Genetic Algorithm Based Split-Fusion Clustering

BarryJuans and Sheng-Uei Guan

***Abstract*—We introduce a new clustering algorithm which is based on the combination of GA and a new technique called split-and-fusion. GA is used to find the initial cluster while split and fusion refines the cluster by continuously breaking apart and merging patterns existing in the cluster. The whole process is repeated until all patterns have been clustered. The algorithm then merges the smallest-sized cluster with other clusters until termination condition is met. In the last step, a heuristic equation is used to evaluate the termination criteria. Experimental results show that the algorithm is accurate in clustering real-world datasets such as Iris and Wine datasets.**

***Index Terms*—Genetic algorithm, clustering, hybrid learning, high-dimensional space;**

I. Introduction

Clustering problem is an optimization problem in which the solution is found by organizing similar data sets into groups [1]. Similarities among the data sets can be based on geometrical distance or descriptive concepts (such as gender, hobbies, etc.), depending on the type of attributes in the data sets. In addition, clustering problem is categorized as unsupervised learning problem since it is concerned with finding commonalities in structures among given data sets. In clustering problem, optimal number of clusters is not known, unlike its similar counterpart - classification problem.

In order to solve the clustering problem, many new algorithms had been introduced, each with differing effectiveness. Those algorithms include the original clustering algorithms, evolutionary algorithms, and the hybrids of the two types [2].

One of the most well-known evolutionary algorithms is called Genetic Algorithm, abbreviated as GA. Like what the name implies, GA borrows the idea of natural selection in order to find the optimal solution to its problem [3]. The algorithm itself works by iterating through several biologically-inspired processes such as: Initialization, Selection, and Reproduction; it only ends when the Termination condition has been fulfilled. One important component of GA is the fitness function, which is used to evaluate the effectiveness of current solution. In each iteration, GA tries to find the solution which has better fitness compared to the previous iteration, which means that designing a good fitness function is crucial.

In most clustering problem, the fitness evaluation can be distance-based, with the closer data points being more likely to be clustered together. One of the most popular distance metric used is the Euclidean distance. This approach has shown to work in cases where the number of attributes is low; however, it becomes less effective as the number of attributes grows [4]. This can happen because the distance between any two data points converges with larger number of attributes. Furthermore, in high dimensions (attributes), all data points become less similar to each other, making it difficult to find the similarities among the data points. This phenomenon is known as the Curse of Dimensionality [5]. Additionally, distance-based measurement also forces clustering algorithms to find clusters that have circular shape [6]. Last but not least, difference in standard deviation among the attributes could skew the distance calculations.

Existing researches have shown that GA works quite effectively in combination with current Clustering algorithms such as k-means Clustering [3] and Hierarchical Clustering [7]; however, the problem with high-dimensions clustering is still yet to be solved.

This research aims to design a working Genetic algorithm based method which is able to accurately cluster the similar data points in high dimensional data space. This also implies that the research aims to find the fitness function which can evaluate the closeness of two high dimensional data points.

II. Problem Formulation

Let X = [x_1, x_2, x_3 … x_n] be the representation of the whole data sets, with x_i being an instance of X and *n* being the total number of instances. Each instance of X will be defined over d dimensions. The algorithm will take X as an input and try to group each instances of X into *k* clusters. As stated before, since the value of *k* is not known, different algorithms are likely to result in different number of clusters. The optimal value of *k* is found when there is a significant pattern difference among the different clusters while dissimilarities among data sets in each cluster are minimized. To illustrate:

TABLE I: Iris Data Set

Cluster	Attribute Number			
	1	2	3	4
1	5.1	3.5	**1.4**	**0.2**
	4.9	3	**1.4**	**0.2**
2	7	3.2	**4.7**	**1.4**
	6.4	3.2	**4.5**	**1.5**
3	6.3	3.3	**6**	**2.5**
	6.5	3	**5.8**	**2.2**

The table above shows 6 instances from iris data set [8]. In the table, there are 3 clusters and 4 dimensions (attributes). The sample seems to be well-clustered as there is an obvious pattern difference among the clusters by looking at both attribute 3 and 4. Additionally, most instances in each cluster are relatively close to one another in all attributes.

Manuscript received September 25, 2012; revised November 11, 2012. This work was supported in part by the National Natural Science Foundation of China under Grant 61070085.

Barry Juans and Sheng-Uei Guan are with the Department of Computer Science and Software Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, China (e-mail: barry.juans10@student.xjtlu.edu.cn; Steven.Guan@xjtlu.edu.cn).

