Initializing K-Means using Genetic Algorithms

Bashar Al-Shboul, Sung-Hyon Myaeng

***Abstract*—** K-Means (KM) is considered one of the major algorithms widely used in clustering. However, it still has some problems, and one of them is in its initialization step where it is normally done randomly. Another problem for KM is that it converges to local minima. Genetic algorithms are one of the evolutionary algorithms inspired from nature and utilized in the field of clustering. In this paper, we propose two algorithms to solve the initialization problem, Genetic Algorithm Initializes KM (GAIK) and KM Initializes Genetic Algorithm (KIGA). To show the effectiveness and efficiency of our algorithms, a comparative study was done among GAIK, KIGA, Genetic-based Clustering Algorithm (GCA), and FCM [19].

***Keywords*—**Clustering, Genetic Algorithms, K-means

I. Introduction

Clustering is the process of grouping data into clusters, where objects within each cluster have high similarity, but are dissimilar to the objects in other clusters [11]. Similarities are assessed based on the attribute value(s) that best describes the object. Often distance measures are used for the purpose. Clustering has its roots in many areas, including data mining, statistics, biology, and machine learning.

Among the various clustering algorithms, K-Means (KM) is one of the most popular methods used in data analysis due to its good computational performance [20]. However, it is well known that KM might converge to a local optimum, and its result depends on the initialization process, which randomly generates the initial clustering. In other words, different runs of KM on the same input data might produce different results.

Genetic Algorithms attempt to incorporate the ideas of natural evolution. In general they start with an initial population, and then a new population is created based on the notion of survival of the fittest. Typically fitness is the measure for how good this population is and can be calculated depending on the nature of the application, where a distance measure is the most common [19]. Then a process called crossover is done over the new population where substrings from selected pairs are swapped. The selection depends on the fitness of both pairs where the fittest pairs have the highest priority to crossover together. After that mutation may occur, where randomly selected points of each cluster are assigned to another cluster. This process continues until a generation where its fitness evolves to a pre-specified threshold or after a specific number of generations. A number of researchers have proposed genetic algorithms for clustering [16, 17, 19]. The basic idea is to simulate the evolution process of nature and evolve solutions from one generation to the next. In contrast to KM, which might converge to a local optimum, these genetic algorithms are insensitive to the initialization process and always converge to the global optimum eventually. However, these algorithms are usually computationally expensive.

The main contribution of this paper is to show the feasibility of applying genetic algorithms as an initialization method for the KM clustering technique, and build a new model that enhances the quality of the clustering (reduces the upcoming error).

This paper is organized as follows. Clustering is surveyed in section 2. In section 3 genetic algorithms are explained. A description of our work is listed in section 4. Section 5 contains data description and result analysis. Finally, we conclude in Section 6.

II. Clustering

One of the basic problems that arise in a variety of fields, including pattern recognition, machine learning and statistics, is clustering. The fundamental data clustering problem may be defined as discovering groups in data or grouping similar objects together. Each of these groups is called a cluster, a region in which density of objects is locally higher than in other regions [20, 13]. Each cluster is a collection of objects which are similar to each other and are dissimilar to the objects belonging to other clusters. The similarity mostly is measured with distance: two or more objects belong to the same cluster if they are close according to a given distance

Bashar Al-Shboul is with the IR & NLP Lab, Korean Advanced Institute of Science and Technology - IT Convergence Campus, 119 Munjiro, Yuseong-gu, Daejeon, 305-732, Republic of Korea, phone: +82-10-7219-1317; e-mail: bashar@kaist.ac.kr, basher@icu.ac.kr).

Sung-Hyon Myaeng, Ph.D. is the professor of IR & NLP Lab and he is with the Computer Science Department, Korean Advanced Institute of Science and Technology - IT Convergence Campus (e-mail: myaeng@kaist.ac.kr).

[6]. Sometimes similarity is measured referring to a concept representing a cluster. Two or more objects belong to the same cluster if it defines a concept common to all these objects. In other words, objects are grouped according to their fit to a descriptive concept.

