PERGAMON

Pattern Recognition 33 (2000) 1455–1465

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Genetic algorithm-based clustering technique

Ujjwal Maulik[a], Sanghamitra Bandyopadhyay[b,*,1]

[a]*Department of Computer Science, Government Engineering College, Kalyani, Nadia, India*
[b]*Machine Intelligence Unit, Indian Statistical Institute, 203, B.T. Road, Calcutta - 700 035, India*

Received 24 June 1998; received in revised form 29 April 1999; accepted 29 April 1999

Abstract

A genetic algorithm-based clustering technique, called GA-clustering, is proposed in this article. The searching capability of genetic algorithms is exploited in order to search for appropriate cluster centres in the feature space such that a similarity metric of the resulting clusters is optimized. The chromosomes, which are represented as strings of real numbers, encode the centres of a fixed number of clusters. The superiority of the GA-clustering algorithm over the commonly used *K*-means algorithm is extensively demonstrated for four artificial and three real-life data sets. © 2000 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved.

Keywords: Genetic algorithms; Clustering metric; *K*-means algorithm; Real encoding; Euclidean distance

1. Introduction

Genetic algorithms (GAs) [1–4] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, having a large amount of implicit parallelism. GAs perform search in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem.

In GAs, 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* funtion 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 GAs along with the programming structure used can be found in Ref. [4]. GAs have applications in fields as diverse as VLSI design, image processing, neural networks, machine learning, jobshop scheduling, etc. [5–10].

In the area of pattern recognition, there are many tasks involved in the process of analyzing/identifying a pattern which need appropriate parameter selection and efficient search in complex spaces in order to obtain optimum solutions. Therefore, the application of GAs for solving certain problems of pattern recognition (which need optimization of computation requirements, and robust, fast and close approximate solution) appears to be appropriate and natural. Research articles in this area have started to come out [11,12]. Recently, an application of GAs has been reported in the area of (supervised) pattern classification in $\mathscr{R}^N$ [13,14] for designing a *GA-classifier*. It attempts to approximate the class boundaries of a given data set with a fixed number (say H) of hyperplanes in such a manner that the associated misclassification of data points is minimized during training.

When the only data available are unlabeled, the classification problems are sometimes referred to as

* Corresponding author. Present address: School of Computer Science and Engineering, University of New South Wales, Sydney 2052, Australia; Tel.: 00-61-2-9385-3975; fax: 00-61-2-9385-1814.

E-mail addresses: umaulik@hotmail.com (U. Maulik), sanghami@cse.unsw.edu.au (S. Bandyopadhyay).

[1] On leave from Indian Statistical Institute.

PII: S0031-3203(99)00137-5

unsupervised classification. Clustering [15–19] is an important unsupervised classification technique where a set of patterns, usually vectors in a multi-dimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. For this it is necessary to first define a measure of similarity which will establish a rule for assigning patterns to the domain of a particular cluster centre. One such measure of similarity may be the Euclidean distance **D** between two patterns **x** and **z** defined by $D = \|\mathbf{x} - \mathbf{z}\|$. Smaller the distance between **x** and **z**, greater is the similarity between the two and vice versa.

Several clustering techniques are available in the literature [19,20]. Some, like the widely used *K*-means algorithm [19], optimize of the distance criterion either by minimizing the within cluster spread (as implemented in this article), or by maximizing the inter-cluster separation. Other techniques like the graph theoretical approach, hierarchical approach, etc., are also available which perform clustering based on other criteria. These are discussed in brief in Section 2. Extensive studies dealing with comparative analysis of different clustering methods [21] suggest that there is no general strategy which works equally well in different problem domains. However, it has been found that it is usually beneficial to run schemes that are simpler, and execute them several times, rather than using schemes that are very complex but need to be run only once [21]. Since our aim is to propose a clustering technique based on GAs, a criterion is required whose optimization would provide the final clusters. An intuitively simple criterion is the within cluster spread, which, as in the *K*-means algorithm, needs to be minimized for good clustering. However, unlike the *K*-means algorithm which may get stuck at values which are not optimal [22], the proposed technique should be able to provide good results irrespective of the starting configuration. It is towards this goal that we have integrated the simplicity of the *K*-means algorithm with the capability of GAs in avoiding local optima for developing a GA-based clustering technique called GA-clustering algorithm. It is known that elitist model of GAs provide the optimal string as the number of iterations goes to infinity [23] when the probability of going from any population to the one containing the optimal string is greater than zero. Therefore, under limiting conditions, a GA based clustering technique is also expected to provide an optimal clustering with respect to the clustering metric being considered.

Experimental results comparing the GA-clustering algorithm with the *K*-means algorithm are provided for several artificial and real-life data sets. Since our purpose is to demonstrate the effectiveness of the proposed technique for a wide variety of data sets, we have chosen artificial and real-life data sets with both overlapping and non-overlapping class boundaries, where the number of dimensions ranges from two to ten and number of clusters ranges from two to nine. Note that we are encoding the centres of the clusters, which will be floating point numbers, in the chromosomes. One way in which this could have been implemented is by performing real representation with a binary encoding [24]. However, in order to keep the mapping between the actual cluster centres and the encoded centres straight forward, for convenience we have implemented real coded GAs over here [3]. (In this context one may note the observations in Ref. [25] after they experimentally compared binary and floating point representations in GAs. They found that floating point representation was faster, consistent and provided a higher degree of precision.)