III. Hybrid of GA and Split-Fusion Algorithm

Flow of the clustering algorithm is as shown:

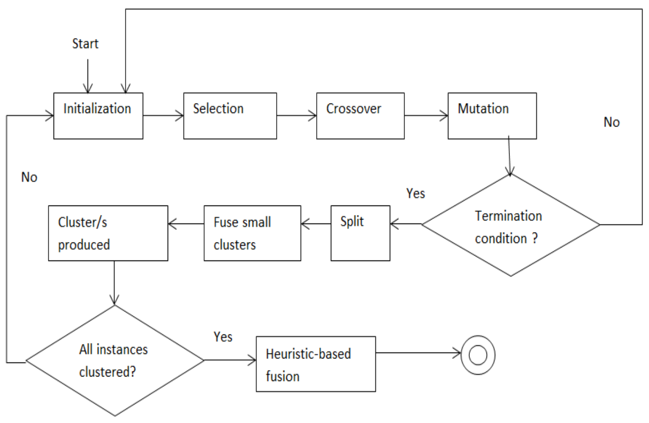

Fig. 1. Flowchart describing the GA and split-fusion hybrid algorithm.

A. Genetic Algorithm

In this algorithm, the GA processes (Initialization, Selection, Crossover, and Mutation) is used to find the best divider line for the data sets. The divider line is defined as vector of size *d*, with *d* being the number of attributes/dimensions of the dataset, plus a constant *c* and a Boolean variable *b*. The divider line itself functions similar to a vector dot product. As an example, for an instance defined as [a_1, a_2, a_3 … a_d] and the divider line defined as [l_1, l_2, l_3 … l_d, *c*, *b*], the calculation will be:

$$Z = a_1 \times l_1 + a_2 \times l_2 + a_3 \times l_3 \ldots + a_d \times l_d + c \qquad (1)$$

If the value of *Z* is larger than zero and the Boolean value *b* is positive (true), then the instance is placed into the ACCEPTED cluster, otherwise it is placed into the REJECTED. Conversely, if the Boolean value is negative (false), the instance will be REJECTED; it will be ACCEPTED only if the value of *Z* is less than zero. The best divider line is the one which will result in the cluster with the best GA fitness value.

The GA fitness calculation uses the concept of majority voting for calculation, and is based on the distance to the centroid of the ACCEPTED cluster relative to the REJECTED cluster.

For each instance, the fitness calculation is:

$$y = \sum_{i \in ACC} drej_i/(drej_i + dacc_i) \qquad (2)$$

y is calculated using (2) if majority of attributes are closer to centroid of ACCEPTED cluster; otherwise:

$$y = -(\sum_{i \in REJ} dacc_i/(drej_i + dacc_i)) \qquad (3)$$

dacc refers to the absolute distance between the instance's attribute and ACCEPTED centroid's attribute. Conversely, *drej* is the absolute distance between the instance's attribute and REJECTED centroid's attribute. The value of *dacc* is inversely correlated with value of drej.

ACC in here refers to the number of attributes that are closer to the centroid of the ACCEPTED cluster and *i* refers to one of the ACC attributes (in the first equation); therefore, if the majority of attributes of an instance are closer to the ACCEPTED, *y* will be positive, otherwise it will be a negative value. Since the algorithm needs a fitness function for the cluster and not for each instance, the cluster fitness is defined as:

$$fitness = \sum_{i \in ACCEPTED} Y_i \qquad (4)$$

Basically, fitness is the sum of fitness of each instance in the ACCEPTED cluster.

The whole GA processes will be looped until the ACCEPTED cluster's fitness in (4) no longer increases.

B. Split and Fusion

After the whole GA processes are completed, all the instances belonging to the ACCEPTED cluster are removed from the original dataset and evaluated for splitting fitness before the actual splitting is done. The splitting fitness *S* is calculated as:

For each attribute $j \in D$ (*D* is the set containing all attributes):

$$d_j = \sum_{i \in ACCEPTED} (-dacc_{ij}) \qquad (5)$$

$$S = \sum(Top\ half\ highest\ values\ of\ d_j) \qquad (6)$$

In other words, the splitting fitness is the sum of the top 50 percent attributes that are closest to the cluster centroid.

Other than the splitting fitness mentioned above, there is also another fitness value S_2 that is used; its calculation is:

$$S_2 = \sum(Top\ half\ lowest\ values\ of\ d_j) \qquad (7)$$

If *S* uses the attributes that are closest to the centroid, S_2 uses attributes that are furthest away from the centroids.

