ELSEVIER

Pattern Recognition Letters 18 (1997) 975–986

Pattern Recognition Letters

A clustering algorithm using an evolutionary programming-based approach

Manish Sarkar, B. Yegnanarayana *, Deepak Khemani

Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600 036, India

Received 24 May 1996; revised 21 August 1997

Abstract

In this paper, an evolutionary programming-based clustering algorithm is proposed. The algorithm effectively groups a given set of data into an optimum number of clusters. The proposed method is applicable for clustering tasks where clusters are crisp and spherical. This algorithm determines the number of clusters and the cluster centers in such a way that locally optimal solutions are avoided. The result of the algorithm does not depend critically on the choice of the initial cluster centers. © 1997 Published by Elsevier Science B.V.

Keywords: Clustering; K-means; Optimization; Evolutionary programming

1. Introduction

Clustering a set of data provides a systematic approach for partitioning the given set of data into different groups such that patterns having similar features are grouped together, and patterns with different features are placed in different groups (Dubes and Jain, 1987). Formally, clustering can be defined as follows (Bezdek, 1981): Given a set $X = \{x_1, x_2, \ldots, x_N\}$ of feature vectors, find an integer K $(2 \leq K < N)$ and the K partitions of X which exhibit categorically homogeneous subsets. In the field of clustering, the K-means algorithm (Tou and Gonzalez, 1974) is a very popular algorithm. It is used for clustering where clusters are crisp and spherical. In the K-means algorithm, clustering is based on minimization of the overall sum of the squared errors between each pattern and the corresponding cluster center. This can be written as minimization of the following objective function,

$$E = \sum_{k=1}^{K} \sum_{x \in C_k} \|x - m_k\|^2 \tag{1}$$

where K is number of clusters and m_k is the center of the kth cluster C_k. Although the K-means algorithm is extensively used in literature (Dubes and Jain, 1987; Selim and Sultan, 1991), it suffers from several drawbacks. Firstly, to apply the method, the user has to know a priori the number of clusters present in the given input data set. Secondly, the objective function is not convex, and hence, it may contain local minima. Therefore, while minimizing the objective function, there is a possibility of getting stuck in local minima (also in local maxima and saddle points). Finally, the performance of the K-means algorithm depends on the choice of the initial cluster centers.

* Corresponding author. Email: yegna@iitm.ernet.in.

0167-8655/97/$17.00 © 1997 Published by Elsevier Science B.V. All rights reserved.
PII S0167-8655(97)00122-0

In this article we propose a clustering algorithm to address the following issues:

1. How to determine the optimum number of the clusters.
2. How to avoid local minima solutions.
3. How to make clustering independent of the initial choice of the cluster centers.
4. How to cluster properly when the clusters differ in size, density and number of patterns.

Since human ability to cluster data is far superior to any of the clustering algorithms, we look at some of the features in our clustering mechanism. For example, when we see a picture, we try to cluster the elements of the picture into different groups. It is interesting to note that, immediately after observing a picture we can find how many clusters there are, and it is done without looking at each point within the clusters. Thus, it appears that clustering depends on the global view of the observer. After deciding the number of clusters, we try to see which point belongs to which cluster. So, we gather global information first, and then we look for local properties. Now the question is, what criterion do we use to gather the global information? Possibly we collect this global information from the isolation and compactness of the clusters in the whole picture. Although the K-means algorithm takes care of the local properties of the picture, it does not take the global view into account.