2. Clustering

Clustering in N-dimensional Euclidean space $\mathbb{R}^N$ is the process of partitioning a given set of n points into a number, say K, of groups (or, clusters) based on some similarity/dissimilarity metric. Let the set of n points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ be represented by the set S and the K clusters be represented by $C_1, C_2, \ldots, C_K$. Then

$$C_i \neq \emptyset \quad \text{for } i = 1, \ldots, K,$$

$$C_i \cap C_j = \emptyset \quad \text{for } i = 1, \ldots, K, j = 1, \ldots, K \text{ and } i \neq j$$

$$\text{and } \bigcup_{i=1}^{K} C_i = S.$$

Some clustering techniques that are available in the literature are *K*-means algorithm [19], branch and bound procedure [26], maximum likelihood estimate technique [27] and graph theoretic approaches [28]. The *K*-means algorithm, one of the most widely used ones, attempts to solve the clustering problem by optimizing a given metric. The branch and bound procedure uses a tree search technique for searching the entire solution space of classifying a given set of points into a fixed number of clusters, along with a criterion for eliminating subtrees which do not contain the optimum result. In this scheme, the number of nodes to be searched becomes huge when the size of the data set becomes large. In these cases, a proper choice of the criterion for eliminating subtrees becomes crucial [20]. The maximum likelihood estimate technique performs clustering by computing the posterior probabilities of the classes after assuming a particular distribution of the data set. In the graph theoretic approach, a directed tree is formed among the data set by estimating the density gradient at each point. The clustering is realized by finding the valley of the density function. It is known that the quality of the result depends wholly on the quality of the estimation technique for the density gradient, particularly in the low-density area of the valley.

Our aim in this article has been to propose a clustering methodology which will not assume any particular underlying distribution of the data set being considered, while, as already mentioned in Section 1, it should be conceptually simple like the K-means algorithm. On the other hand, it should not suffer from the limitation of the K-means algorithm which is known to provide sub optimal clusterings depending on the choice of the initial clusters. Since the principles of the K-means algorithm are utilized for devising such a technique, along with the capability of GAs for providing the requisite perturbation to bring it out of the local optima, we have compared the performance of the former with that of the proposed technique. The steps of the K-means algorithm are therefore first described in brief.

Step 1: Choose K initial cluster centres $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_K$ randomly from the n points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$.

Step 2: Assign point $\mathbf{x}_i$, $i = 1, 2, \ldots, n$ to cluster $C_j, j \in \{1, 2, \ldots, K\}$ iff

$$\|\mathbf{x}_i - \mathbf{z}_j\| < \|\mathbf{x}_i - \mathbf{z}_p\|, \; p = 1, 2, \ldots, K, \text{ and } j \neq p.$$

Ties are resolved arbitrarily.

Step 3: Compute new cluster centres $\mathbf{z}_1^*, \mathbf{z}_2^*, \ldots, \mathbf{z}_K^*$ as follows:

$$\mathbf{z}_i^* = \frac{1}{n_i} \sum_{x_j \in C_i} \mathbf{x}_j, \quad i = 1, 2, \ldots, K,$$

where n_i is the number of elements belonging to cluster C_i.

Step 4: If $\mathbf{z}_i^* = \mathbf{z}_i$, $i = 1, 2, \ldots, K$ then terminate. Otherwise continue from step 2.

Note that in case the process does not terminate at Step 4 normally, then it is executed for a maximum fixed number of iterations.

It has been shown in Ref. [22] that K-means algorithm may converge to values that are not optimal. Also global solutions of large problems cannot be found with a reasonable amount of computation effort [29]. It is because of these factors that several approximate methods are developed to solve the underlying optimization problem. One such method using GAs is described in the next section.

3. Clustering using genetic algorithms

3.1. Basic principle

The searching capability of GAs has been used in this article for the purpose of appropriately determining a fixed number K of cluster centres in $\mathbb{R}^N$; thereby suitably clustering the set of n unlabelled points. The clustering metric that has been adopted is the sum of the Euclidean distances of the points from their respective cluster centres. Mathematically, the clustering metric $\mathcal{M}$ for the K clusters $C_1, C_2, \ldots, C_K$ is given by

$$\mathcal{M}(C_1, C_2, \ldots, C_K) = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|\mathbf{x}_j - \mathbf{z}_i\|.$$

The task of the GA is to search for the appropriate cluster centres $\mathbf{z}_1, \mathbf{z}_2, \ldots, z_K$ such that the clustering metric $\mathcal{M}$ is minimized.

```
Begin
1.  t=0
2.  initialize population P(t)
3.  compute fitness P(t)
4.  t = t+1
5.  if termination criterion achieved go to step 10
6.  select P(t) from P(t-1)
7.  crossover P(t)
8.  mutate P(t)
9.  go to step 3
10.  Output best and stop
End
```

Fig. 1. Basic steps in GAs.

