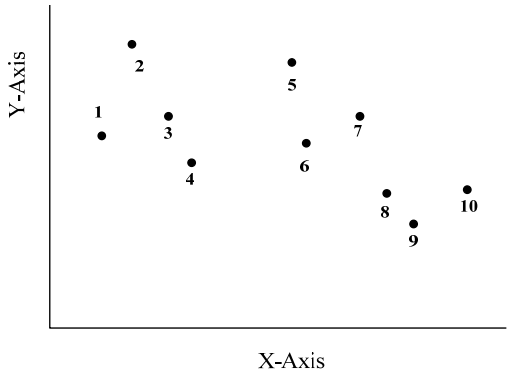

Fig. 1 Diagram of ten data points in 2D, sorted by its X value, with an ordering number for each data point

For instance, consider Fig. 1. Suppose ten data points in 2D data space are given.

The goal is to partition the ten data points in Fig. 1 into two disjoint cells where the sum of total clustering errors of the two cells is minimal, see Fig. 2. Suppose a cutting plane perpendicular to X-axis will be used to partition the data. Let C_1 and C_2 be the first cell and the second cell respectively and $\bar{c}_1$ and $\bar{c}_2$ be the cell centroids of the first cell and the second cell, respectively. The total clustering error of the first cell is thus computed by:

$$\sum_{c_i \in C_1} d(c_i, \bar{c}_1) \quad (1)$$

and the total clustering error of the second cell is thus computed by:

$$\sum_{c_i \in C_2} d(c_i, \bar{c}_2) \quad (2)$$

where c_i is the i^{th} data in a cell. As a result, the sum of total clustering errors of both cells are minimal (as shown in Fig. 2.)

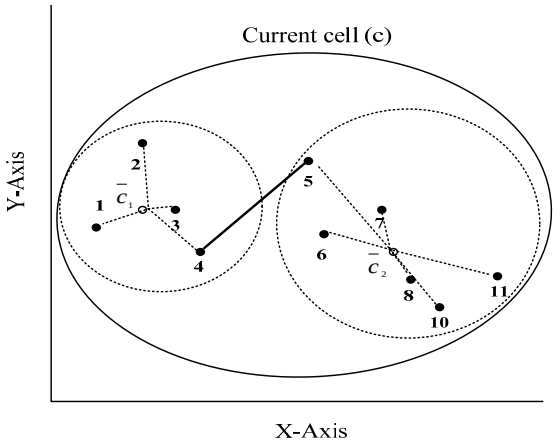

Fig. 2 Diagram of partitioning a cell of ten data points into two smaller cells, a solid line represents the intercluster distance and dash lines represent the intracluster distance

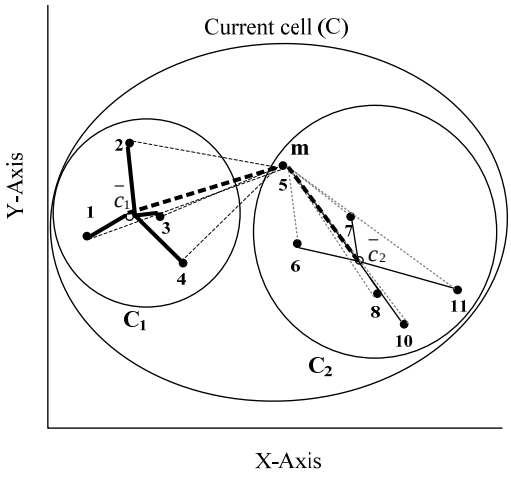

Fig. 3 Illustration of partitioning the ten data points into two smaller cells using m as a partitioning point. A solid line in the square represents the distance between the cell centroid and a data in cell, a dash line represents the distance between m and data in each cell and a solid dash line represents the distance between m and the data centroid in each cell

The partition could be done using a cutting plane that passes through m. Thus

$$d(c_i, \bar{c}_1) \le d(c_i, c_m) + d(\bar{c}_1, c_m) \text{ and}$$
$$d(c_i, \bar{c}_2) \le d(c_i, c_m) + d(\bar{c}_2, c_m) \quad (3)$$

