Journal of Computer Applications (JCA)
ISSN: 0974-1925, Volume IV, Issue 2, 2011

Refinement of K-Means Clustering Using Genetic Algorithm

K.Arun Prabha [a,*], **R.Saranya** [b,1]

Abstract--- **K-means clustering is a popular clustering algorithm based on the partition of data. However, there are some shortcomings of it, such as its requiring a user to give out the number of clusters at first, and its sensitiveness to initial conditions, and second it can only find linearly separable clusters. There are a lot of variations of the k-means clustering algorithm. Kernel k-means is an extension of the standard k–means algorithm to solve the second limitation of k-means clustering. Recent attempts have adapted the k-means clustering algorithm as well as genetic algorithms based on rough sets to find interval sets of clusters. And an important point is, so far, the researchers haven't contributed to improve the cluster quality once it is clustered. In this paper, we have proposed a new context to improve the cluster quality from k-means clustering using Genetic Algorithm (GA). The performance is analyzed and compared with the standard and kernel k-means clustering in medical domain.**

Index Terms **– K-means, Kernel K-means, Genetic Algorithm.**

I. INTRODUCTION

Clustering techniques have become very popular in a number of areas, such as engineering, medicine, biology and data mining [1,2]. A good survey on clustering algorithms can be found in [3]. The k-means algorithm [4] is one of the most widely used clustering algorithms. The algorithm partitions the data points (objects) into C groups (clusters), so as to minimize the sum of the (squared) distances between the data points and the center (mean) of the clusters. In spite of its simplicity, the k-means algorithm involves a very large number of nearest neighbor queries. The high time complexity of the k-means algorithm makes it impractical for use in the case of having a large number of points in the data set. Reducing the large number of nearest neighbor queries in the algorithm can accelerate it. In addition, the number of distance calculations increases exponentially with the increase of the dimensionality of the data [5-7].

Many algorithms have been proposed to accelerate the k-means. In [5,6], the use of kd-trees[8] is suggested to accelerate the k-means. However, backtracking is required, a case in which the computation complexity is increased [7]. Kd-trees are not efficient for higher dimensions. Furthermore, it is not guaranteed that an exact match of the nearest neighbor

can be found unless some extra search is done as discussed in [9]. Elkan[10] suggests the use of triangle inequality to accelerate the k-means. In [11], it is suggested to use RTrees. Nevertheless, R-Trees may not be appropriate for higher dimensional problems. In [12-14], the Partial Distance (PD) algorithm has been proposed. The algorithm allows early termination of the distance calculation by introducing a premature exit condition in the search process. Recently, Kernel -means [15] is an extension of the standard k-means algorithm that maps data points from the input space to a feature space through a nonlinear transformation and minimizes the clustering error in feature space. Thus, nonlinearly separated clusters in input space are obtained, overcoming the second limitation of k-means.

As seen in the literature, the researchers contributed only to accelerate the algorithm; there is no contribution in cluster refinement. In this study, we propose a new algorithm to improve the k-means using Genetic Algorithm (GA) is applied to refine the cluster to improve the quality.

The paper is organized as follows: the following section presents the general k-means algorithm. Section 3 presents the kernel k-means clustering and Section 4 discusses the proposed cluster refinement algorithm with genetic algorithm. Section 5 presents the results and the work is concluded in section 6.

II. STANDARD K-MEANS CLUSTERING

One of the most popular clustering techniques is the k-means clustering algorithm. Starting from a random partitioning, the algorithm repeatedly (i) computes the current cluster centers (i.e. the average vector of each cluster in data space) and (ii) reassigns each data item to the cluster whose centre is closest to it. It terminates when no more reassignments take place. By this means, the intra-cluster variance, that is, the sum of squares of the differences between data items and their associated cluster centers is locally minimized. k -means' strength is its runtime, which is linear in the number of data elements, and its ease of implementation. However, the algorithm tends to get stuck in suboptimal solutions (dependent on the initial partitioning and the data ordering) and it works well only for spherically shaped clusters. It requires the number of clusters to be provided or to be determined (semi-) automatically. In our experiments, we run k-means using the correct cluster number.