3.2. GA-clustering algorithm

The basic steps of GAs, which are also followed in the GA-clustering algorithm, are shown in Fig. 1. These are now described in detail.

3.2.1. String representation

Each string is a sequence of real numbers representing the K cluster centres. For an N-dimensional space, the length of a chromosome is $N*K$ words, where the first N positions (or, genes) represent the N dimensions of the first cluster centre, the next N positions represent those of the second cluster centre, and so on. As an illustration let us consider the following example.

Example 1. Let $N = 2$ and $K = 3$, i.e., the space is two-dimensional and the number of clusters being considered is three. Then the chromosome

51.6 72.3 18.3 15.7 29.1 32.2

represents the three cluster centres (51.6, 72.3), (18.3, 15.7) and (29.1, 32.2). Note that each real number in the chromosome is an indivisible gene.

3.2.2. Population initialization

The K cluster centres encoded in each chromosome are initialized to K randomly chosen points from the

data set. This process is repeated for each of the P chromosomes in the population, where P is the size of the population.

3.2.3. Fitness computation

The fitness computation process consists of two phases. In the first phase, the clusters are formed according to the centres encoded in the chromosome under consideration. This is done by assigning each point $\mathbf{x}_i$, $i = 1, 2, \ldots, n$, to one of the clusters C_j with centre $\mathbf{z}_j$ such that

$$||\mathbf{x}_i - \mathbf{z}_j|| < ||\mathbf{x}_i - \mathbf{z}_p||, p = 1, 2, \ldots, K, \text{ and } p \neq j.$$

All ties are resolved arbitrarily. After the clustering is done, the cluster centres encoded in the chromosome are replaced by the mean points of the respective clusters. In other words, for cluster C_i, the new centre $\mathbf{z}_i^*$ is computed as

$$\mathbf{z}_i^* = \frac{1}{n_i} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j, \quad i = 1, 2, \ldots, K.$$

These $\mathbf{z}_i^*$**s** now replace the previous $\mathbf{z}_i$**s** in the chromosome. As an illustration, let us consider the following example.

Example 2. The first cluster centre in the chromosome considered in Example 1 is (51.6, 72.3). With (51.6, 72.3) as centre, let the resulting cluster contain two more points, viz., (50.0, 70.0) and (52.0, 74.0) besides itself i.e., (51.6, 72.3). Hence the newly computed cluster centre becomes $((50.0 + 52.0 + 51.6)/3, \quad (70.0 + 74.0 + 72.3)/3) = (51.2, 72.1)$. The new cluster centre (51.2, 72.1) now replaces the previous value of (51.6, 72.3).

Subsequently, the clustering metric $\mathcal{M}$ is computed as follows:

$$\mathcal{M} = \sum_{i=1}^{K} \mathcal{M}_i,$$

$$\mathcal{M}_i = \sum_{\mathbf{x}_j \in C_i} ||\mathbf{x}_j - \mathbf{z}_i||.$$

The fitness function is defined as $f = 1/\mathcal{M}$, so that maximization of the fitness function leads to minimization of $\mathcal{M}$.

3.2.4. 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.

3.2.5. Crossover

Crossover is a probabilistic process that exchanges information between two parent chromosomes for generating two child chromosomes. In this article single-point crossover with a fixed crossover probability of μ_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.

3.2.6. Mutation

Each chromosome undergoes mutation with a fixed probability μ_m. For binary representation of chromosomes, a bit position (or gene) is mutated by simply flipping its value. Since we are considering floating point representation in this article, we use the following mutation. A number δ in the range [0, 1] is generated with uniform distribution. If the value at a gene position is v, after mutation it becomes

$$v \pm 2 * \delta * v, \quad v \neq 0,$$

$$v \pm 2 * \delta, \quad v = 0.$$

The '+' or '−' sign occurs with equal probability. Note that we could have implemented mutation as

$$v \pm \delta * v.$$

However, one problem with this form is that if the values at a particular position in all the chromosomes of a population become positive (or negative), then we will never be able to generate a new chromosome having a negative (or positive) value at that position. In order to overcome this limitation, we have incorporated a factor of 2 while implementing mutation. Other forms like

$$v \pm (\delta + \varepsilon) * v,$$

where $0 < \varepsilon < 1$ would also have satisfied our purpose. One may note in this context that similar sort of mutation operators for real encoding have been used mostly in the realm of evolutionary strategies (see Ref. [3], Chapter 8).

3.2.7. Termination criterion

In this article the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of iterations. The best string seen upto the last generation provides the solution to the

clustering problem. We have implemented elitism at each generation by preserving the best string seen upto that generation in a location outside the population. Thus on termination, this location contains the centres of the final clusters.

The next section provides the results of implementation of the GA-clustering algorithm, along with its comparison with the performance of the *K*-means algorithm for several artificial and real-life data sets.

4. Implementation results

The experimental results comparing the GA-clustering algorithm with the *K*-means algorithm are provided for four artificial data sets (*Data 1*, *Data 2*, *Data 3* and *Data 4*) and three real-life data sets (*Vowel*, *Iris* and *Crude Oil*), respectively. These are first described below:

4.1. Artificial data sets

Data 1: This is a nonoverlapping two-dimensional data set where the number of clusters is two. It has 10 points. The value of K is chosen to be 2 for this data set.

