ijecs open access www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 6 June, 2014 Page No. 6442-6447

A Genetic Algorithm Approach for Clustering

Mamta Mor[1], *Poonam Gupta*[2], *Priyanka Sharma*[3]

[1] OITM, Dept. Of CSE, GJUS&T,
India
mamtamor12121990@gmail.com

[2] OITM, Dept. Of CSE, GJUS&T,
India
poonamjindal3@gmail.com

[3]GJUS&T, Dept. Of CSE,
India
pinki.sharma2912@gmail.com

Abstract: The paper deals with the applicability of GA to clustering and compares it with the standard K-means clustering technique. K-means clustering results are extremely sensitive to the initial centroids, so many a times it results in sub-optimal solutions. On the other hand the GA approach results in optimal solutions and finds globally optimal disjoint partitions. Fitness calculated on the basis of intra-cluster and inter-cluster distance is the performance evaluation standard in this paper. The experimental results show that the proposed GA is more effective than K-means and converges to more accurate clusters.

Keywords: clustering, genetic algorithm, k-means, fitness function

1. Introduction

Data mining is the process of extracting useful and hidden information or knowledge from data sets. The information so extracted can be used to improve the decision making capabilities of a company or an organization [1][2][3]. Data mining consists of six basic types of tasks which are Anomaly detection, Association rule learning, Clustering, Classification, Regression and Summarization. Clustering is one of the important tasks of data mining. Clustering is defined as the task of grouping objects in such a way that the objects in the same group/cluster share some similar properties/traits. There is a wide range of algorithms available for clustering like hierarchical, K-means clustering [4][5][6]. K-means is one of the most popular and frequently used clustering algorithm. It clusters objects into K number of groups, where K is a positive integer. But K-means has a major drawback that many a times it converges to a sub-optimal solution due to large clustering search space. Therefore, Evolutionary algorithms like genetic algorithm are suitable for clustering task. A good GA explores the search space properly as well as exploits the better solutions to find the globally optimal solution [7].

A GA is a stochastic search method[8][9] which works on a population of individuals (chromosomes) and produces new population with every generation by applying genetic operators. The proposed GA has been applied to UCI repository [19] of Machine Learning datasets i.e. ‘Seeds’, ‘Data_User_Modeling’, ‘Wholesale customers data’. The experimental results show that the proposed GA is consistently better and more effective than the k-means algorithm.

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3, 4 discusses the proposed GA design and an example respectively. Section 5 presents the data set descriptions and experimental results. Section 6 discusses the future scope and conclusion. Section 7 gives the references.

2. Related Work

Data mining is a field with a large area of application. Evolutionary algorithm particularly genetic algorithm and genetic programming have been used in the field of data mining & knowledge discovery [10]. Several GAs have been used for mining real world datasets in medical domain and in the field of education etc. [11][12]. A number of researchers have focused on using GA for data mining tasks of classification & clustering. Interest in the field of clustering has increased recently due to the emergence of several areas of application including bioinformatics, web use data analysis and image analysis etc. [13][14]. A few of the earlier models proposed for clustering are ‘Genetic K- means’ and ‘Fastest Genetic K- means’ models, which find a globally optimal partition of a given data into a specified number of clusters [15][16]. Many other GA models have also been proposed for clustering [17][18]. The GA model proposed earlier for clustering have particularly used intra-cluster distance as the parameter for calculating fitness function. This paper proposes a GA model which uses both intra-cluster as well as the inter-cluster distance to calculate the fitness.

3. Proposed GA Design

GA takes as input a population of individuals (binary or real valued) which evolves over generation by applying genetic operators (crossover and mutation).

3.1 Encoding Scheme:

Initialization: The initial population corresponds to X no. of centroids (where X=pop_size*k) randomly selected from the normalized data set, where k is the number of clusters to be formed. The data sets taken from the UCI repository are normalized before applying GA.