1. Choose a number of clusters k
2. Initialize cluster centers $\mu_1,\ldots \mu_k$
    a. Could pick k data points and set cluster centers to these points
    b. Or could randomly assign points to clusters and take means of clusters
3. For each data point, compute the cluster center it is closest to (using some distance measure) and assign the data point to this cluster.
4. Re-compute cluster centers (mean of data points in cluster)

Ms. K.Arun Prabha [a,*]
Assistant Professor, Department of Computer Science,
Vellalar College for Women (Autonomous), Erode, India.
(Email: arunjeevesh@gmail.com)
R.Saranya [b,1]
Research Scholar, Department of Computer Science,
Vellalar College for Women (Autonomous), Erode, India.
(Email: saranya.har@gmail.com)

5. Stop when there are no new re-assignments.

III. KERNEL K-MEANS CLUSTERING

Kernel k-means [15] is a generalization of the standard k-means algorithm where data points are mapped from input space to a higher dimensional feature space through a nonlinear transformation ϕ and then k-means is applied in feature space. This results in linear separators in feature space which correspond to nonlinear separators in input space. Thus, kernel k-means avoids the limitation of linearly separable clusters in input space that k-means suffers from. The objective function that kernel k-means tries to minimize is the clustering error in feature space. We can define a kernel matrix $K \in R^{N\times N}$, where $K_{ij} = \phi (X_i)^T \phi(X_j)$. Any positive-semidefinite matrix can be used as a kernel matrix. Notice that in this case cluster centers m_k in feature space cannot be calculated. Usually, a kernel function $K(x_i, x_j)$ is used to directly provide the inner products in feature space without explicitly defining transformation ϕ (for certain kernel functions the corresponding transformation is intractable), hence $K_{ij} = K(x_i, x_j)$. Some kernel function examples are given in Table 1. Kernel k-means is described in the following algorithm.

Input: Kernel Matrix K, number of clusters k, initial cluster centers $C_1,\ldots,C_k$

Output: Final Clusters $C_1,\ldots,C_k$ with clustering error E

a. For all points x_n, n = 1,…,N do
   i. For all clusters C_i where i = 1 to k do
      Compute $\| \phi(x_n) - m_i \|^2$
   ii. End
   iii. Find $c^*(x_n) = \arg\min_i (\| \phi(x_n) - m_i \|^2)$
b. End for
c. For all clusters C_i where i = 1 to k do
   Update cluster $C_i = \{x_n\ c^*(x_n) = i$
d. End
e. If converged then
   Return final clusters $C_1,\ldots,C_k$ and the Error
f. Else
   Goto Step (a)
g. End if

Table 1. Examples of Kernel Functions

Polynomial Kernel	$K(x_i, x_j) = [(x_i)^T x_j + \gamma]^\delta$
Gaussian Kernel	$K(x_i, x_j) = \exp (-\| x_i - x_j \|^2 / 2\sigma^2)$
Sigmoid Kernel	$K(x_i, x_j) = \tanh (\gamma(x_i^T x_j) + \theta)$

It can be shown that kernel k-means monotonically converges if the kernel matrix is positive semidefinite, i.e., is a valid kernel matrix. If the kernel matrix is not positive semidefinite, the algorithm may still converge, but this is not guaranteed.

IV. GENETIC ALGORITHM BASED REFINEMENT

Genetic algorithm (GA) [16] is randomized search and optimization techniques guided by the principles of evolution and natural genetics, having a large amount of implicit parallelism. GA perform search in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem.

In GA, the parameters of the search space are encoded in the form of strings (called chromosomes). A collection of such strings is called a population. Initially, a random population is created, which represents different points in the search space. An objective and fitness function is associated with each string that represents the degree of goodness of the string. Based on the principle of survival of the fittest, a few of the strings are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators like cross-over and mutation are applied on these strings to yield a new generation of strings. The process of selection, crossover and mutation continues for a fixed number of generations or till a termination condition is satisfied. An excellent survey of GA along with the programming structure used can be found in [17]. GA have applications in fields as diverse as VLSI design, image processing, neural networks, machine learning, job shop scheduling, etc.

The basic reason for our refinement is, in any clustering algorithm the obtained clusters will never give 100% quality. There will be some errors known as mis-clustered. That is, a data item can be wrongly clustered. These kinds of errors can be avoided by using our refinement algorithm.

The cluster obtained from the kernel k-means clustering is considered as input to our refinement algorithm. Initially a random point is selected from each cluster; with this a chromosome is build. Like this an initial population with 10 chromosomes is build. For each chromosome the entropy is calculated as fitness value and the global minimum is extracted. With this initial population, the genetic operators such as reproduction, crossover and mutation are applied to produce a new population. While applying crossover operator, the cluster points will get shuffled means that a point can move from one cluster to another. From this new population, the local minimum fitness value is calculated and compared with global minimum. If the local minimum is less than the global minimum then the global minimum is assigned with the local minimum, and the next iteration is continued with the new population. Otherwise, the next iteration is continued with the same old population. This process is repeated for N number of iterations.

A. String Representation

Here the chromosomes are encoded with real numbers; the number of genes in each chromosome is equal to the number of clusters. Each gene will have 5 digits for vector index. For example, our data set contains 5 clusters, so a sample chromosome may looks like as follows:

00100 10010 00256 01875 00098

Here, the 00098 represents, the 98th instance is available at first cluster and the second gene says that the 1875 instance is at second cluster. Once the initial population is generated now we are ready to apply genetic operators.