Data 2: This is a nonoverlapping two-dimensional data set where the number of clusters is three. It has 76 points. The clusters are shown in Fig. 2: The value of K is chosen to be 3 for this data set.

Data 3: This is an overlapping two-dimensional triangular distribution of data points having nine classes where all the classes are assumed to have equal a priori probabilities ($=\frac{1}{9}$). It has 900 data points. The $X-Y$ ranges for the nine classes are as follows:

Class 1: $[-3.3, -0.7] \times [0.7, 3.3]$,
Class 2: $[-1.3, 1.3] \times [0.7, 3.3]$,
Class 3: $[0.7, 3.3] \times [0.7, 3.3]$,
Class 4: $[-3.3, -0.7] \times [-1.3, 1.3]$,

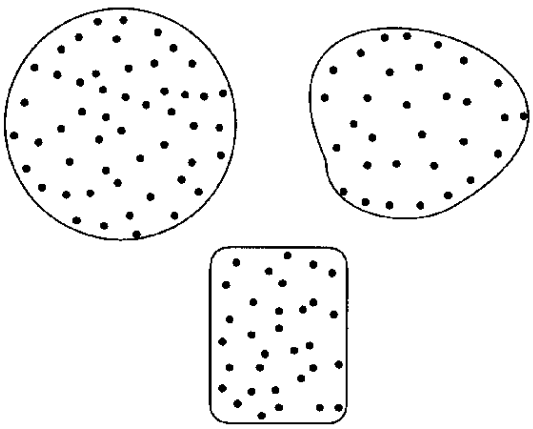

Fig. 2. *Data 2*.

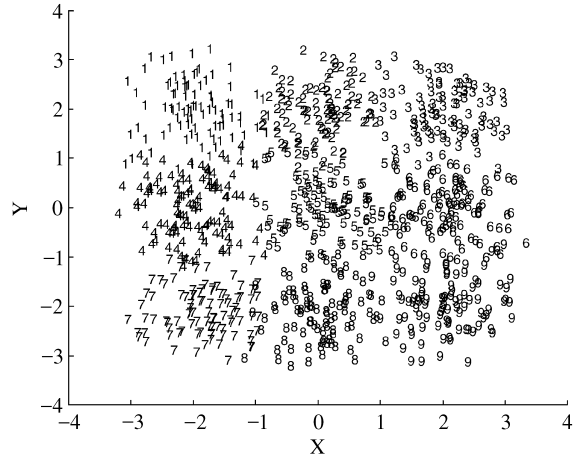

Fig. 3. *Data 3* ('1'—points from class 1, '2'—points from class 2, …, '9'—points from class 9).

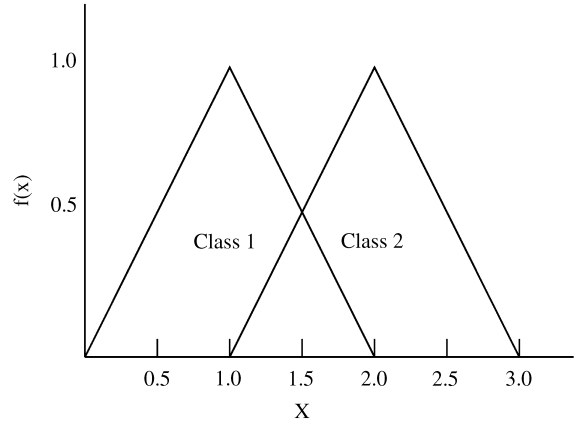

Fig. 4. Triangular distribution along the X-axis.

Class 5: $[-1.3, 1.3] \times [-1.3, 1.3]$,
Class 6: $[0.7, 3.3] \times [-1.3, 1.3]$,
Class 7: $[-3.3, -0.7] \times [-3.3, -0.7]$,
Class 8: $[-1.3, 1.3] \times [-3.3, -0.7]$,
Class 9: $[0.7, 3.3] \times [-3.3, -0.7]$.

Thus the domain for the triangular distribution for each class and for each axis is 2.6. Consequently, the height will be $\frac{1}{1.3}$ (since $12*2.6*\text{height} = 1$). The data set is shown in Fig. 3. The value of K is chosen to be 9 for this data set.