Chromosome length: Each chromosome in the population is a real valued vector of length k*nv where k is the number of clusters to be formed, nv is the number of attributes/variables in the data set, which means k rows are randomly selected from the dataset to represent an individual where each k_i (i=1,2,…m) represents one of the centroid of chromosome$_x$(x=1 to pop_size).

Initial population size: pop_size (no of rows), k*nv (no of attributes), which means pop_size*k number of centroids are actually selected for initial population.

3.2 Fitness Function:

The objective of fitness function is to maximize inter-cluster distance and minimize intra-cluster distance. The objects are clustered on the basis of Euclidean distance, each object belongs to the cluster whose centroid to object Euclidean distance is minimum. Let $\{X_i; i=1,2,\ldots n\}$ be a set of n objects, each with p attributes. The n objects are divided into k clusters with $\{C_m; m=1,2..k\}$ be the set centroids corresponding to k clusters.

Object-Centroid Distance (Euclidean distance): The distance between an object and a centroid can be calculated by Euclidean distances as follows:

$$\mathbf{E^D\ (X_i,\ C_j)} = \sqrt{\sum_{l=1}^{p}(X_{il} - C_{jl})^2} \qquad (1),$$

where i=1, 2,..n; j=1,2,….k

Intra-Cluster Distance: The intra-cluster distance is the distance between a cluster's elements. The intra-cluster distance of q th cluster where q=1,2,..k is calculated as follows:

$$\mathbf{D^q_{INTRA}(X_i,X_j)} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m}(X_i - X_j)^2/(m*m)} \qquad (2),$$

where m is no of elements in the q th cluster

The total intra-cluster distance is computed as below:

$$\mathbf{S\ (D_{INTRA})} = \sum_{q=1}^{k}(\mathbf{D^q_{INTRA}}) \qquad (3)$$

Inter-Cluster Distance: The inter-cluster distance is the distance between two cluster's elements. The inter-cluster distance between q th and r th cluster where q, r=1,2..k is calculated as follows:

$$\mathbf{D^{q,r}_{INTER}(X_i,X_j)} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(X_i - X_j)^2/(m*n)}$$

(4) where m, n is no of elements in q th and r th cluster respectively.

It is to be noted that for r=q inter-cluster distance is null and Inter-cluster distance between r,q & q,r is same.

The total inter-cluster distance is computed as below:

$$\mathbf{S(D_{INTER})} = \sum_{q=1}^{k-1}\sum_{r=q+1}^{k}(\mathbf{D^{q,r}_{INTER}}) \qquad (5)$$

Fitness: The fitness is computed by using the following formula:

$$F_{max=}\max(\mathbf{S\ (D_{INTER})/S\ (D_{INTRA})}) \qquad (6)$$

We have used the roulette wheel as the selection operator.

3.3 Crossover Operator

Genetic operators are applied to maintain genetic diversity. Genetic diversity/variation is necessary for the process of evolution. Crossover operator is one of the genetic operators. Crossover is applied to (p_c*pop_size) chromosomes where p_c is the probability of crossover [7]. The chromosomes are real valued vectors and the crossover applied is arithmetic crossover which works as follow:

Offspring1= (α * parent1) + ((1-α) * parent2)
Offspring2= ((1-α) * parent1) + (α* parent2)

Table1 (The Example Dataset)

10	20	10
12	18	8
11	21	11
9	20	9
10	17	11
40	50	60
42	48	58
41	51	59
38	47	60
40	52	57
80	100	120
81	101	119
78	98	118
80	100	121
82	102	120

Table 2 (Normalized Example Dataset)

0.0137	0.0353	0.0177
0.0411	0.0118	0
0.0274	0.0471	0.0265
0	0.0353	0.0088
0.0137	0	0.0265
0.4247	0.3882	0.4602
0.4521	0.3647	0.4425
0.4384	0.4000	0.4513
0.3973	0.3529	0.4602
0.4247	0.4118	0.4336
0.9726	0.9765	0.9912
0.9863	0.9882	0.9823
0.9452	0.9529	0.9735
0.9726	0.9765	1.0000
1.0000	1.0000	0.9912

3.4 Mutation Operator

The mutation applied is uniform mutation. Mutation is applied to (p_{m*}pop_size*u) number of elements/gene where p_m is the probability of mutation & u is the chromosome length. The uniform mutation replaces the value of chosen element/gene by a value randomly generated between the upper and lower bounds for that gene. Since the data is normalized, so the value of all genes lie between 0 & 1.