The goal of clustering is to find groups of similar objects based on a similarity metric. However, a similarity metric is mainly defined by the user to ensure it suits his needs. Until now, there is still no absolute measure that always fit all applications.

Some of the problems associated with current clustering algorithms are that they do not address all the requirements adequately, and need high time complexity when dealing with a large number of dimensions and large data sets. Effectiveness of a method depends on the definition of distance, meaning that if a distance measure is not defined we have to define it; even though it might be somehow impossible in high dimensional space. However, the result of the clustering algorithm can be interpreted in different ways [11].

A. K-Means Clustering

K-Means [18] is one of the algorithms that solve the well known clustering problem. The algorithm classifies objects to a pre-defined number of clusters, which is given by the user (assume *k* clusters). The idea is to choose random cluster centers, one for each cluster. These centers are preferred to be as far as possible from each other. Starting points affect the clustering process and results. After that, each point will be taken into consideration to calculate similarity with all cluster centers through a distance measure, and it will be assigned to the most similar cluster, the nearest cluster center. When this assignment process is over, a new center will be calculated for each cluster using the points in it. For each cluster, the mean value will be calculated for the coordinates of all the points in that cluster and set as the coordinates of the new center. Once we have these *k* new centroids or center points, the assignment process must start over. As a result of this loop we may notice that the *k* centroids change their locations step by step until no more changes are made. When the centroids do not move any more or no more errors exist in the clusters, we call the clustering has reached a minima. Finally, this algorithm aims at minimizing an objective function, which is in this case a squared error function. The algorithm is expressed in Figure 1.

One drawback of KM is that it is sensitive to the initially selected points, and so it does not always produce the same output. Furthermore, this algorithm does not guarantee to find the global optimum, although it will always terminate. To reduce the effect of randomness, the user can run the algorithm many times before taking an average values for all runs, or at least take the median value.

(1) Choose random k points and set as cluster centers.
(2) Assign each object to the closest centroid's cluster.
(3) When all objects have been assigned, recalculate the positions of the centroids.
(4) go back to Steps 2 unless the centroids are not changing.

Fig. 1 Pseudo-code for K-Means algorithm

One popular way to start KM is to randomly choose k points from data. Initial starting points are important in the clustering process; however, the results mainly depend on the initial means. The standard solution is to try a number of different starting points. Moreover, the results also depend on the metric used to measure distance which is not always easy to implement especially in the high-dimensional space. Additionally, the results depend on the value of k, which in the real world are not always known or determined in advance [20].

Unfortunately, there is no general theoretical solution to find the optimal number of clusters for any given data set. A simple approach is to compare the results of multiple runs with different k clusters and choose the best one according to a given criterion. However, we need to be careful as increasing k results in smaller error-function values by definition, due to the few number of data points each center will represent, and thus it will lose its generalization ability, as well as increasing the risk of overfitting.

B. Initialization for Clustering Techniques

The main purpose of clustering algorithm modifications is to improve the performance of the underlying algorithms by fixing their weaknesses. And because randomness is one of the techniques used in initializing many of clustering techniques, and giving each point an equal opportunity to be an initial one, it is considered the main point of weakness that has to be solved. However, because of the sensitivity of K-Means to its initial points, which is considered very high, we have to make them as near to global minima as possible in order to improve the clustering performance. [3, 5]

III. GENETIC ALGORITHMS

Genetic algorithms were inspired from Darwin's theory of evolution and were pioneered by John Holland [12]. A genetic algorithm can be defined as a search algorithm based on the mechanics of natural selection and natural genetics [7], or as software and procedures modeled after genetics and evolution [1]. Genetic algorithms have at least the following elements in common: Populations of chromosomes,

selection according to fitness, crossover to produce offspring, and random mutation of a new offspring [2].