Similar to the splitting fitness, the split method is also based on attribute value. First the algorithm will find the attribute that can be used for splitting base:

For each attribute $j \in D$ (*D* is the set containing all attributes):

$$e_j = \sum_{i \in ACCEPTED} |dacc_{ij}| \qquad (8)$$

If ($S_2/S <= 5$) then (8) is used, otherwise:

$$e_j = \sum_{i \in ACCEPTED} |dacc_{ij}| / (|centroid_j| + |i_j|) \qquad (9)$$

If S_2 value in (7) is significantly larger than *S* in(6), it shows that there is significant difference among the attributes' standard deviation; in order to resolve this issue, the value of e_j in (9) will be divided by the sum of (centroid's attribute value and the instance's attribute value).

The attribute that will be used for the splitting base will be the attribute which has the highest value of e_j. Each instance in the ACCEPTED cluster is then split into 2 clusters; the first cluster contains all instances whose selected attribute is higher than the centroid's, and the second cluster contains the remaining instances. The algorithm will then try to adjust the each of the centroid's attribute value by shifting all the instances among the 2 clusters until the centroids' values no longer change. This part of splitting algorithm is similar to the k-means method.

After splitting, the splitting fitness S_2 of each of the 2 clusters is calculated. If the average of the 2 values is higher than the $S_2$of original cluster, then the split is considered successful, otherwise the algorithm will roll back to its pre-split state. The algorithm will keep on repeating the splitting phase for the whole clusters until rollback occurs.

Splitting will turn the original cluster into many smaller clusters. The next part of the algorithm will merge the clusters that are deemed too small back into the larger-sized ones. Heuristic-based equation is used for deciding on whether the cluster is small enough:

$$limit = X/\sqrt{X/5} \tag{10}$$

X is the total number of instances in the original data set. Basically, any cluster with size lesser than limit in (10) will be broken down into individual instances and each instance will be merged with the cluster whose centroid is closest to it. The closeness calculation is also based on the top 50 percent attributes that are furthest away from it, similar to S_2in (7). The equation is as such:

For each attribute $j \in D$ (D is the set containing all attributes) and $i \in C$ (C is the set containing all clusters):

$$dist_{ij} = |i_j - centroid_{ij}| \tag{11}$$

If ($S_2/S <= 5$), (11) is used; otherwise:

$$dist_{ij} = (|i_j - centroid_{ij}|)/(|centroid_{ij}| + |i_j|) \tag{12}$$

Distance to cluster i is then calculated as:

$$distance_i = \sum(Top\ half\ highest\ values\ of\ dist_{ij}) \tag{13}$$

Each instance will be merged with the cluster with smallest distance value. To summarize the split and fusion phase:

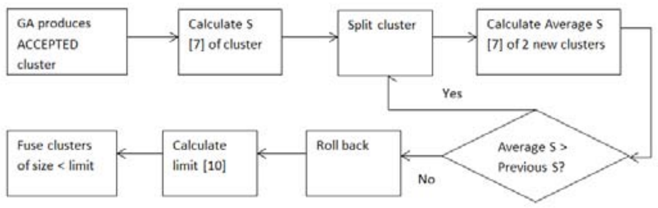
GA produces ACCEPTED cluster
Calculate S [7] of cluster
Split cluster
Calculate Average S [7] of 2 new clusters
Yes
Average S > Previous S?
No
Roll back
Calculate limit [10]
Fuse clusters of size < limit

Fig. 2. Flowchart describing the Split-Fusion phase.

The split and fusion phase will be repeated as long as there is an instance that is left un-clustered.

C. *Heuristic-Based Fusion*

The split and fusion phase will result in several clusters; the heuristic-based fusion aims to find the optimal number of clusters by continuously fusing the smallest-sized cluster until the termination condition is reached. The termination condition in this case is when the post-fusion average value of S in(6) for all clusters multiplied by 0.9 is lower than the pre-fusion average value of S. (v is pre-fusion S and $v_2$is post-fusion S)

$$(0.9 \times v_2) > v \tag{14}$$

While the above-condition is true, the algorithm will keep on fusing the smallest cluster.

IV. Experimental Results

The whole algorithm is programmed in Java language. Using the real world datasets such as iris and wine dataset [8], 30 experiments are conducted on each dataset. Diagrams are taken from [9].