4: An Example

Let us consider a dataset with n=15 & nv= 3, where n is the number of rows and nv is the number of attributes. Table 1 shows the actual dataset whereas Table 2 shows the normalized dataset. Let pop_size be 4 & k=3.For pop_size=4, rows actually selected (X=pop_size*k) =4*3=12. Let Y= [4,12,14,7,9,1,2,3,5,13,11,15] be the indices returned of the selected 12 rows. The rows corresponding to the first 3 indices represent the 1st chromosome, where 1st index represents the 1st centroid, 2nd index represents the 2nd centroid, and 3rd index represents the 3rd centroid. Each chromosome has a length (u=k*nv) =3*3=9. It will become clear with the Table 3 given below:

Table 3

Chromosome No.	*Selected rows indices*	*Chromosome*
1	4,12,14	0 0.0353 0.0088 0.9863 0.9882 0.9823 0.9726 0.9765 1.0000
2	7,9,1	0.4521 0.3647 0.4425 0.3973 0.3529 0.4602 0.0137 0.0353 0.0177
3	2,3,5	0.0411 0.0118 0 0.0274 0.0471 0.0265 0.0137 0 0.0265
4	13,11,15	0.9452 0.9529 0.9735 0.9726 0.9765 0.9912 1.0000 1.0000 0.9912

The first three elements in each row corresponds to the 1st centroid, the next three elements in each row corresponds to the 2ndcentroid and the last three elements in each row corresponds to the 3rd centroid of every chromosome

The fitness of each chromosome will be calculated by the fitness formula proposed above:

Let us consider chromosome No. 1 where, 1st centroid (C_1) = 0 0.0353 0.0088 represents cluster1, 2nd centroid (C_2) = 0.9863 0.9882 0.9823 represents cluster2 and 3rd centroid (C_3) = 0.9726 0.9765 1.0000, represents cluster3.

Fitness Function returns a 1-by-15 vector IDX containing the cluster indices of each of the 15 points/rows by using squared Euclidean distances equation (1) given above:

1	1	1	1	1	1	1	1	1	1	3	2	3	3	2

Table 4 (IDX)

, which shows that the first 10 points of the example dataset belong to the 1st cluster1, 11th,13th,14th points belong to the 2nd cluster, 12th and 15th belong to the 3rd cluster.

The intra-cluster (Table 5) and inter-cluster distance (Table 6) of the clusters calculated by the equation No. 2 & 4 respectively given above is:

Table 5 (Intra-cluster distance of each cluster)

Cluster No.	Intra-cluster distance
1	4.9275
2	0.0284
3	0.0861

Table 6 (Inter-cluster distance b/w two clusters)

Cluster1-Cluster2	*Inter-cluster distance*
1-2	1.3805
1-3	1.3505
2-3	0.0452

Inter-cluster distance between clusters 1-1, 2-2, 3-3 is zero and between 1-2and 2-1, 2-3 and 3-2, 1-3 and 3-1 is same. So, it needs not to be calculated twice.
The total intra-cluster and inter-cluster distance is 5.0420 and 2.7763 respectively.

The fitness corresponding to chromosome No.1 = 2.7763 / 5.0420 = 0.5506

Similarly, the fitness corresponding to chromosome No. 2, 3 and 4 calculated are 0.4368, 0.1907 & 0.3434 respectively. We can see that chromosome No.1 has the best fitness among all chromosomes for 1st iteration.