(as shown in Fig. 3). Thus

$$\sum_{c_i \in c_1} d(c_i, \bar{c}_1) \le \sum_{c_i \in c_1} d(c_i, c_m) + d(\bar{c_1}, c_m) \cdot |C_1| \text{ and}$$
$$\sum_{c_i \in c_2} d(c_i, \bar{c}_2) \le \sum_{c_i \in c_2} d(c_i, c_m) + d(\bar{c_2}, c_m) \cdot |C_2| \quad (4)$$

We will call *m* as the partitioning data point where $|C_1|$ and $|C_2|$ are the numbers of data points in cluster C_1 and C_2 respectively. The total clustering error of the first cell can be minimized by reducing the total discrepancies between all data in first cell to *m*, which is computed by:

$$\sum_{c_i \in C_1} d(c_i, c_m) \quad (5)$$

The same argument is also true for the second cell. The total clustering error of the second cell can be minimized by reducing the total discrepancies between all data in second cell to *m*, which is computed by:

$$\sum_{c_i \in C_2} d(c_i, c_m) \quad (6)$$

where $d(c_i, c_m)$ is the distance between *m* and each data in each cell. Therefore the problem to minimize the sum of total clustering errors of both cells can be transformed into the problem to minimize the sum of total clustering error of all data in the two cells to *m*.

The relationship between the total clustering error and the clustering point may is illustrated in Fig. 4, where the horizontal-axis represents the partitioning point that runs from 1 to *n* where *n* is the total number of data points and the vertical-axis represents the total clustering error. When *m=0*, the total clustering error of the second cell equals to the total clustering error of all data points while the total clustering error of the first cell is zero. On the other hand, when *m=n*, the total clustering error of the first cell equals to the total clustering error of all data points, while the total clustering error of the second cell is zero.

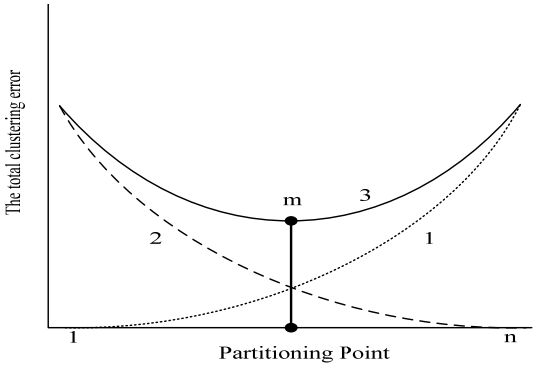

Fig. 4 Graphs depict the total clustering error, lines 1 and 2 represent the total clustering error of the first cell and second cell, respectively, Line 3 represents a summation of the total clustering errors of the first and the second cells

A parabola curve shown in Fig. 4 represents a summation of the total clustering error of the first cell and the second cell, represented by the dash line 2. Note that the lowest point of the parabola curve is the optimal clustering point (*m*). At this point, the summation of the total clustering error of the first cell and the second cell are minimum.

Since time complexity of finding the optimal point *m* is $O(n^2)$, we then use the distances between adjacent data along the X-axis to find the approximated point of *n* but with time of *O(n)*.

Let $D_j = d(c_j, c_{j+1})^2$ be the squared Euclidean distance of adjacent data points along the X-axis.

If *i* is in the first cell then $d(c_m, c_i) \le \sum_{j=i}^{m} D_j$. On the one hand, if *i* is in the second cell then $d(c_m, c_i) \le \sum_{j=m}^{i} D_j$ (as shown in Fig. 5).