TABLE II: Experiment Results

Algorithm	Number of Mis-clustered Patterns	
	Iris (150 patterns, 3 clusters)	Wine (178 patterns, 3 clusters)
Hybrid of GA and Split-Fusion	**8**	**17**
Recursive SOMs	8	51

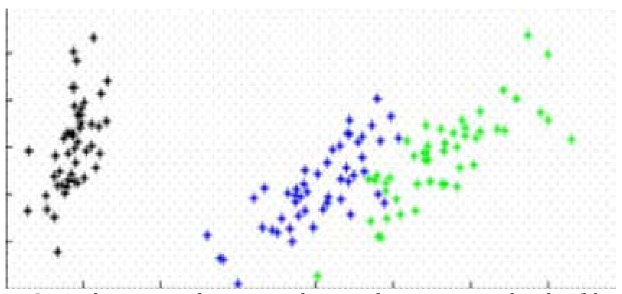

Fig. 3. Iris dataset samples projected over 2 dimensions. Each color (blue, black, and green) represent a cluster.

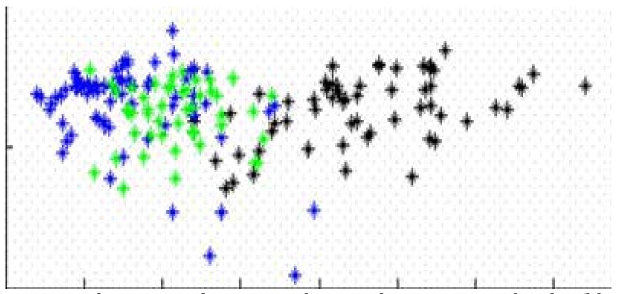

Fig. 4.Wine dataset samples projected over 2 dimensions. Each color (blue, black, and green) represent a cluster.

V. Conclusion

Experimental results have shown that the algorithm is capable of clustering real-world dataset accurately. Iris dataset is considered the benchmark dataset for clustering and it seems that the algorithm can cluster accurately; the same can be said for the more challenging wine dataset.

However, since the algorithm is reliant on heuristics, the size of initial dataset may significantly affect the final clustering results. If the size is too large, the algorithm may create more clusters than needed; on the other hand, if the size is too small, lesser clusters will be formed. In order to improve the overall accuracy of the algorithm, the heuristics equations need to be improved.

Acknowledgement

This research is supported by the National Natural Science Foundation of China under Grant 61070085.

References

[1] J. Han and M. Kamber, (2001) Data Mining: Concepts and Techniques. Morgan Kaufmann, CA, USA.

[2] K. Ramanathan and S. U. "Guan,Clustering and Combinatorial Optimization in Recursive Supervised Learning," *Journal of Combinatorial Optimization*, vol. 13, no. 2, pp. 137-152, 2007.

[3] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognition*, vol. 33, pp. 1455-1465, 2000.

[4] A. Rajaraman and J. D. Ullman. Clustering. Mining of Massive Datasets Chapter 7, pp. 221-260, 2011.

[5] L. Parsons, E. Haque, and H. Liu, "A Review.ACM SIGKDD Explorations Newsletter - Special issue on learning from imbalanced datasets," *Subspace Clustering for High Dimensional Data,* vol. 6, no. 1, pp. 90-105, 2004.

[6] A. Bhattacharya, and R. K. De, "Divisive Correlation Clustering Algorithm (DCCA) for groupingof genes: detecting varying patterns in expression profiles," *Bioinformatics,* vol.24. no.11, pp. 1359–1366, 2008.

[7] D. W. Kim, K. H. Lee, and D. Lee, "Detecting clusters of different geometrical shapes inmicroarray gene expression data," *Bioinformatics*, vol. 21, pp.1927–1934, 2005.

[8] UCI Machine Learning Repository: [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[9] Ramanathan, K. and Guan, S. U. (2006). Recursive Self Organizing Maps with Hybrid Clustering. IEEE CIS.

Sheng-Uei Guan received his M.Sc. & Ph.D. from the University of North Carolina at Chapel Hill. He is currently a professor and head of the computer science and software engineering department at XianJiaotong-LiverpoolUniversity. Before joining XJTLU, he was a professor and chair in intelligent systems at Brunel University, UK.

Prof. Guan has worked in a prestigious R&D organization for several years, serving as a design engineer, project leader, and manager. After leaving the industry, he joined Yuan-ZeUniversity in Taiwan for three and half years. He served as deputy director for the ComputingCenter and the chairman for the Department of Information & Communication Technology. Later he joined the Electrical & Computer Engineering Department at National University of Singapore as an associate professor.

Barry Juans is is currently an undergraduate student at Xi'an Jiaotong–Liverpool University.