The crossover operator is applied on two parents to produce two new off springs. Let us apply crossover on 3rd & 2nd chromosome of Table 3.

So, Parent 1= 0.0411 0.0118 0 0.0274 0.0471 0.0265 0.0137 0 0.0265

Parent 2= 0.4521 0.3647 0.4425 0.3973 0.3529 0.4602 0.0137 0.0353 0.0177

Let α= 0.6, then

Offspring 1= 0.2055 0.1530 0.1770 0.1754 0.1694 0.2000 0.0137 0.0131 0.0230

Offspring 2= 0.2877 0.2235 0.2655 0.2493 0.2306 0.2867 0.0137 0.0212 0.0212

The mutation operator is applied to the genes/elements. Let us apply mutation on the 5th element of 1st chromosome of table 3. The selected element is replaced by a random element between the lower and upper limit of that element which is 0 &1 respectively in this case.

Parent 3= 0 0.0353 0.0088 0.9863 0.9882 0.9823 0.9726 0.9765 1.0000

Offspring 3= 0 0.0353 0.0088 0.9863 **0.7982** 0.9823 0.9726 0.9765 1.0000

5. Experimental Data & Results

5.1 Datasets & platform description

The proposed GA design in the paper is implemented in MATLAB version 7.12.0 on a machine having 1 GB of RAM and INTEL core duo processor with 1.66 GHz speed.

The efficiency of the proposed GA design is evaluated by conducting experiments on three datasets downloaded from UCI repository [19]. The description of the data sets used for evaluating the proposed GA model is given below in Table 7:

All the three datasets are converted into csv files and the attribute values of 'string' type are converted into real values.

5.2 Results

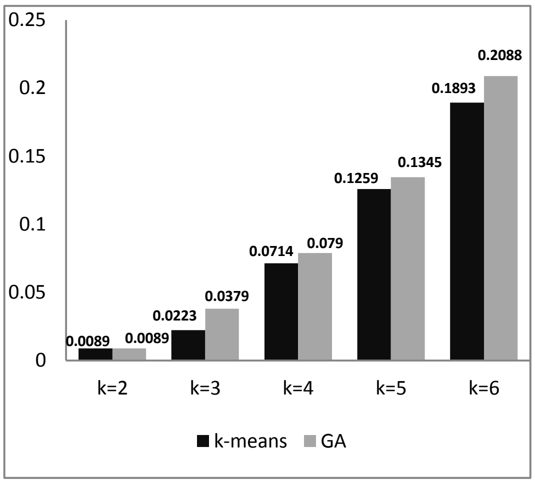
The results found during the simulation of the GA model are described as follows: Table1, 2 and 3 show the comparison of GA model with k-means algorithm of dataset 'seeds', 'Data_User_Modeling' and 'Whole sale customers' respectively. Figure 1, 2 and 3 show the comparison of GA model with k-means algorithm of dataset 'seeds', 'Data_User_Modeling' and 'Whole sale customers' respectively through bar charts. Figure 4, 5 shows the fitness versus generation graph and it can be seen that genetic algorithm has high fitness in all cases thus better and efficient to use.

Comparison between GA (fitness) & K-means (fitness) for dataset 'seeds'

Table 8

K	*GA(**fitness**)*	*Kmeans(**fitness**)*
2	.0089	.0089
3	.0379	.0223
4	.0790	.0713
5	.1345	.1259
6	.2088	.1893

Figure 1



Comparison between GA (fitness) & K-means (fitness) for dataset 'Whole Sale Customer'

