A NOVEL APPROACH TO CLUSTERING USING GENTIC ALGORITH

Rahul Kala, Anupam Shukla, Ritu Tiwari

Department of Information Technology
Indian Institute of Information Technology and Management Gwalior,
Gwalior, Madhya Pradesh, India

rahulkalaiiitm@yahoo.co.in, dranupamshukla@gmail.com, rt_twr@yahoo.co.in
Ph: +91-9993746487

Citation: R. Kala, A. Shukla, R. Tiwari (2010) A Novel Approach to Clustering using Genetic Algorithm, *International Journal of Engineering Research and Industrial Applications* 3(1), 81-88.

Final Version Available At:
http://www.ascent-journals.com/IJERIA/Vol3No1/Paper_7.pdf

ABSTRACT

Clustering is one of the most fundamental algorithms which have got huge applications especially in the area of Neuro Fuzzy Systems, Data Analysis, Linear Vector Quantization, Bio-informatics etc. Various approaches exist for clustering of data. A few of the commonly used approaches are K-Means clustering, Fuzzy C-Means Clustering, Subtractive Clustering, etc. Clustering may involve varied uses. Based on these we may be required to control our clustering algorithm. The earlier mentioned algorithms do not give us this adaptability. Hence we propose the use of Genetic Algorithms for the purpose of clustering. In order to test the working of the proposed algorithm, we coded it and made it work on a randomized data set. We found that in general use, the performance of the

algorithm was equivalent to that of the k-Means clustering. We modified the fitness function and found that the algorithm behaved as per expectations and generated better clusters that the k-Means clustering algorithm, that were desired.

Keywords

Clustering, Genetic Algorithms, K-Means Clustering, Fuzzy C-Means Clustering, Mutation, Crossover, Data Analysis

1. Introduction

Ever since the explosion of data started, the need and necessity of clustering gained importance. Clustering is needed by many basic algorithms as a prerequisite. Clustering enables us to group up similar data into clusters. If the original input data is too large to be worked upon, the clusters provide a good way where we can represent big groups of data by their representatives. Clustering hence is of ample importance to the industry [3, 6, 10, 12].

Clustering algorithms [16] usually apply an iterative approach to find the cluster center. The iterative approach selects arbitrary cluster centers and then at every iteration tries to improve the center. If the improvement is too low or the number of iterations spent has been too large, the algorithm stops and the position of the cluster centers and clusters at that point are regarded as the final positions for the clusters.

In this paper we have used Genetic Algorithms to solve the purpose of Clustering. Just like Fuzzy C Means Clustering and K-Means Clustering, we assume here that the number of clusters is known. We first arbitrarily place the centers of the clusters anywhere in the space. Then we genetically keep modifying the solutions. The greatest advantage of genetic algorithms is that we can alter the fitness function to change the behavior of the algorithm.

There are many clustering algorithms that are extensively being used like K-Means Clustering, Fuzzy C Means clustering, etc [4, 7, 8]. Various approaches have been adopted to enhance the speed of the algorithms by better modeling [2,

9, 13]. Other algorithms include hierarchical clustering, Mixture of Gaussians, Correlation Clustering, Sublinear Clustering, QT (quality threshold) clustering, etc. A lot of research is going on for specific Applications of clustering. Time taken to form the cluster is one of the areas of major research [1, 7, 13]. Genetic Algorithms also have been applied to an extent for the problem of clustering [5, 6, 11]. There is very little research in this domain.

2. The Algorithm

In this section we would discuss the algorithm and all its steps. Here we have proposed the use of Genetic Algorithms for clustering. Genetic Algorithms also like the other algorithms normally used for clustering are iterative in nature. They keep improving the solution at every iteration.

2.1 Individual Representation

In Genetic Algorithms, it is important to express how a solution is represented. In this problem are solution is a set of k nodes. Here k is the number of clusters we need to form and this is specified at the start itself. A solution may thus be represented by $S = \{C_1, C_2, C_3 \ldots C_k\}$. Here $C_1, C_2, C_3 \ldots C_k$ are the centers of the clusters.

A center in turn is a collection of values of the data in n dimensions. Hence a center may be represented as given by $C = \langle I_1, I_2, I_3, I_4 \ldots I_n \rangle$. We may say that a solution may be represented as $S = \{\langle I_{11}, I_{12}, I_{13}, I_{14} \ldots I_{1n} \rangle, \langle I_{21}, I_{22}, I_{23}, I_{24} \ldots I_{2n} \rangle, \ldots\ldots, \langle I_{k1}, I_{k2}, I_{k3}, I_{k4} \ldots I_{kn} \rangle\}$. Every solution has two major properties. These are discussed in the next two subsections

2.1.1 Mean of coordinates: It is mandatory for the point stated as the center of the cluster in solution to be on the arithmetic mean of the points in that cluster. In order to ensure this property, we make sure that after every genetic operation; we recalculate the clusters and move the center to the mean of the points in the cluster.