We propose a clustering algorithm which mimics the above mentioned features of the human way of clustering. The proposed algorithm is applicable when clusters are crisp and spherical. In this algorithm, two objective functions are minimized simultaneously. The global view of the input data set is considered by an objective function called *Davies–Bouldin index* (DB-index) (Dubes and Jain, 1987). Minimization of this objective function takes place by randomly merging and splitting the clusters. The objective function E given in Eq. (1), is minimized to take care of the local property, i.e., to determine which input pattern should belong to which cluster. It turns out that minimization of the global performance index, i.e., the DB-index, gives the optimum number of clusters, whereas minimization of E leads to proper positioning of the cluster centers. In other words, the task of minimizing the DB-index can be considered as a major one, while the task of minimizing E can be regarded as a minor one. The role played by the DB-index and E are quite similar to the role played by the generalization error and training error in configuring an artificial neural network (Pao et al., 1996). Minimization of both objective functions may yield locally optimal solutions. To circumvent the local minima problem, we propose an optimization technique based on evolutionary programming (EP) (Fogel, 1995). EP optimizes both objective functions by using a controlled stochastic search. It performs the search in parallel from more than one point. While searching for the global minimum, this technique explores simultaneously many paths. Certain search paths may be less promising in the initial stages, whereas due to random perturbation of the search parameters it may become highly promising after some time. In the EP-based approach, the less promising solutions are also kept along with the highly promising solutions, hoping that in future they would lead to new search paths towards the global minimum. These new paths help the search process to avoid locally optimal solution. Also, by splitting and merging the clusters or by small perturbation of the cluster centers, the search operation may jump over locally minimal solution. Due to these two reasons, the proposed method can avoid local minima. In the EP-based clustering approach, initially, more than one solution is generated, and the solutions are then repeatedly adapted by splitting and merging the clusters or by small perturbation of the cluster centers. Thus, the initial choice of the cluster centers is also not very critical in the proposed EP-based algorithm.

In many pattern recognition problems, including clustering, EP is considered to be a more powerful optimization tool than other existing optimization methods, namely simulated annealing (SA) (Selim and Sultan, 1991) and genetic algorithms (GA) (Goldberg, 1989). In particular, SA is a sequential search operation, whereas EP is a parallel search algorithm. In fact, we can say that EP is more than a parallel search. Parallel search starts with a number of different paths (say $P > 1$) and continues until all the search paths get stuck in blind alleys or any one of them finds the solution. EP also starts with P different paths. But, it always tries to generate new paths which are better than the current paths. Due to this inherent parallelism, an EP-based search operation is more efficient than the SA-based search operation (Porto et al., 1995). Moreover, through mathematical analysis it can be shown that under mild conditions, the probability of success in stationary Markovian optimization techniques like EP is better than nonstationary Markovian optimization processes like SA (Hart, 1996). Although EP and GA are both parallel search operations based on the principle of the stationary Markov chain, for the clustering problem an EP-based optimization approach is advantageous over a GA-based approach. One major problem with the GA-based approach is the permutation problem (Yao, 1993). The permutation problem stems from the fact that in GA two functionally identical sets of clusters which order their clusters differently have two different genotype representations. Therefore, the probability of producing a highly fit offspring from them by crossover will be very low. The EP-based optimization method, however, does not suffer from this problem. Hence, for clustering problems, EP is a better optimization tool than GA.

2. Background of evolutionary programming

2.1. Basics of evolutionary programming

Let us consider the problem of finding the global minimum of a function

$\mathcal{F}(\boldsymbol{x}):\mathbb{R}^n \to \mathbb{R}$

where $\boldsymbol{x}$ is an n-dimensional vector. EP uses the following steps to solve the problem (Fogel, 1994b, 1995):

1. Initially a population of parent vectors $\boldsymbol{x}_i$, $i = 1,2,\ldots,P$, is selected at random (uniformly) from a feasible range in each dimension.
2. An offspring vector $\hat{\boldsymbol{x}}_i$, $i = 1,2,\ldots,P$, is created from each parent $\boldsymbol{x}_i$, by adding a Gaussian random variable with zero mean and predefined standard deviation to each component of $\boldsymbol{x}_i$.
3. A selection procedure then compares the values $\mathcal{F}(\boldsymbol{x}_i)$ and $\mathcal{F}(\hat{\boldsymbol{x}}_i)$ to determine which of these vectors are to be retained. The P vectors that possess the least error become the parents for the new generation.
4. This process of generating new trials and selecting those with least error continues until a sufficient solution is reached or the number of generations becomes greater than some prespecified constant.