In further details, the algorithm starts with a population of "individuals", each representing a possible solution to a given problem. Each possible solution within the population of a biological individual is coded in a so-called chromosome. Each chromosome (sequences of genes) is assigned a "fitness" according to how good a solution is to the problem based on a given fitness function. The solutions (individuals) are selected into the process according to their fitness, specifically those that follow the principles first laid down by Charles Darwin of the survival of the fittest for reproduction by "cross breeding" with other individuals in the population and used to construct new individuals as offspring with a hope that the offspring will be fit better than the old individuals and a generation is complete [12]. This process is repeated until certain criteria are met. Figure 2 shows the basic steps for GAs [7].

```
t = 0;
Initialize P(t);
Evaluate P(t);
While not (termination condition)
begin
t=t+1;
Select P(t) from P(t - 1);
Recombine pairs in P(t);
Mutate P(t);
Evaluate P(t);
End
```

Fig. 2 Goldberg's Pseudo-code of Genetic Algorithms

In Figure 2, t represents the generation number, and P stands for population. The first population is initialized by coding it into a specific type of representation (i.e. binary, decimal, float, etc) then assigned to a cluster. Fitness is calculated in the evaluation step. While the termination condition is not met, which might be number of generations or a specific fitness threshold, the processes of selection, recombination, mutations and fitness calculations are done. Selection process chooses individuals from population for the process of crossover. Recombination (or crossover) is done by exchanging a part (or some parts) between the chosen individuals, which is dependent on the type of crossover (Single point, Two points, Uniform, etc). Mutation is done after that by replacing few points among randomly chosen individuals. Then fitness has to be recalculated to be the basis for the next cycle. This is the general form for GAs.

Utilizing GAs into clustering, an initial population of random clusters is set or accepted. At each generation, each individual is evaluated and recombined with others on the basis of its fitness. The expected number of times an individual is selected for recombination is proportional to its fitness relative to the rest of the population. New individuals are created using two main genetic recombination operators known as crossover and mutation. Crossover operates by selecting a random location in the genetic string of the parents (crossover point) and concatenating the initial part of one parent with the final part of the second parent to create a new child. A second child is simultaneously generated using the remaining parts of the two parents. Mutation is provided to occasional disturbances in the crossover operation by inverting one or more genetic elements during the reproduction process. This operation insures diversity in the genetic strings over long periods of time and prevents stagnation in the convergence of the optimization technique. In addition to fitness, generation crossover rate (or percentage) and mutation rate (or percentage) issues such as the size of the population (defines the crossover and mutation rates), coding and selection strategy (defines the fitness measure and type) are called configuration parameters [7].

Before a GA is run, a suitable encoding (or representation) for the problem must be devised. The coding is a population of strings, each of which represents a solution to the problem. GAs operate on a number of potential solutions, called a population, consisting of some encoding of the parameter, set simultaneously. Coding is the first step in GAs that translates the real problem into biological terms and describes it in a manner which is suitable for GAs. The format of a chromosome is called encoding.

The population size is specified by the number of chromosomes in the population, where the best population size depends on both the application and the length of the chromosome. Longer chromosomes allow for larger population sizes and increased variety for the initial population and would result in better exploration of the search space at the expense of requiring more fitness evaluations. If there are too many chromosomes, GAs slow down. If there are too few chromosomes, however, GAs has only few possibilities to perform the crossover operation and only a small part of search space is explored. If the population loses diversity, it is said to have a premature convergence and little exploration is being done [14].

Next step is Crossover. Crossover or recombination is done independently without respect to the problem of encoding or the fitness scores. It

takes two individuals and cuts their chromosome strings at some chosen position to produce two "head" segments and two "tail" segments. The tail segments are then swapped over to produce two new full length chromosomes. Each of the two offspring inherits some genes from each parent. Crossover is made with the hope that new chromosomes will contain good parts of old chromosomes. As a result, the new chromosomes are expected to be better. If crossover is performed, the genes between the parents are swapped, and the offspring is made from parts of both parents' chromosomes. If no crossover is performed, the offspring is an exact copy of its parents.

The most common recombination is the uniform crossover method. In this method, a crossover point is selected along the chromosome, and the genes up to that point are swapped between the two parents. Mutation is applied to each child individually after the crossover that alters each gene with a low probability, typically in the range 0.001 and 0.01, and modifies elements in the chromosomes [8]. Mutation is often seen as providing a guarantee that the probability of searching any given string will never be zero, it acts as a safety net to recover the good genetic material that may be lost through the action of selection and crossover. Mutation prevents the GA from falling into local extremes and provides a small amount of random search that helps ensure that no point in the search space has a zero probability of being examined. If mutation is performed, one or more parts of a chromosome are changed, and if there is no mutation, the offspring is generated immediately after the crossover (or directly copied) without any change [8].