Data 4: This is an overlapping ten-dimensional data set generated using a triangular distribution of the form shown in Fig. 4 for two classes, 1 and 2. It has 1000 data points. The value of K is chosen to be 2 for this data set. The range for class 1 is $[0, 2] \times [0, 2] \times [0, 2] \ldots$ 10 times, and that for class 2 is $[1, 3] \times [0, 2] \times [0, 2] \ldots$ 9 times, with the corresponding peaks at (1, 1) and (2, 1). The

distribution along the first axis (X) for class 1 may be formally quantified as

$$f_1(x) = \begin{cases} 0 & \text{for } x \leqslant 0, \\ x & \text{for } 0 < x \leqslant 1, \\ 2 - x & \text{for } 1 < x \leqslant 2, \\ 0 & \text{for } x > 2. \end{cases}$$

for class 1. Similarly for class 2

$$f_2(x) = \begin{cases} 0 & \text{for } x \leqslant 1, \\ x - 1 & \text{for } 1 < x \leqslant 2, \\ 3 - x & \text{for } 2 < x \leqslant 3, \\ 0 & \text{for } x > 3. \end{cases}$$

The distribution along the other nine axes (Y_i, $i = 1, 2, \ldots, 9$) for both the classes is

$$f(y_1) = \begin{cases} 0 & \text{for } y_i \leqslant 0, \\ y_i & \text{for } 0 < y_i \leqslant 1, \\ 2 - y_i & \text{for } 1 < y_i \leqslant 2, \\ 0 & \text{for } y_i > 2. \end{cases}$$

4.2. Real-life data sets

Vowel data: This data consists of 871 Indian Telugu vowel sounds [30]. These were uttered in a consonant–vowel–consonant context by three male speakers in the age group of 30–35 years. The data set has three features F_1, F_2 and F_3, corresponding to the first, second and third vowel formant frequencies, and six overlapping classes $\{\delta, a, i, u, e, o\}$. The value of K is therefore chosen to be 6 for this data. Fig. 5 shows the distribution of the six classes in the F_1–F_2 plane.

Iris data: This data represents different categories of irises having four feature values. The four feature values represent the sepal length, sepal width, petal length and the petal width in centimeters [31]. It has three classes (with some overlap between classes 2 and 3) with 50 samples per class. The value of K is therefore chosen to be 3 for this data.

Crude oil data: This overlapping data [32] has 56 data points, 5 features and 3 classes. Hence the value of K is chosen to be 3 for this data set.

GA-clustering is implemented with the following parameters:

$\mu_c = 0.8$ $\mu_m = 0.001$. The population size P is taken to be 10 for *Data 1*, since it is a very simple data set, while it is taken to be 100 for the others. Note that it is shown in Refs. [15,29] if exhaustive enumeration is used to solve a clustering problem with n points and K clusters, then one requires to evaluate

$$\frac{1}{K}\sum_{j=1}^{K}(-1)^{K-j}j^n$$

partitions. For a data set of size 10 with 2 clusters, this value is $2^9 - 1 (= 511)$, while that of size 50 with 2 clusters is $2^{49} - 1$ (i.e. of the order of 10^{15}).

For K-means algorithm we have fixed a maximum of 1000 iterations in case it does not terminate normally. However, it was observed that in all the experiments the K-means algorithm terminated much before 1000 iterations.

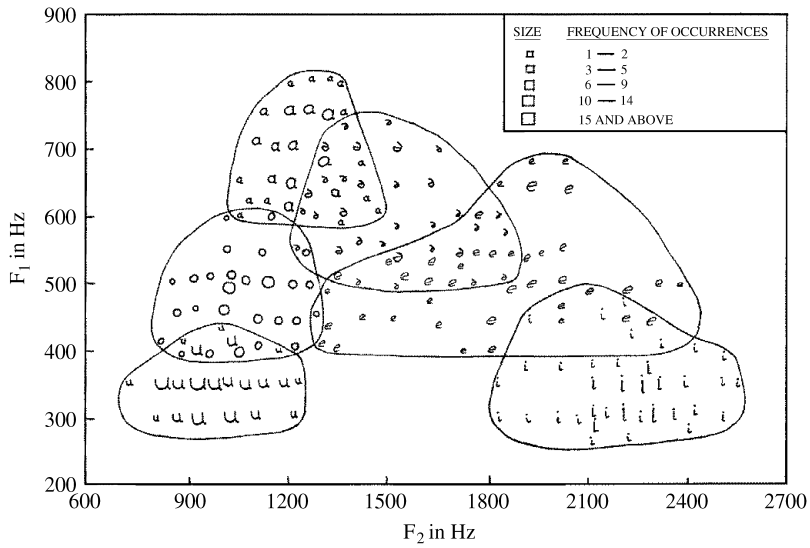

Fig. 5. *Vowel* data in the F_1–F_2 plane.

Table 1
$\mathcal{M}$ obtained by K-means algorithm for five different initial configurations for *Data 1* when $K = 2$

Initial configuration	K-means
1	5.383132
2	2.225498
3	2.225498
4	5.383132
5	2.225498

Table 2
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Data 1* after 100 iterations when $K = 2$

Initial population	GA-clustering
1	2.225498
2	2.225498
3	2.225498
4	2.225498
5	2.225498

Table 3
$\mathcal{M}$ obtained by K-means algorithm for five different initial configurations for *Data 2* when $K = 3$

Initial configuration	K-means
1	51.013294
2	64.646739
3	67.166768
4	51.013294
5	64.725676

Table 4
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Data 2* after 100 iterations when $K = 3$

Initial population	GA-clustering
1	51.013294
2	51.013294
3	51.013294
4	51.013294
5	51.013294

Table 5
$\mathcal{M}$ obtained by K-means algorithm for five different initial configurations for *Data 3* when $K = 9$