As an example, minimization of the following quadratic function by EP is described in (Fogel, 1994b),

$$\mathcal{F}(\boldsymbol{x}) = \sum_{l=1}^{n} x_l^2, \qquad (2)$$

which shows that solution of the above function quickly converges to the global minimum.

2.2. Evolutionary programming in clustering

In this section, we use the above ideas to cluster a given set of data. The objective is to find the optimum number of clusters and optimum position of each cluster center. Formally, this can be treated as a problem of finding the global minimum of the following function,

$$\mathcal{F}(\boldsymbol{\xi}):\mathbb{R}^{nK} \to \mathbb{R}, \qquad (3)$$

where $\boldsymbol{\xi}$ is an nK-dimensional vector representing $[\boldsymbol{m}_1,\boldsymbol{m}_2,\ldots,\boldsymbol{m}_K]$ and $\mathcal{F}(\boldsymbol{\xi})$ signifies how good the clustering is. In the above relation, $\boldsymbol{\xi}$ is like $\boldsymbol{x}$ in Eq. (2). However, there is a fundamental difference. Unlike $\boldsymbol{x}$, here the dimension of $\boldsymbol{\xi}$ is variable. So, EP should be able to find the optimum value of $\boldsymbol{\xi}$ as well as the optimum value of K.

Now we describe how the above idea can be used in a practical situation. To cluster an input data set, initially EP needs to create a population of sets of clusters. Hence, EP initializes the population with sets of clusters having randomly generated (uniform distribution) cluster centers. Thus, P such sets of cluster centers are formed, each set having any number of cluster centers between two and some prespecified positive integer. The number of cluster centers in each set is determined randomly. These sets are called *parents*. The entire data set is then clustered based on the set of parent cluster centers by the *modified K-means* (MKM) algorithm, which is described in the next section. A *fitness* value for each parent set is measured. Each parent is allowed to create one offspring. Thus, P offspring sets of cluster centers are generated. The method of creating the offsprings is described in the next section. After creating these offsprings, the entire data set is clustered by the MKM based on the set of offspring

1. Randomly generate a population of sets of cluster centers (call them parents).
2. Cluster each set using the MKM.
3. Find the fitness value of each parent set.
4. Create the offspring of each parent set.
5. Cluster each offspring set using the MKM.
6. Find the fitness value of each offspring set.
7. Competition starts among all parent and offspring sets based on the fitness value.
8. Survival of the fittest sets (call them parents).
9. If number of generations is less than some prespecified constant then go to step 4.

Fig. 1. The proposed evolutionary programming-based clustering algorithm.

cluster centers, and then the fitness value of each offspring set is measured. As a result, we obtain $2P$ sets of clusters comprising of parents as well as offsprings. Now the competition phase starts. In this phase, the fitness values of all sets (parents as well as offspring) are compared. For each solution, the algorithm chooses 10 randomly selected opponents from all parents and offsprings with uniform probability. In each comparison, if the conditioned set offers as good performance as the randomly selected opponent, it receives a *win* (Porto et al., 1995; Saravanan and Fogel, 1995). Based on the wins, sets scoring in the top 50% are designated as parents. All other sets are discarded. Again these parents are used to create offsprings. The whole procedure is continued until the number of generations becomes larger than some prespecified constant. We can formalize the above idea in the form of an algorithm as shown in Fig. 1. Finally, the set with maximum fitness value is considered as the desired output.