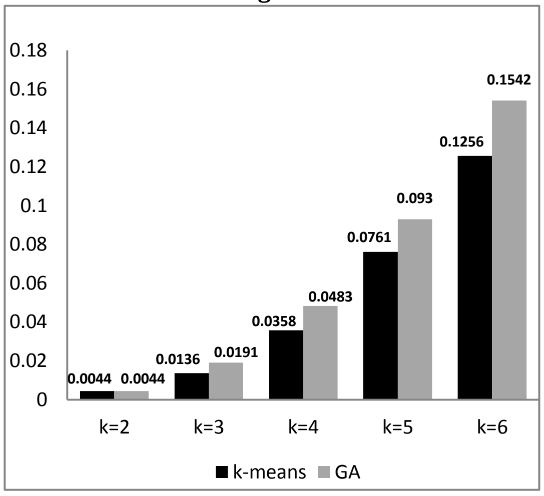
Table 9

K	*GA(**fitness**)*	*Kmeans(**fitness**)*
2	.0044	.0044
3	.0191	.0136
4	.0483	.0358
5	.0930	.0761
6	.1542	.1256

Table 7

Data Set	No. of instances	No. of attributes
Data_User_Modeling (Training data)	258	6
Seeds	210	8
Whole sale customers	440	8

Figure 2

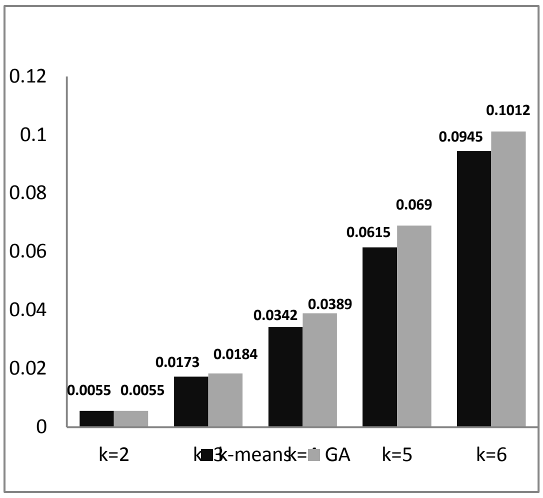


Comparison between GA (fitness) & K-means (fitness) for dataset 'Data_User_Modeling''

Table 10

K	*GA(**fitness**)*	*Kmeans(**fitness**)*
2	.0055	.0055
3	.0184	.0173
4	.0389	.0342
5	.0690	.0615
6	.1012	.0945

Figure 3



The above results makes it evident that GA gives consistently better results than k-means algorithm across all the three sets, except for the value of k=2.

<u>Fitness versus Generation graph of dataset 'Data_User_Modeling' for k= 3, 4</u>

Figure 4 (K=3)

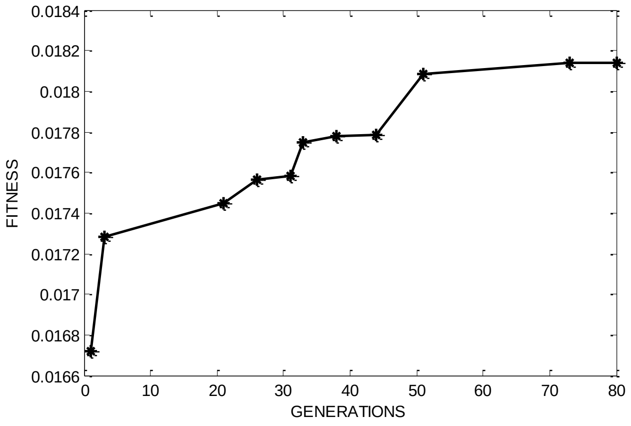
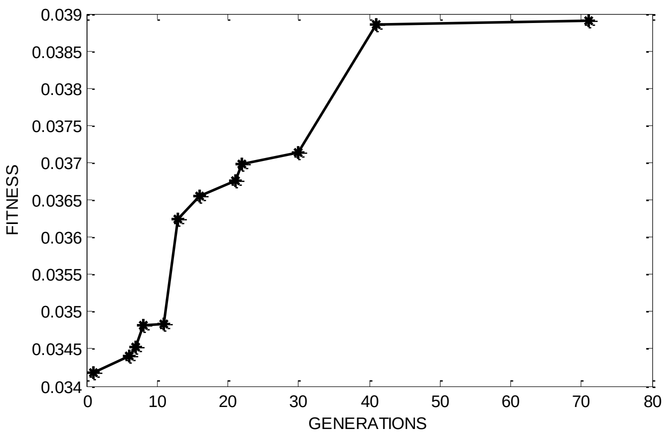