B. Reproduction (selection)

The selection process selects chromosomes from the mating pool directed by the survival of the fittest concept of natural genetic systems. In the proportional selection strategy adopted in this article, a chromosome is assigned a number of copies, which is proportional to its fitness in the population,

that go into the mating pool for further genetic operations. Roulette wheel selection is one common technique that implements the proportional selection strategy.

C. Crossover

Crossover is a probabilistic process that exchanges information between two parent chromosomes for generating two child chromosomes. In this paper, single point crossover with a fixed crossover probability of p_c is used. For chromosomes of length l, a random integer, called the crossover point, is generated in the range [1, l-1]. The portions of the chromosomes lying to the right of the crossover point are exchanged to produce two offspring.

D. Mutation

Each chromosome undergoes mutation with a fixed probability p_m. For binary representation of chromosomes, a bit position (or gene) is mutated by simply flipping its value. Since we are considering real numbers in this paper, a random position is chosen in the chromosome and replace by a random number between 0-9.

After the genetic operators are applied, the local minimum fitness value is calculated and compared with global minimum. If the local minimum is less than the global minimum then the global minimum is assigned with the local minimum, and the next iteration is continued with the new population. The cluster points will be repositioned corresponding to the chromosome having global minimum. Otherwise, the next iteration is continued with the same old population. This process is repeated for N number of iterations. From the following section, it is shown that our refinement algorithm improves the cluster quality. The algorithm is given as:

1. Choose a number of clusters k
2. Initialize cluster centers $\mu_1,\ldots \mu_k$ based on mode
3. For each data point, compute the cluster center it is closest to (using some distance measure) and assign the data point to this cluster.
4. Re-compute cluster centers (mean of data points in cluster)
5. Stop when there are no new re-assignments.
6. GA based refinement
    a. Construct the initial population (p1)
    b. Calculate the global minimum (Gmin)
    c. For i = 1 to N do
        i. Perform reproduction
        ii. Apply the crossover operator between each parent.
        iii. Perform mutation and get the new population. (p2)
        iv. Calculate the local minimum (Lmin).
        v. If Gmin < Lmin then
            a. Gmin = Lmin;
            b. p1 = p2;
    d. Repeat

V. EXPERIMENTS & RESULTS

For clustering, two measures of cluster "goodness" or quality are used. One type of measure allows us to compare different sets of clusters without reference to external knowledge and is called an internal quality measure. The other type of measures lets us evaluate how well the clustering is working by comparing the groups produced by the clustering techniques to known classes. This type of measure is called an external quality measure. One external measure is entropy [18], which provides a measure of "goodness" for un-nested clusters or for the clusters at one level of a hierarchical clustering. Another external measure is the F-measure, which, as we use it here, is more oriented toward measuring the effectiveness of a hierarchical clustering. The F measure has a long history, but was recently extended to data item hierarchies in [19].

Entropy

We use entropy as a measure of quality of the clusters (with the caveat that the best entropy is obtained when each cluster contains exactly one data point). Let CS be a clustering solution. For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the "probability" that a member of cluster j belongs to class i. Then using this class distribution, the entropy of each cluster j is calculated using the standard formula

$$E_j = -\sum_i p_{ij} \log(p_{ij})$$

where the sum is taken over all classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster:

$$E_{CS} = \sum_{j=1}^{m} \frac{n_j * E_j}{n}$$

where n_j is the size of cluster j, m is the number of clusters, and n is the total number of data points.

F measure

The second external quality measure is the F measure [19], a measure that combines the precision and recall ideas from information retrieval [20]. We treat each cluster as if it were the result of a query and each class as if it were the desired set of data items for a query. We then calculate the recall and precision of that cluster for each given class. More specifically, for cluster j and class i

$$\text{Recall}(i, j) = n_{ij} / n_i$$
$$\text{Precision}(i, j) = n_{ij} / n_j$$

where n_{ij} is the number of members of class i in cluster j, n_j is the number of members of cluster j and ni is the number of members of class i.

The F measure of cluster j and class i is then given by

$$F(i, j) = (2 * \text{Recall}(i, j) * \text{Precision}(i, j)) / ((\text{Precision}(i, j) + \text{Recall}(i, j))$$

For an entire hierarchical clustering the F measure of any class is the maximum value it attains at any node in the tree and an overall value for the F measure is computed by taking

the weighted average of all values for the F measure as given by the following.

$$F = \sum_i \frac{n_i}{n} \max\{F(i,j)\}$$

where the max is taken over all clusters at all levels, and n is the number of data items.

The following table presents the results, shows that our proposed method outperforms than the standard method.