3. Implementation issues in evolutionary programming-based clustering

3.1. Fitness function

The fitness function of a set of clusters is given by

$$\text{fitness value} = \frac{1}{\text{Davies–Bouldin index}}, \tag{4}$$

where the Davies–Bouldin index is determined as follows (Dubes and Jain, 1987):

Given a partition of the N input data into K clusters, one first defines the following measure of within-to-between cluster spread for two clusters, C_j and C_k for $1 \leq j,k \leq K$ and $j \neq k$.

$$R_{j,k} = \frac{e_j + e_k}{D_{jk}}, \tag{5}$$

where e_j and e_k are the average dispersion of C_j and C_k, and $D_{j,k}$ is the Euclidean distance between C_j and C_k. If $\boldsymbol{m}_j$ and $\boldsymbol{m}_k$ are the centers of C_j and C_k, then

$$e_j = \frac{1}{N_j} \sum_{x \in C_j} \|\boldsymbol{x} - \boldsymbol{m}_j\|^2$$

and $D_{jk} = \|\boldsymbol{m}_j - \boldsymbol{m}_k\|^2$, where $\boldsymbol{m}_j$ is the center of cluster C_j consisting of N_j patterns. We define a term R_k for C_k as

$$R_k = \max_{j \neq k} R_{j,k}. \tag{6}$$

Now, the DB-index for K-clustering is defined as

$$\text{DB}(K) = 1/K \sum_{k=1}^{K} R_k. \tag{7}$$

It is to be noted that we have two objective functions, E in Eq. (1) and the DB-index in Eq. (7), to minimize. Of these two, we are treating only the inverse of the DB-index as the fitness function (given in Eq. (4)). The reason is that the evaluation of E in Eq. (1) requires K to be predefined and fixed. Hence, when K varies, the value of E for a set with optimal number of clusters may not attain the minimum value. For example, if the number of clusters of a set is very close to the number of data, then the value of E is close to zero. Obviously, this kind of situation may not signify optimal clustering. However, instead of minimizing both objective functions, we could have minimized only the DB-index. But, our search for a better set of clusters becomes more efficient when minimization of E is viewed as a clue to minimize the DB-index. In other words, use of DB-index and E are meant for *exploration* and *exploitation* in the search space, respectively (Renders and Flasse, 1996).

3.2. Generation of offsprings

To generate offsprings, the following three steps are needed:

3.2.1. Replicate the parent

In the first step each parent is represented by the number of clusters and cluster centers. In this step, these values are copied from the parent to generate a new offspring.

3.2.2. Mutation

The aim of creating offsprings is to minimize E and the DB-index. Basically, the creation of an offspring is nothing but searching one step forward or backward in the search space. But, the length of a step size and the step direction are unknown. The step size cannot be too big or too small, because it may cause the search process to jump over a global minimum or it may take lot of time to reach the global minimum. Therefore, it is necessary to determine the stepsize and step direction of the search method probabilistically. The nondeterminism associated with the step size and step direction selection further helps to avoid the local minima in the search process. To minimize E, we come across local minima which we call parametric local minima, and to minimize the DB-index we come across local minima which we call structural local minima. Parametric local minima and structural local minima are reduced by the parametric mutation and structural mutation, respectively. Using parametric mutation, each cluster center $\boldsymbol{m}_k$, $1 \le k \le K$, is perturbed with Gaussian noise. This can be expressed as

$$\boldsymbol{m}_k = \boldsymbol{m}_k + N(0,T) \tag{8}$$

Specifically, the mutation step size $N(0,T)$ is a Gaussian random vector with each component having mean 0 and variance T.

The intensity of parametric mutation however needs to be controlled, i.e., it is to be high when the fitness value of the parent is low and vice versa. This can be accomplished if we consider T of a particular set of clusters as its temperature, and define it as

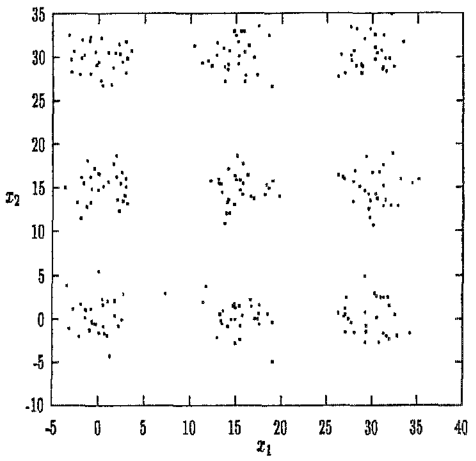
$$T = \alpha U(0,1)\left[\frac{\text{minimum fitness}}{\text{fitness of the set of clusters}}\right], \tag{9}$$