Figure 5 (K=4)



It is clear from the figure shown below that fitness increases with no. of generations and then it stabilizes

6. Conclusion and Future scope

Clustering has a wide range of application. A good clustering algorithm yields a good quality cluster with high intra-cluster similarity/low intra-cluster distance and low-inter cluster similarity/high inter-cluster distance. It also produces a global optimal or near to global optimal solution/result. The paper proposed a genetic algorithm which produces better clusters with low intra-cluster & high inter-cluster distance as compared to k-mean algorithm. The proposed GA code also overcomes the problem of local optimal solution faced in k-means by providing optimal solution for a given data set. Experimental results demonstrate that the proposed GA has clearly outperformed the standard K-means in terms of providing optimal solution.

The GA design presented in this paper overcomes one of the two major drawbacks of k-means clustering algorithm i.e. converging at sub optimal solution due to bad seed initialization. The other drawback of K-means is that K (number of clusters) has to be predetermined before applying k-means/clustering algorithm on a dataset. The future directions of the work presented in this paper would be to modify the GA in such a way that the best value of k will be calculated automatically by the GA model.

7. References

[1] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan kaufmann, 2006.
[2] A. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," in *Advances in evolutionary computing*, Springer, 2003, pp. 819–845.
[3] A. A. Freitas, *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer, 2002.
[4] K. Alsabti, S. Ranka, and V. Singh, "An efficient k-means clustering algorithm," 1997.
[5] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *Pattern Anal. Mach. Intell. IEEE Trans. On*, vol. 24, no. 7, pp. 881–892, 2002.
[6] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *ICML*, 2001, vol. 1, pp. 577–584.
[7] Z. Michalewicz, *Genetic algorithms+ data structures= evolution programs*. springer, 1996.
[8] M. C. Cowgill, R. J. Harvey, and L. T. Watson, "A genetic algorithm approach to cluster analysis," *Comput. Math. Appl.*, vol. 37, no. 7, pp. 99–108, 1999.
[9] J. J. Grefenstette, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*. Psychology Press, 2013.
[10] A. A. Freitas, "A review of evolutionary algorithms for data mining," in *Soft Computing for Knowledge Discovery and Data Mining*, Springer, 2008, pp. 79–111.
[11] P. Vishwakarma, Y. Kumar, and R. K. Nath, "Data Mining Using Genetic Algorithm (DMUGA)."
[12] B. Minaei-Bidgoli and W. F. Punch, "Using genetic algorithms for data mining optimization in an educational web-based system," in *Genetic and Evolutionary Computation—GECCO 2003*, 2003, pp. 2252–2263.

[13] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognit.*, vol. 33, no. 9, pp. 1355–1365, 2000.

[14] R. H. Sheikh, M. M. Raghuwanshi, and A. N. Jaiswal, "Genetic algorithm based clustering: a survey," in *Emerging Trends in Engineering and Technology, 2008. ICETET'08. First International Conference on*, 2008, pp. 313–319.

[15] K. Krishna and M. N. Murty, "Genetic K-means algorithm," *Syst. Man Cybern. Part B Cybern. IEEE Trans. On*, vol. 29, no. 3, pp. 433–439, 1999.

[16]Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown, "FGKA: A fast genetic k-means clustering algorithm," in *Proceedings of the 2004 ACM symposium on Applied computing*, 2004, pp. 622–623.

[17] R. M. Cole, *Clustering with genetic algorithms*. Citeseer, 1998.

[18] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognit.*, vol. 33, no. 9, pp. 1355–1365, 2000.

[19]Department of Information and Computer Science, University of California at Irvine, UCI Repository of Machine Learning databases.