Table 2. Performance Analysis of Cluster Quality

	Wisconsin Breast Cancer Dataset			Dermatology Dataset		
	K-Means	Kernel K-Means	Refined K-Means with GA	K-Means	Kernel K-Means	Refined K-Means with GA
No. of Classes	2	2	2	6	6	6
No. of Clusters	2	2	2	6	6	6
Entropy	0.3637	0.2373	0.1502	0.1826	0.0868	0.0103
F-measure	0.9125	0.9599	0.9799	0.8303	0.8537	0.8841

VI. CONCLUSION

In this paper, we have proposed a new framework to improve the cluster quality from k-means clustering using genetic algorithm. The proposed algorithm is tested in medical domain and show that refined initial starting points and post processing refinement of clusters indeed lead to improved solutions. The method is scalable and can be coupled with a scalable clustering algorithm to address the large-scale clustering problems in data mining. Experimental results show that the proposed algorithm achieves better results than the conventional and kernel k-means algorithm when applied to real data sets.

REFERENCES

[1] Lv T., Huang S., Zhang X., and Wang Z, Combining Multiple Clustering Methods Based on Core Group. Proceedings of the Second International Conference on Semantics, Knowledge and Grid (SKG'06), pp: 29-29, 2006.

[2] Nock R., and Nielsen F., On Weighting Clustering. IEEE Transactions and Pattern Analysis and Machine Intelligence, 28(8): 1223-1235, 2006.

[3] Xu R., and Wunsch D., Survey of clustering algorithms. IEEE Trans. Neural Networks, 16 (3): 645-678, 2005.

[4] MacQueen J., Some methods for classification and analysis of multivariate observations. Proc. 5th Berkeley Symp. Math. Stat. and Prob, pp: 281-97, 1967.

[5] Kanungo T., Mount D.M., Netanyahu N., Piatko C., Silverman R., and Wu A.Y., An efficient k-means clustering algorithm: Analysis and implementation. IEEE Trans. Pattern Analysis and Machine Intelligence, 24 (7): 881-892, 2002.

[6] Pelleg D., and Moore A., Accelerating exact k-means algorithm with geometric reasoning. Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, pp. 727-734, 1999.

[7] Sproull R., Refinements to Nearest-Neighbor Searching in K-Dimensional Trees. Algorithmica, 6: 579-589, 1991.

[8] Bentley J., Multidimensional Binary Search Trees Used for Associative Searching. Commun. ACM, 18 (9): 509-517, 1975.

[9] Friedman J., Bentley J., and Finkel R., An Algorithm for Finding Best Matches in Logarithmic Expected Time. ACM Trans. Math. Soft. 3 (2): 209-226, 1977.

[10] Elkan, C., Using the Triangle Inequality to Accelerate k-Means. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), pp. 609-616, 2003.

[11] Hjaltason, R. and Samet H., Distance Browsing in Spatial Databases. ACM Transactions on Database Systems, 24 (2): 26-42, 1999.

[12] Proietti, G. and Faloutsos C., Analysis of Range Queries and Self-spatial Join Queries on Real Region Datasets Stored using an R-tree. IEEE Transactions on Knowledge and Data Engineering, 5 (12): 751-762, 2000.

[13] Cheng D., Gersho B., Ramamurthi Y., and Shoham Y., 1984. Fast Search Algorithms for Vector Quantization and Pattern Recognition. Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing, 1, pp:1-9, 1984.

[14] Bei C., and Gray, R., An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization. IEEE Transactions on Communications, 33 (10): 1132-1133, 1985.

[15] Scholkopf B., Smola J., and Muller R., Nonlinear component analysis as a kernel eigenvalue problem," Neural Comput., 10(5):1299–1319, 1998.

[16] Davis (Ed.) L., Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.

[17] Michalewicz Z., "Genetic Algorithms, Data Structures" Evolution Programs, Springer, New York, 1992.

[18] Shannon CE., A mathematical theory of communication, Bell System Technical Journal, 27:379-423 and 623-656, July and October, 1948.

[19] Kowalski G, Information Retrieval Systems – Theory and Implementation, Kluwer Academic Publishers, 1997.

[20] Larsen B., and Aone C. Fast and Effective Text Mining Using Linear-time Document Clustering, KDD-99, San Diego, California, 1999.

BIOGRAPHY

Ms. K. Arun Prabha M.C.A., M.Phil., is currently working as an Assistant Professor in the Department of Computer Science, Vellalar College for Women (Autonomous), Erode. She has got 14 years of teaching experience and 3 years of research experience. She has published 4 papers in the national/International journals/conferences and also presented 5 papers in national/International journals /conferences. Her areas of interest include Data Mining and Soft Computing.

R. Saranya received her Bachelors Degree (B.C.A) in Computer Application and Master Degree (M.Sc) in Computer Science from Vysya College, Periyar University, Salem. She is currently pursuing her M.Phil research in Vellalar College for Women (Autonomous), Erode. Her areas of interests include Data mining and Soft Computing.