For every solution to our population, it is necessary to be able to judge the quality of that solution. This is referred to as measuring the fitness of the solution. The fitness function is the most crucial aspect of GAs that returns a single numerical "fitness", which is supposed to be the proportional ability of the individual which that chromosome represents. Ideally, we want the fitness function to be smooth and regular, so that chromosomes with a reasonable fitness are close to chromosomes with slightly better fitness. [4]

The general rule in constructing a fitness function is that it should reflect the value of the chromosome in some real way. If the fitness function is excessively slow or complex to evaluate, an approximate function evaluation can sometimes be used. If a much faster function can be devised (which approximately gives the value of the true fitness function) the GA may find a better chromosome in a given amount of CPU time than when using the true fitness function.

At the beginning of a run, the values of each gene for different members of the population are randomly distributed. Consequently, there is a wide spread of individual fitness. As the run progresses, particular values for each gene begin to predominate. As the population converges, the range of fitness in the population reduces. This variation in fitness range throughout a run often leads to the problems of premature convergence and slow finishing. *Premature convergence* is a classical problem with GAs is that the genes from a few comparatively highly fit (but not optimal) individual may rapidly come to dominate the population causing it to converge on a local maximum, and thus the ability of the GA to continue to search for better solutions is effectively eliminated. Mutation may be a factor that helps to explore new offspring, but still have a small effect, which makes the process of exploration slower. Another problem is *Slow Finishing* that is when termination is a matter of fitness not the number of generations and in a certain point in time where fitness for all individuals are almost the same, average fitness moves slowly to the maxima, but still very slow [4]. A common practice is to terminate the GA after a pre-specified number of generations and then test the quality of the best members of the population against the problem definition. If no acceptable solutions are found, the GA may be restarted, or a fresh search initiated [12].

IV. Proposed Techniques

The proposed techniques considered in this paper are Genetic Algorithm Initializes K-means (GAIK) and K-means Initializes Genetic Algorithm (KIGA). GAIK is a combination of K-means and GKA, where GKA is executed first to give initial values to K-means to start with, rather than choosing random ones. This hybrid system is expected to minimize the number of iterations that K-means needs in order to converge to local minima. Besides, it solves the problem of blind search for this algorithm. However, it will increase the time needed for calculations because GA algorithm will need more time for distance calculations and crossovers in each generation than K-means needs in one iteration. On the other hand, another technique was experimented, KIGA, where K-means is used first to initialize the GA clustering technique. Two experiments were done in this part to show the effect of changing the number of K-means iterations in addition to the number of generations on the clustering efficiency. Good results are expected to be gained to solve the problem of blind search. However, the time of processing is also expected to increase. Testing parameters for selections, crossovers and mutations were considered the same as in [19].

V. DATE DESCRIPTION AND RESULT ANALYSIS

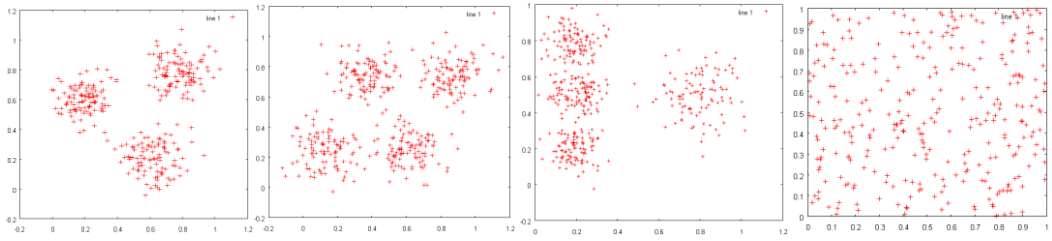
Data files used in these experiments are chosen among a huge variety given by MATLAB©. Experiments were done over eight different datasets, however; we will present here the results of four of them as each has different characteristics over the others. We chose to do the tests over a mathematically generated 2D datasets. Figure 3 shows the datasets' distribution. Dataset 2, 3 and 4, shown in Figure 3-a, 3-b and 3-c respectively are made based on a mathematical model to form their clusters with small amount of points interleaving. On the contrary, Dataset 5 shown in Figure 3-d is a random dataset with no clear topology and a uniform distribution.