Initial configuration	K-means
1	976.235607
2	976.378990
3	976.378990
4	976.564189
5	976.378990

Table 6
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Data 3* after 100 iterations when $K = 9$

Initial population	GA-clustering
1	966.350481
2	966.381601
3	966.350485
4	966.312576
5	966.354085

Table 7
$\mathcal{M}$ obtained by K-means algorithm for five different initial configurations for *Data 4* when $K = 2$

Initial configuration	K-means
1	1246.239153
2	1246.239153
3	1246.236680
4	1246.239153
5	1246.237127

The results of implementation of the K-means algorithm and GA-clustering algorithm are shown, respectively, in Tables 1 and 2 for *Data 1*, Tables 3 and 4 for *Data 2*, Tables 5 and 6 for *Data 3*, Tables 7 and 8 for *Data 4*, Tables 9 and 10 for *Vowel*, Tables 11 and 12 for *Iris* and Tables 13 and 14 for *Crude Oil*. Both the algorithms were run for 100 simulations. For the purpose of demonstration, five different initial configurations of the K-means algorithm and five different initial populations of the GA-clustering algorithm are shown in the tables.

For *Data 1* (Tables 1 and 2) it is found that the GA-clustering algorithm provides the optimal value of 2.225498 in all the runs. K-means algorithm also attains this value most of the times (87% of the total runs). However in the other cases, it gets stuck at a value of 5.383132. For *Data 2* (Tables 3 and 4), GA-clustering attains the best value of 51.013294 in all the runs. K-means, on the other hand, attains this value in 51% of the total runs, while in other runs it gets stuck at different sub-optimal values. Similarly, for *Data 3* (Tables 5 and 6) and *Data 4* (Tables 7 and 8) the GA-clustering algorithm attains the best values of 966.312576 and 1246.218355 in 20% and 85% of the total runs, respectively. The best

Table 8
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Data 4* after 100 iterations when $K = 2$

Initial population	GA-clustering
1	1246.221381
2	1246.218355
3	1246.218355
4	1246.218355
5	1246.218355

Table 9
$\mathcal{M}$ obtained by K-means algorithm for five different initial configurations for *Vowel* when $K = 6$

Initial configuration	K-means
1	157460.164831
2	149394.803983
3	161094.118096
4	149373.097180
5	151605.600107

Table 10
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Vowel* after 100 iterations when $K = 6$

Initial population	GA-clustering
1	149346.490128
2	149406.851288
3	149346.152189
4	149355.823103
5	149362.780998

Table 11
$\mathcal{M}$ obtained by K-means algorithm for five different initial configurations for *Iris* when $K = 3$

Initial configuration	K-means
1	97.224869
2	97.204574
3	122.946353
4	124.022373
5	97.204574

values provided by the K-means algorithm for these data sets are 976.235607 (obtained in 20% of the total runs) and 1246.236680 (obtained in 25% of the total runs), respectively, Notably, even the worst values obtained by the GA-clustering algorithm are better than the best values provided by the K-means algorithm for these data sets.

Table 12
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Iris* after 100 iterations when $K = 3$

Initial population	GA-clustering
1	97.10077
2	97.10077
3	97.10077
4	97.10077
5	97.10077

Table 13
$\mathcal{M}$ obtained by K-means algorithm for five different initial configurations for *Crude Oil* when $K = 3$

Initial configuration	K-means
1	279.743216
2	279.743216
3	279.484810
4	279.597091
5	279.743216

Table 14
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Crude Oil* after 100 iterations when $K = 3$

Initial population	GA-clustering
1	278.965150
2	278.965150
3	278.965150
4	278.965150
5	278.965150

For *Vowel Data*, (Tables 9 and 10), the K-means algorithm attains the best value of 149373.097180 only once (out of 100 runs). The best value obtained by GA-clustering algorithm is 149346.152189 (which is obtained in 18% of the total runs). The best value obtained by the latter is better than that obtained by K-means algorithm. Notably, the latter attains values less than 150000 in all the runs, while the former attains values greater than this in the majority of its runs.

For *Iris* (Tables 11 and 12) and *Crude Oil* (Tables 13 and 14) data sets, the GA-clustering algorithm again attains the best values of 97.10077 and 278.965150 respectively in all the runs. The K-means algorithm, on the other hand, fails to attain this value in any of its runs. The best that K-means algorithm achieved are 97.204574

Table 15
$\mathcal{M}$ obtained by GA-clustering algorithm for five different initial populations for *Vowel* after 500 iterations when $K = 6$

Initial population	GA-clustering
1	149344.229245
2	149370.762900
3	149342.990377
4	149352.289363
5	149362.661869

(reached 60% of the times) and 279.484810 (reached 30% of the times), respectively.

From Tables 9 and 10 for *Vowel*, it is found that unlike the other cases, GA-clustering algorithm attains one value (149406.851288) that is poorer than the best value of K-means algorithm (149373.097180). In order to investigate whether the GA-clustering algorithm can improve its clustering performance, it was executed upto 500 iterations (rather than 100 iterations as was done previously). The results are shown in Table 15. As expected, it is found that the performance of GA-clustering improves. The best value that it now attains is 149342.990377 and the worst is 149370.762900, both of which are better than those obtained after 100 iterations. Moreover, now its performance in all the runs is better than the performance of K-means algorithm for any of the 100 runs.