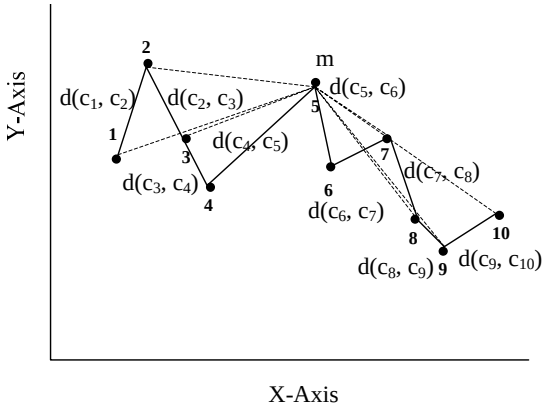

Fig. 5 Illustration of ten data points, a solid line represents the distance between adjacent data along the X-axis and a dash line represents the distance between m and any data point

The task of approximating the optimal point (*m*) in 2D is thus replaced by finding *m* in one-dimensional line as shown in Fig. 6.

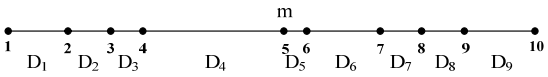

Fig. 6 Illustration of the ten data points on a one-dimensional line and the relevant D_j

The point (*m*) is therefore a centroid on the one-dimensional line (as shown in Fig. 6), which yields

$$\sum_{i=1}^{m-1} d(c_m, c_i) \approx \sum_{i=m}^{n} d(c_m, c_i)$$

Let $dsum_i = \sum_{j=1}^{i} D_j$ and a *centroidDist* can be computed by:

$$centroidDist = \frac{\sum_{i=1}^{n} dsum_i}{n} \quad (7)$$

Therefore, the total clustering errors of the two smaller cells partitioned by the plane passing through the data point nearest to *centroidDist* are similar.

It is possible to choose either the X-axis or Y-axis as the principal axis for data partitioning. However, data axis with the highest variance will be chosen as the principal axis for data partitioning. The reason is to make the inter distance between the centers of the two cells as large as possible while the sum of total clustering errors of the two cells are reduced from that of the original cell. To partition the given data into *k* cells, we start with a cell containing all given data and partition the cell into two cells. Later on we select the next cell to be partitioned that yields the largest reduction of total clustering errors (or Delta clustering error). This can be defined as *Total clustering error of the original cell – the sum of Total clustering errors of the two sub cells of the original cell*. This is done so that every time we perform a partition on a cell, the partition will help reduce the sum of total clustering errors for all cells, as much as possible.

We can now use the partitioning algorithm to partition a given set of data into *k* cells. The centers of the cells can then be used as good initial cluster centers for the *K*-means algorithm. Following are the steps of the proposed algorithm.

1. Let cell *c* contain the entire data set.
2. Sort all data in the cell *c* in ascending order on each attribute value and links data by a linked list for each attribute.
3. Compute variance of each attribute of cell *c*. Choose an attribute axis with the highest variance as the principal axis for partitioning.
4. Compute squared Euclidean distances between adjacent data along the data axis with the highest variance $D_j = d(c_j, c_{j+1})^2$ and compute the $dsum_i = \sum_{j=1}^{i} D_j$
5. Compute centroid distance of cell *c*:
$$centroidDist = \frac{\sum_{i=1}^{n} dsum_i}{n}$$
Where $dsum_i$ is the summation of distances between the adjacent data.
6. Divide cell *c* into two smaller cells. The partition boundary is the plane perpendicular to the principal axis and passes through a point *m* whose $dsum_i$ approximately equals to *centroidDist*. The sorted linked lists of cell *c* are scanned and divided into two for the two smaller cells accordingly
7. Compute Delta clustering error for *c* as the total clustering error before partition minus total clustering error of its two sub cells and insert the cell into an empty Max heap with Delta clustering error as a key.
8. Delete a max cell from Max heap and assign it as a current cell.
9. For each of the two sub cells of *c*, which is not empty, perform step 3 - 7 on the sub cell.
10. Repeat steps 8 - 9. Until the number of cells (Size of heap) reaches *K*.
11. Use centroids of cells in max heap as the initial cluster centers for *K*-means clustering

III. Experiments and Results

We evaluated the proposed algorithm on 10 data sets from UCI Machine Learning Repository [4]. We compared clustering results achieved by the *K*-Means algorithm using random initial centers and initial centers derived by the proposed algorithm. The clustering results of *K*-means using random initial centers are the average results over 10 runs since each run gives different results.