Dataset 2 consists of 100 points gathered around 3 clusters. The points are scattered with a radius of 0.2 around 3 specific points (0.2 ,0.6) , (0.6,0.2) and (0.8,0.8). The first two points' clusters have some interleaving in their boundary points. This dataset is to be clustered into 3 clusters.

Dataset 3 Consists of 100 points scattered around 4 specific points with a radius of 0.2. points are (0.125,0.25) , (0.625,0.25) , (0.375,0.75) , (0.875,0.75). The first two points' clusters have horizontal interleaving on the boundary. In addition, points two and three have the same thing in common. This dataset is to be clustered into 4 clusters.

Dataset 4 Consists of 100 points scattered around 4 points with a radius of 0.3 for the first 3 centers and radius 0.4 for the last center. Points are (0.2,0.2) , (0.2,0.5) , (0.2,0.8) , (0.8,0.5). The last point's cluster seems to be clearly isolated except for the points between the cluster center and the other three; however, the first three clusters have common boundary points. This dataset is to be clustered into 4 clusters.

Dataset 5 consists of 300 points chosen randomly with uniform distribution over the surface, no clear topology or clusters. This dataset is to be tested for two clusters. For each of the previously mentioned datasets the number of clusters is chosen based on our choice; moreover, choosing the number of clusters is still a wide research area that we are not going to discuss in our work. In addition, for each dataset we tested running K-means for 5 or 10 iterations only in addition to running it until it satisfies its original termination condition. Also, we fixed number of GA generations for testing to 1000 generations at most, and then we considered the fittest generation as the error rate. Each dataset was tested for each algorithm for 20 times and then we calculated the average time and error as listed in Table 1.



(a) (b) (c) (d)

Fig. 3 Datasets distribution (a) dataset 2 (b) dataset 3 (c) dataset 4 (d) dataset 5

Table 1 Results comparison between the different approaches

		KM	KM (5)	KM (10)	GAIK	KM (5) + GA	KM (10) + GA	GA
Dataset2 (4)	Avg Error	36.389	38.491	36.660	40.257	35.848	**34.891**	48.485
	Avg Time	0.015	0.011	0.013	10.216	18.977	18.647	9.966
Dataset3 (4)	Avg Error	44.128	44.635	44.554	62.328	43.646	**43.047**	76.352
	Avg Time	0.012	0.010	0.014	9.180	14.842	16.203	6.608
Dataset4 (3)	Avg Error	35.094	35.588	35.554	62.848	**33.606**	33.769	70.711
	Avg Time	0.013	0.007	0.009	8.158	12.057	11.326	8.2729
Dataset5 (2)	Avg Error	20.069	20.541	20.329	20.315	18.877	**18.491**	21.207
	Avg Time	0.004	0.004	0.008	3.728	4.186	4.326	3.865

As explained earlier and shown in Table 1, for each dataset we considered specific number of clusters shown beside the dataset name between brackets. Moreover, Average Error and Average Time is listed to show the trade-off between them. For K-Means, we tried to test the effect of number of iterations in the clustering process and on the initialization process as well. The table above shows the following: KM was always the fastest, but not the most accurate. In addition, initializing KM using GA will definitely lead it to fall early into a local minima. GA was not a good approach to solve the clustering problem due to its probabilistic nature. Even though mutations help not to fall into local minimas, it still needs a lot of time and computations to find the global one. Finally, running KM as an initializer to GA definitely guides to the best solution among the group, this appears clearly from the results obtained after running it for 5 or 10 iterations before GA starts.