5. Discussion and conclusions

A genetic algorithm-based clustering algorithm, called GA-clustering, has been developed in this article. Genetic algorithm has been used to search for the cluster centres which minimize the clustering metric $\mathcal{M}$. In order to demonstrate the effectiveness of the GA-clustering algorithm in providing optimal clusters, several artificial and real life data data sets with the number of dimensions ranging from two to ten and the number of clusters ranging from two to nine have been considered. The results show that the GA-clustering algorithm provides a performance that is significantly superior to that of the K-means algorithm, a very widely used clustering technique.

Floating-point representation of chromosomes has been adopted in this article, since it is conceptually closest to the problem space and provides a straight forward way of mapping from the encoded cluster centres to the actual ones. In this context, a binary representation may be implemented for the same problem, and the results may be compared with the present floating point form. Such an investigation is currently being performed.

Note that the clustering metric $\mathcal{M}$ that the GA attempts to minimize is given by the sum of the absolute Euclidean distances of each point from their respective cluster centres. We have also implemented the same algorithm by using the sum of the squared Euclidean distances as the minimizing criterion. The same conclusions as obtained in this article are still found to hold good.

It has been proved in Ref. [23] that an elitist model of GAs will definitely provide the optimal string as the number of iterations goes to infinity, provided the probability of going from any population to the one containing the optimal string is greater than zero. Note that this has been proved for nonzero mutation probability values and is independent of the probability of crossover. However, since the *rate* of convergence to the optimal string will definitely depend on these parameters, a proper choice of these values is imperative for the good performance of the algorithm. Note that the mutation operator as used in this article also allows nonzero probability of going from any string to any other string. Therefore, our GA-clustering algorithm will also provide the optimal clusters as the number of iterations goes to infinity. Such a formal theoretical proof is currently being developed that will effectively serve as a theoretical proof of the optimality of the clusters provided by the GA-clustering algorithm. However, it is imperative to once again realize that for practical purposes a proper choice of the genetic parameters, which may possibly be kept adaptive, is crucial for a good performance of the algorithm. In this context, one may note that although the K-means algorithm got stuck at sub-optimal solutions, even for the simple data sets, GA-clustering algorithm did not exhibit any such unwanted behaviour.

6. Summary

Clustering is an important unsupervised classification technique where a set of patterns, usually vectors in a multi-dimensional space, are grouped into clusters in such a way that patterns in the same cluster are similar in some sense and patterns in different clusters are dissimilar in the same sense. For this it is necessary to first define a measure of similarity which will establish a rule for assigning patterns to the domain of a particular cluster centre. One such measure of similarity may be the Euclidean distance $\mathbf{D}$ between two patterns $\mathbf{x}$ and $\mathbf{z}$ by $\mathbf{D} = \|\mathbf{x} - \mathbf{z}\|$. Smaller the distance between $\mathbf{x}$ and $\mathbf{z}$, greater is the similarity between the two and vice versa.

An intuitively simple and effective clustering technique is the well-known K-means algorithm. However, it is known that the K-means algorithm may get stuck at suboptimal solutions, depending on the choice of the initial cluster centres. In this article, we propose a solution to the clustering problem where genetic algorithms (GAs) are used for searching for the appropriate cluster

centres such that a given metric is optimized. GAs are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and having a large amount of implicit parallelism. GAs perform search in complex, large and multimodal landscapes, and provide near optimal solutions for objective or fitness function of an optimization problem. It is known that elitist model of GAs provide the optimal string as the number of iterations goes to infinity when the probability of going from any population to the one containing the optimal string is greater than zero. Therefore, under limiting conditions, a GA-based clustering technique is also expected to provide an optimal clustering with respect to the clustering metric being considered.

In order to demonstrate the effectiveness of the GA-based clustering algorithm in providing optimal clusters, several artificial and real-life data sets with the number of dimensions ranging from two to ten and the number of clusters ranging from two to nine have been considered. The results show that the GA-clustering algorithm provides a performance that is significantly superior to that of the *K*-means algorithm for these data sets.

Acknowledgements

This work was carried out when Ms. Sanghamitra Bandyopadhyay held the Dr. K. S. Krishnan fellowship awarded by the Department of Atomic Energy, Govt. of India. The authors acknowledge the reviewer whose valuable comments helped immensely in improving the quality of the article.