The measurements we used to compare the clustering results were

1) The sum of the squared error distances between the data and the centroid of their clusters (*SSE*). The *SSE* results on 10 UCI data sets are shown in Fig. 7 and 8.

2) Entropy to measure impurity of each cluster

$$E = -\sum_{j=1}^{c} P_j \log P_j \quad (8)$$

where *c* is number of classes of data, P_j is the proportion of data of class *j* in a given cluster. The averaged Entropy for all clusters are then used for comparison. The averaged Entropy on 10 UCI data sets are shown in Fig. 9 and 10.

From the two measurements we can see that the proposed algorithm outperform the random initialization algorithm in most cases. The proposed algorithm also performs much better than the random initialization algorithm as the required number of clusters increases. The execution times of *K*-Means when using proposed algorithm were also much less than the average execution times of *K*-Means when using random initialization algorithm, especially for large data sets e.g. Letter, Pendigits and Optdigits. This may be due to the initial cluster centers generated by the proposed algorithm are quite close to the optimal solutions. The execution time comparisons for the three UCI data sets are shown in Fig.11. We also compare the clustering results from proposed algorithm with the results from the Clustering Center Initialization Algorithm (CCIA). Fig. 12 shows the clustering results in terms of classification error (%) when a class of data in a cluster is predicted to be the majority class of data in the cluster.

It can be seen that proposed algorithm performances are comparable to the CCIA. However, the proposed algorithm is much simpler to implement than CCIA.

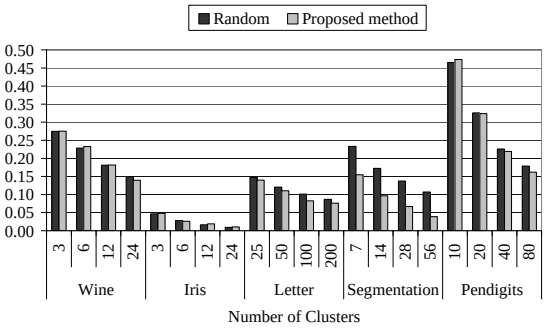

Fig. 7 SSE results on UCI data sets: Wine, Iris, Letter, Segmentation and Pendigits

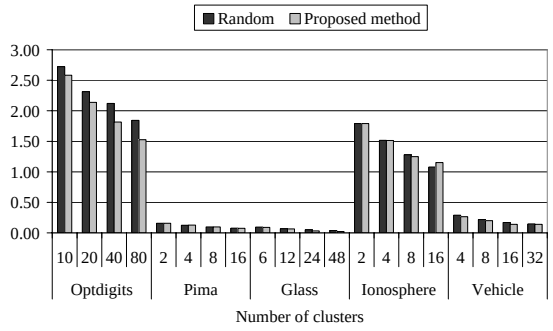

Fig. 8 SSE results on UCI data sets: Optdigits, Pima, Glass, Ionosphere and Vehicle

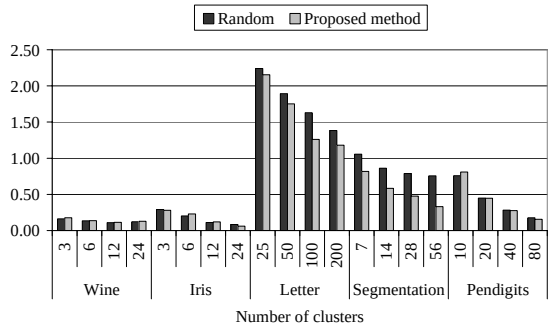

Fig. 9 Entropy results on UCI data sets: Wine, Iris, Letter, Segmentation and Pendigits