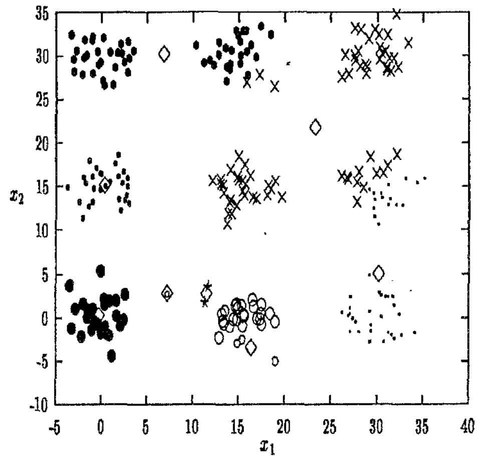
where $U(0,1)$ is a uniform random variable over the interval $[0,1]$, α is a constant ($\alpha \le 1$). Obviously, from the definition the range of T lies in between 0 and 1. This temperature determines how close the set is to being a solution for the task (Angeline et al., 1994), and the amount of parametric mutation is controlled depending on T. Like simulated annealing, this temperature is used to anneal the mutation parameters. Initially when the temperature is high, mutation parameters are annealed quickly with coarse grains. At low temperatures, they are annealed slowly with fine grains. Large mutations are indeed needed to escape a parametric local minimum during search. But, many times it adversely affects the offspring's ability to perform better than its parent (Angeline et al., 1994). So, to lessen the frequency of large parametric mutation we have multiplied the right-hand side of Eq. (9) by $\alpha U(0,1)$. The value of the minimum fitness used in Eq. (9) is determined as given in the Appendix.

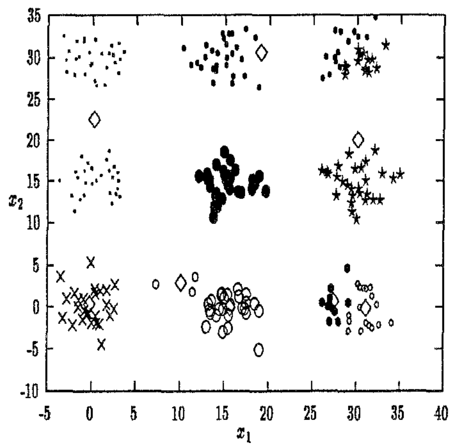
Structural mutation is used to obtain the optimum number of clusters to avoid structural local minima. The determination of the optimum number of clusters can be considered as a search problem in a structure space where each point represents a particular set of clusters. If a performance index like the DB-index is assigned to each set of clusters, the performance level of all possible sets of clusters forms a surface in the structure space. Thus, determination of the optimum number of clusters is equivalent to finding the lowest point on this surface. However, this searching operation becomes complicated as the surface has the following characteristics (Yao, 1993; Miller et al., 1989):

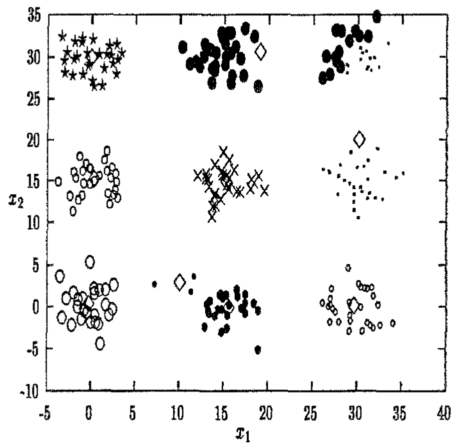
1. The surface is very large since the number of possible sets of clusters can be very high.
2. The surface is nondifferentiable as the change in the number of clusters is discrete.