2.1.2 Sorted Centers: The second fundamental property that exists in every solution is that the centers are arranged in a sorted manner. This sorting is based on the sorting algorithm given below

Sort(C)
SortedSequence = φ
While C ≠ φ
Do
If SortedSequence = φ
then Add top leftmost center C_i in SortedSequence
else Add center C_i in SortedSequence that is closest to any center existing in SortedSequence
Delete C_i from C
Return SortedSequence

2.2 Initial Solutions

We generate random set of solutions. All the centers generated lie within the solution space in the n dimensional input space. We also ensure that the above mentioned properties are followed by these solutions that are generated. A good number of initial solutions are generated to ensure that all types of features or centers are present in the solution pool.

2.3 Selection

Selection refers to the selection of parents that ultimately evolve the new solution. With a probability of 0.5, we select amongst the best solution generated so far as one of the parent and any random solution as the other parent. With a probability of 0.5, we select any two distinct solutions as the two parents.

2.4 Crossover

The crossover is done on the parents to generate the new solution. The cross over operation is described below.

CrossOver(p,q)

P ← Centers in solution p
Q ← Centers in solution q
With a probability 0.5 Generate s by mean of P and Q
With a probability 0.5 Generate s by first half centers from P and second half from Q

Return s

2.5 Mutation

In this algorithm we perform mutation by moving all the centers of the selected individuals randomly in some direction by a random amount. This amount can be a maximum of 10% of the size of the input space. If the center after movement lies outside the input space for any dimension, then the movement for that dimension is cancelled.

2.6 Fitness Function

This algorithm does not insist on a specific fitness function. The fitness function may be changed by the user to better adapt to the needs of the algorithm. In fact this is the best part of this algorithm that we can generate different types of results by changing the fitness function. The algorithm tries to optimize the fitness function, thereby giving us the desired results.

3. Comparison with K-Means Clustering

In order test the working of the algorithm, we coded the algorithm and made it run using JAVA APPLETS. The inputs were shown as crosses and the centers that the algorithm finds out were shown with filled rectangles and circles. Red rectangles were the outputs of the GA and the blue circles were the outputs of the K-Means algorithm. In the next sub sections we would be discussing the 3 types of test cases that were processed. All tests were performed on a randomly generated data set based on heuristic based generation. Two runs of each test case were conducted.

3.1 General Clustering

Our first aim was to imitate the working of the K-Means Clustering algorithm as it is. We chose a much generalized fitness function for this purpose. The fitness function was the sum of the lengths of the input points to their cluster centers from every point that was input. We executed the two algorithms and saw the results. The two algorithms had almost similar results. Both of them were very efficient in

time. The results of the two algorithms for two random runs are shown in Figure 1. It may be seen in Figure 1(b) that both the outputs overlap. The closeness of the points in the two algorithms shows that the two algorithms are very similar in their outputs.

3.2 Clustering with equal number of elements per cluster

Now we try to cluster the data in a way that there is almost same number of elements in every cluster. For this we modify only the fitness function in our algorithm which mimics the disparity in the number of elements in clusters. The K-means algorithm is not modified and the same algorithm is used for this purpose only. The results clearly show that our algorithm tried to generate solutions that had equal number of elements. This is shown in Figure 2 for two random runs of the algorithm.

3.3 Clustering with unequal number of elements per cluster

We also try to do the opposite of the above. Here we try to get the maximum number of elements in a single cluster and very few in the other. The fitness function was reversed as that presented in section 3.2. The results again support the cause. The results are shown in figure two random runs are shown in Figure 3.

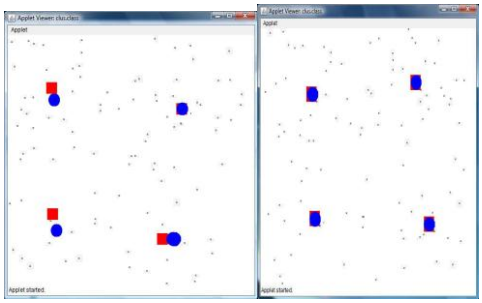

Figure 1(a) and 1(b): Genetic Algorithm and K-Means clustering for general fitness function for two random runs