References

[1] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, New York, 1989.
[2] L. Davis (Ed.), Handbook of Genetic Algorithms, Van Nostrand Reinhold, New York, 1991.
[3] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, New York, 1992.
[4] J.L.R. Filho, P.C. Treleaven, C. Alippi, Genetic algorithm programming environments, IEEE Comput. 27 (1994) 28–43.
[5] S.K. Pal, D. Bhandari, Selection of optimal set of weights in a layered network using genetic algorithms, Inform. Sci. 80 (1994) 213–234.
[6] S.K. Pal, D. Bhandari, M.K. Kundu, Genetic algorithms for optimal image enhancement, Pattern Recognition Lett. 15 (1994) 261–271.
[7] D. Whitley, T. Starkweather, C. Bogart, Genetic algorithms and neural networks: optimizing connections and connectivity, Parallel Comput. 14 (1990) 347–361.
[8] R.K. Belew, J.B. Booker (Eds.), Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, 1991.
[9] S. Forrest (Ed.), Proceedings of the Fifth International Conference Genetic Algorithms, Morgan Kaufmann, San Mateo, 1993
[10] L.J. Eshelman (Ed.), Proceedings of the Sixth International Conference Genetic Algorithms, Morgan Kaufmann, San Mateo, 1995.
[11] E.S. Gelsema (Ed.), Special Issue on Genetic Algorithms, Pattern Recognition Letters, vol. 16(8), Elsevier Sciences Inc., Amsterdam, 1995.
[12] S.K. Pal, P.P. Wang (Eds.), Genetic Algorithms for Pattern Recognition, CRC Press, Boca Raton, 1996.
[13] S. Bandyopadhyay, C.A. Murthy, S.K. Pal, Pattern classification using genetic algorithms, Pattern Recognition Lett. 16 (1995) 801–808.
[14] S.K. Pal, S. Bandyopadhyay, C.A. Murthy, Genetic algorithms for generation of class boundaries, IEEE Trans. Systems, Man Cybernet. 28 (1998) 816–828.
[15] M.R. Anderberg, Cluster Analysis for Application, Academic Press, New York, 1973.
[16] J.A. Hartigan, Clustering Algorithms, Wiley, New York, 1975.
[17] P.A. Devijver, J. Kittler, Pattern Recognition: A Statistical Approach, Prentice-Hall, London, 1982.
[18] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[19] J.T. Tou, R.C. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Reading, 1974.
[20] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, 1990.
[21] R.C. Dubes, A.K. Jain, Clustering techniques: the user's dilemma, Pattern Recognition 8 (1976) 247–260.
[22] S.Z. Selim, M.A. Ismail, *K*-means type algorithms: a generalized convergence theorem and characterization of local optimality, IEEE Trans. Pattern Anal. Mach. Intell. 6 (1984) 81–87.
[23] D. Bhandari, C.A. Murthy, S.K. Pal, Genetic Algorithm with elitist model and its convergence, Int. J. Pattern Recognition Artif. Intell. 10 (1996) 731–747.
[24] A. Homaifar, S.H.Y. Lai, X. Qi, Constrained Optimization via genetic algorithms, Simulation 62 (1994) 242–254.
[25] C.Z. Janikow, Z. Michalewicz, An experimental comparison of binary and floating point representations in genetic algorithms, in: R.K. Belew, J.B. Booker (Eds.), Proceedings of the Fourth International Conference Genetic Algorithms, Morgan Kaufmann, San Mateo, 1991, pp. 31–36.
[26] W.L.G. Koontz, P.M. Narendra, K. Fukunaga, A branch and bound clustering algorithm, IEEE Trans. Comput. C-24 (1975) 908–915.
[27] J.H. Wolfe, Pattern Clustering by multivariate mixture analysis, Multivariate Behav. Res. 5 (1970) 329–350.
[28] W.L.G. Koontz, P.M. Narendra, K. Fukunaga, A graph theoretic approach to nonparametic cluster analysis, IEEE Trans. Comput. C-25 (1975) 936–944.
[29] H. Spath, Cluster Analysis Algorithms, Ellis Horwood, Chichester, UK, 1989.
[30] S.K. Pal, D.D. Majumder, Fuzzy sets and decision making approaches in vowel and speaker recognition, IEEE Trans. Systems, Man Cybernet. SMC-7 (1977) 625–629.
[31] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugenics 3 (1936) 179–188.
[32] R.A. Johnson, D.W. Wichern, Applied Multivariate Statistical Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1982.

About the Author—UJJWAL MAULIK did his Bachelors in Physics and Computer Science in 1986 and 1989, respectively. Subsequently, he did his Masters and Ph.D in Computer Science in 1991 and 1997, respectively, from Jadavpur University, India. Dr. Maulik has visited Center for Adaptive Systems Applications, Los Alamos, New Mexico, USA in 1997. He is currently the Head of the Department of Computer Science, Kalyani Engineering College, India. His research interests include Parallel Processing and Interconnection Networks, Natural Language Processing, Evolutionary Computation and Pattern Recognition.

About the Author—SANGHAMITRA BANDYOPADHYAY did her Bachelors in Physics and computer Science in 1988 and 1991, respectively. Subsequently, she did her Masters in Computer Science from Indian Institute of Technology, Kharagpur in 1993 and Ph.D in Computer Science from Indian Statistical Institute, Calcutta in 1998. Dr. Bandyopadhyay is the recipient of Dr. Shanker Dayal Sharma Gold Medal and Institute Silver Medal for being adjudged the best all round post graduate performer in 1993. She has visited Los Alamos National Laboratory in 1997. She is currently on a post doctoral assignment in University of New South Wales, Sydney, Australia. Her research interests include Evolutionary Computation, Pattern Recognition, Parallel Processing and Interconnection Networks.