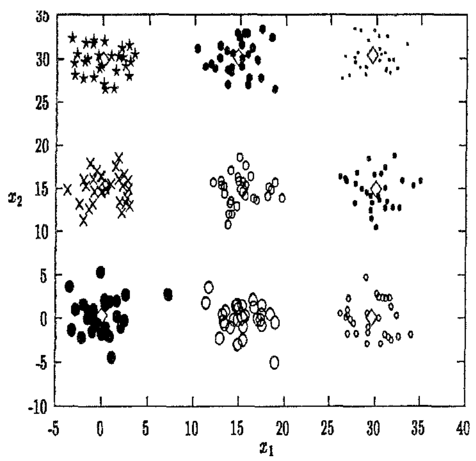
In order to find the proper number of clusters, i.e., to find the global minimum in the structure space, sometimes one cluster is added to or deleted from an offspring (Tou and Gonzalez, 1974), and these addition and deletion operations are controlled by structural mutation. The addition of one cluster into an offspring set is done by splitting an existing cluster of that set. To identify a cluster for splitting, it is required to find the cluster with maximum hypervolume, where hypervolume V_k of cluster C_k is defined

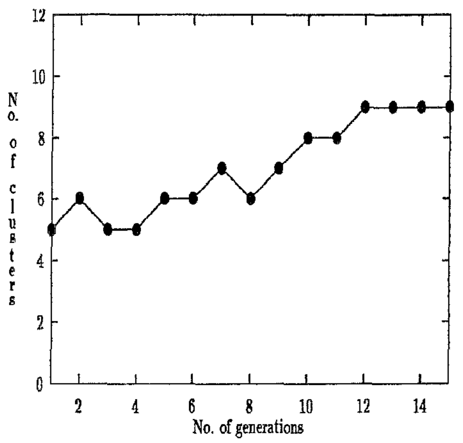

x_2
x_1
-5 0 5 10 15 20 25 30 35 40
-10 -5 0 5 10 15 20 25 30 35
(a)


x_2
x_1
-5 0 5 10 15 20 25 30 35 40
-10 -5 0 5 10 15 20 25 30 35
(b)


x_2
x_1
-5 0 5 10 15 20 25 30 35 40
-10 -5 0 5 10 15 20 25 30 35
(c)


x_2
x_1
-5 0 5 10 15 20 25 30 35 40
-10 -5 0 5 10 15 20 25 30 35
(d)


x_2
x_1
-5 0 5 10 15 20 25 30 35 40
-10 -5 0 5 10 15 20 25 30 35
(e)


No. of clusters
No. of generations
2 4 6 8 10 12 14
0 2 4 6 8 10 12
(f)

as (Gath and Geva, 1989)

$$V_k = \sqrt{\det\left(\frac{1}{N_k}\sum_{x\in C_k}(\boldsymbol{x}-\boldsymbol{m}_k)(\boldsymbol{x}-\boldsymbol{m}_k)'\right)}. \tag{10}$$

Let C_k be the cluster with maximum hypervolume V_k. In order to break this cluster into two parts, the center of this cluster, i.e., $\boldsymbol{m}_k$, is split into two new cluster centers $\boldsymbol{m}_k^+$ and $\boldsymbol{m}_k^-$, and then $\boldsymbol{m}_k$ is deleted (Tou and Gonzalez, 1974). As a result, the number of clusters of this set, i.e., K is incremented by one. Here, the cluster center $\boldsymbol{m}_k^+$ is formed by adding a certain quantity γ_k to the component of $\boldsymbol{m}_k$ which corresponds to the maximum component of $\boldsymbol{\sigma}_k$ (variance of the kth cluster), i.e., $\sigma_{k_{\max}}$; and in a similar way $\boldsymbol{m}_k^-$ is formed by subtracting γ_k from the same component of $\boldsymbol{m}_k$. One simple way of specifying γ_k is to make it equal to some fraction of $\sigma_{k_{\max}}$, that is

$$\gamma_k = \kappa\sigma_{k_{\max}}, \quad \text{where } 0 < \kappa \le 1. \tag{11}$$