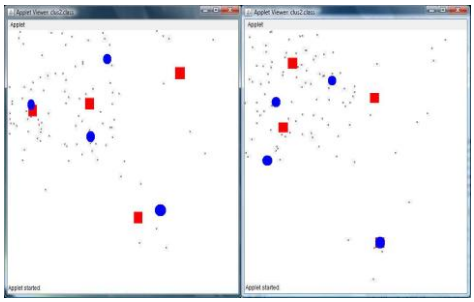

Figure 2(a) and 2(b): Genetic Algorithm and K-Means clustering for fitness function to accommodate almost same number of elements per cluster for two random runs

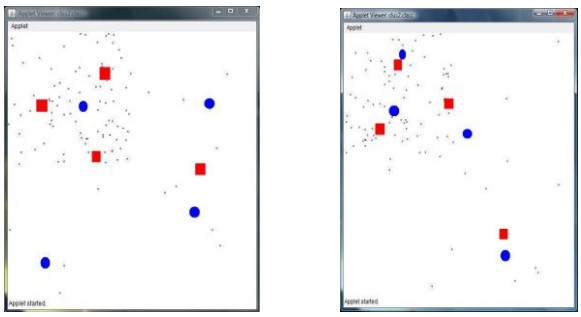

Figure 3(a) and 3(b): Genetic Algorithm and K-Means clustering for fitness function to accommodate maximum difference in number of elements per cluster

4. Conclusion

In this paper we presented a novel approach to clustering using Genetic Algorithms. Here we used Genetic Algorithms mainly for the purpose of controlled nature of clustering that was not being done by any of the other clustering algorithm. We coded the algorithm using JAVA APPLETS and the results were recorded. We compared the results with the standard K-Means Clustering. We found that our algorithm worked much better than the standard K-Means Clustering algorithm and gave more intended results.

Here we have been able to control the working of the clustering using Genetic Algorithms. But with the increase of the input sizes or dimensionality the Genetic Algorithms may get computationally expensive. Also Particle Swarm Optimizations may give even better results than the Genetic Algorithms. These issues may be dealt in the future.

Acknowledgement

The work has been supported and sponsored by Indian Institute of Information Technology and Management Gwalior

References

[1] Aggarwal, Charu C.; Procopiuc, Cecilia; Wolf, Joel L.; Yu, Philip S. & Park, Jong Soo, “Fast Algorithms for Projected Clustering”, ACM SIGMOD Record, Vol 28, Issue 2, June 1999, pp 61-72.

[2] Alsabti, Khaled; Ranka, Sanjay & Singh, Vineet, “An Efficient K-Means Clustering Algorithm”, Proceedings of IPPS/SPDP Workshop on High Performance Data Mining, 1998.

[3] Bolelli, Levent ; Ertekin, Seyda & Giles, C. Lee, “Clustering Scientific Literature Using Sparse Citation Graph Analysis”, Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Sept 2006, pp 30-41.

[4] Castro, Vladimir Estivill, “Why so many clustering algorithms — A Position Paper”, ACM SIGKDD Explorations Newsletter, Vol 4, Issue 1, June 2002, pp 65-75.

[5] Chen, Qingzhan; Han, Jianghong; Lai, Yungang; He, Wenxiu & Mao, Keji, “Clustering Problem Using Adaptive Genetic Algorithm”, ICNC 2005, Springer Lecture Notes in Computer Science, Vol. 3612, 2005, pp. 782 – 786.

[6] Doval, D; Mancoridis, S. & Mitchell, B. S., “Automatic Clustering of Software Systems using a Genetic Algorithm”, Proceedings of the Software Technology and Engineering Practice, 1999, pp 73.

[7] Eick, Christoph F.; Zeidat, Nidal & Zhao, Zhenghong, “Supervised Clustering – Algorithms and Benefits”, Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), 2004

[8] Fung, Glenn, “A Comprehensive Overview of Basic Clustering Algorithms”, May 2001

[9] Kanungo, Tapas; Netanyahu, Nathan S. & Wu, Angela Y., “An Efficient k-Means Clustering Algorithm: Analysis and Implementation”, IEEE Transactions on Pattern Analysis and Machine Learning, Vol. 24, No. 7, July 2002

[10] Maddah, Mahnaz; Wells, William M.; Warfield, Simon K.; Westin, Carl-Fredrik & Grimson, W. Eric L., “Probabilistic Clustering and Quantitative Analysis of White Matter Fiber Tracts”, Inf Process Med Imaging, vol. 20, 2007, pp 372-83.

[11] Maulik U., Bandyopadhyay S., “Genetic algorithm-based clustering technique”, Pattern Recognition, Vol. 33, 2000, pp 1455-1465

[12] Osinski, Stanis law; Stefanowski, Jerzy & Weiss, Dawid, “Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition”, Intelligent Information Systems, 2004, pp 359-368

[13] Zhang, Bin; Hsu, Meichun & Dayal, Umeshwar, “K-Harmonic Means - A Data Clustering Algorithm”, 1999