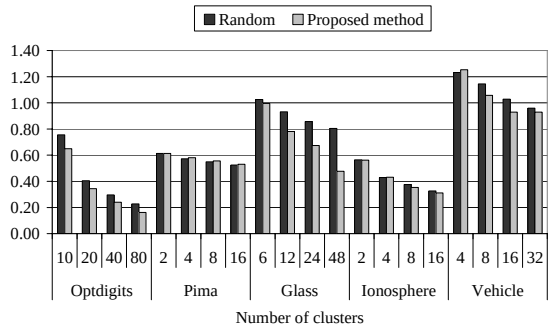

Fig. 10 Entropy results on UCI data sets: Optdigits, Pima, Glass, Ionosphere and Vehicle

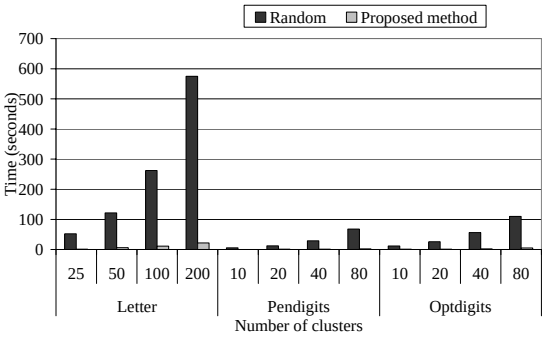

Fig. 11 Execution time results on three large UCI data sets: Letter, Pendigits and Optdigits

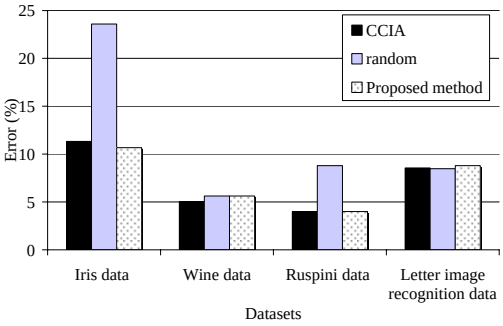

Fig. 12 Classification error comparisons among the three methods, Cluster Center Initialization Algorithm (CCIA), Random Initialization Algorithm and Proposed Algorithm

IV. Conclusion

A novel initialization algorithm of cluster centers for *K*-means algorithm has been proposed. The algorithm was based on the data partitioning algorithm used for color quantization. A given data set was partitioned into *k* clusters in such a way that the sum of the total clustering errors for all clusters was reduced as much as possible while inter distances between clusters are maintained to be as large as possible.

The proposed algorithm is very effective, converges to better clustering results and almost all cluster have some data in its. The experimental results show that the proposed algorithm performs better than random initialization in most of the experimental cases and can reduce running time of K-Means significantly for large data sets. The performances of proposed algorithm are also comparable to the CCIA however the proposed algorithm is much simpler and easier to implement.

REFERENCES

[1] A. Jain, and R. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs. NJ, 1998.
[2] A. Likas, N. Vlassis and J.J. Verbeek, "The Global k-means Clustering algorithm", Pattern Recognition , Volume 36, Issue 2, 2003, pp. 451-461.
[3] P.S. Bradley and U.M. Fayyad, "Refining initial points for K-means Clustering", Proceeding of The Fifteenth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, CA, 1998, pp. 91-99.
[4] C.L. Blake, C.J. Merz. UCI Repository of machine learning databases. University of California, Irvine, Department of Information and Computer Science, 1998.
[5] J. Han and M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, San Diego, 2001.
[6] P. Mitra, C.A. Murthy, S.K. Pal, "Density based multiscale data condensation", IEEE Trans, Pattern Anal, Machine Intell, 24 (6), 2002, pp. 734–747.
[7] S. S. Khan and A. Ahmad, "Cluster Center Initialization for K-mean Clustering", Pattern Recognition Letters, Volume 25, Issue 11, 2004, pp. 1293-1302
[8] Y. Sirisathitkul, S. Auwatanamongkol and B. Uyyanonvara, "Color image quantization using distances between adjacent colors along the color axis with highest color variance", Pattern Recognition Letters, Volume 25, Issue 9, 2004, pp. 1025-1043.