VI. Conclusion

Our experimental evaluation scheme was used to provide a common base of performance assessment and comparison with other methods. From the experiments on the eight data sets, we find that pre-initialized algorithms work well and yield meaningful and useful results in terms of finding good clustering configurations which contain interdependence information within clusters and discriminative information for clustering. In addition, it is more meaningful in selecting, from each cluster, significant centers, with high multiple interdependence with other points within each cluster.

Finally, when comparing the experimental results of K-Means, GKA, GAIK and KIGA we find that KIGA is better than the others. As shown by the results on all datasets KIGA is ready to achieve high clustering accuracy if compared to other algorithms.

References

[1] R.J. Bauer, "Genetic Algorithms and Investment Strategies", Wiley Finance Editions, New York, John Wiley and Sons, 1994.

[2] H. Ben Amor, and A. Rettinger, "Intelligent exploration for genetic algorithms: Using self-organizing maps in evolutionary computation", In proceedings of the 2005 conference on Genetic and evolutionary computation, 2005, pp. 1531–1538.

[3] P. Bradley, and U. Fayyad, "Refining Initial Points for K-Means Clustering," In Proceeding of 15th International Conference on Machine Learning, 1998, pp. 91-99.

[4] P. Chi, "Genetic Search with Proportion Estimation", In proceedings of the Third Int. Con. on Genetic Algorithms (ICGA), San Mateo, California, 1989, pp. 92-97.

[5] U. Fayyad, C. Reina, and J. Bradley, "Initialization of iterative refinement clustering algorithms", In proceedings of Fourth Int. Con. on Knowledge Discovery and Data Mining, AAAI, pp.194-198, 1998.

[6] C. Fraley, and A. Raftery, "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis," Technical Report 329, Dept. of Statistics, University of Washington, Seattle, 1998.

[7] D. Goldberg, "Computer-Aided Gas Pipeline Operation Using Genetic Algorithms And Rule Learning," Ph.D. thesis, University of Michigan, Ann Arbor, 1983.

[8] D. Goldberg, "Genetic Algorithms In Search, Optimization, And Machine Learning," Addison-Wesley, New York, 1989.

[9] J. Grefenstette, C. Ramsey, and A. Schultz, Learning Sequential Decision Rules Using Simulation Models And Competition, Machine Learning 5, Vol. 4, 1990, pp. 355-381.

[10] L. Hall, B. Ozyurt, and J. Bezdek, "Clustering With A Genetically Optimized Approach," IEEE Transactions on Evolutionary computation, Vol. 3, No. 2, 1999, pp 103-112.

[11] J. Han, and M. Kamber, "Data Mining: Concepts And Techniques," Morgan Kaufmann Publishers, 2001.

[12] J. Holland, "Adaptation In Natural And Artificial Systems," University of Michigan Press, 1975.

[13] L. Kaufman, and P. Rousseeuw, "Finding Groups In Data: And Introduction To Cluster Analysis," John Wiley and Sons, 1990.

[14] A. Korol, I. Preygel, and S. Preygel, "Algorithms Of Estimation And Population-Genetic Models," Central Library of Imperial College, Chapman and Hall, 1994.

[15] K. Krishna, and M. Murty, "Genetic K-Means Algorithm," IEEE Transactions on Systems, Vol. 29, NO. 3, 1999, pp. 433-439.

[16] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. Brown, "FGKA: A Fast Genetic K-Means Clustering Algorithm," ACM Symposium on Applied Computing, 2004

[17] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. Brown, "Incremental Genetic K-Means Algorithm And Its Application In Gene Expression Data Analysis," BMC Bioinformatics, 2004.

[18] J. MacQueen, "Some Methods For Classification And Analysis Of Multivariate Observations," In proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, 1967, pp. 281-297.

[19] U. Maulik, and S. Bandyopadhyay, "Genetic Algorithm-Based Clustering Technique", Pattern Recognition 33, 1999, pp. 1455-1465.

[20] J. Pena, J. Lozano, and P. Larranaga, "An Empirical Comparison Of Four Initialization Methods For The K-Means Algorithm," Pattern Recognition Letters, Vol. 20 No. 10, 1999, pp. 1027-10.