Deletion of one cluster from an offspring set is executed by merging two existing clusters of that set. In order to accomplish it, in the set of clusters, the two closest clusters with centers $\boldsymbol{m}_{k_1}$ and $\boldsymbol{m}_{k_2}$ are identified for merging. Thereafter, these two clusters are merged by a lumping operation as

$$\boldsymbol{m}_k^* = \frac{1}{N_{k_1}+N_{k_2}}\left[N_{k_1}\boldsymbol{m}_{k_1} + N_{k_2}\boldsymbol{m}_{k_2}\right],$$

where $\boldsymbol{m}_k^*$ is the center of the new cluster. Next, $\boldsymbol{m}_{k_1}$ and $\boldsymbol{m}_{k_2}$ are deleted, and the number of clusters K is reduced by one.

It is important to note that the splitting and merging operations employed in the proposed scheme are quite similar to the splitting and merging operations found in ISODATA (Tou and Gonzalez, 1974). However, unlike in ISODATA, here cluster merging and splitting are executed in a nondeterministic fashion. This inherent nondeterministic property plays the key role in avoiding local minima while finding the optimum number of clusters, and eventually it guarantees the asymptotic convergence of the EP-based clustering scheme towards the global minimum (Fogel, 1994a).

The specific instants of cluster addition or deletion depend upon the amount of structural mutation, which further depends upon the value of the probability of structural mutation (p_{m}). A large value of p_{m} transfers EP into a purely random search algorithm, on the other hand some mutation is indeed needed to prevent the premature convergence of EP to a suboptimal solution (Srinivas and Patnaik, 1994). So, the value of p_{m} of each solution is adaptively changed in response to the fitness ($\mathscr{F}$) of that particular solution. Let the average fitness value of the population and maximum fitness value of the population be denoted by $\overline{\mathscr{F}}$ and $\mathscr{F}_{\max}$, respectively. Here we borrow a result from (Srinivas and Patnaik, 1994), which says that $(\mathscr{F}_{\max} - \overline{\mathscr{F}})$ is likely to be less for a population that has converged to an optimal solution than that for a population scattered in the solution space (Srinivas and Patnaik, 1994). The value of p_{m} is increased when the population tends to get stuck in a local minimum, and is decreased when the population is scattered in the solution space. Hence, p_{m} of the above average offsprings (i.e., the offsprings with $\mathscr{F} \ge \overline{\mathscr{F}}$) should be inversely proportional to $\mathscr{F}_{\max} - \overline{\mathscr{F}}$. In addition, p_{m} of the above average offsprings is made proportional to $\mathscr{F}_{\max} - \mathscr{F}$. This is done to achieve low values of p_{m} for highly fit offsprings so that these offsprings are preserved. Therefore,

$$p_{\mathrm{m}} = k_{\mathrm{m}}(\mathscr{F}_{\max} - \mathscr{F})/(\mathscr{F}_{\max} - \overline{\mathscr{F}}) \quad \text{if } \mathscr{F} \ge \overline{\mathscr{F}}. \tag{12}$$

The value of the proportionality constant k_{m} is taken as 0.5, such that p_{m} varies linearly from 0 (for the best set) to 0.5 (for the average set). Since below average offsprings should undergo a large amount of mutation, p_{m} for them is assumed to be equal to k_{m}. Hence,

$$p_{\mathrm{m}} = k_{\mathrm{m}} \quad \text{if } \mathscr{F} < \overline{\mathscr{F}}. \tag{13}$$

Fig. 2. (a) Input data set. (b), (c) and (d) are clustered outputs of K-means algorithm with different initializations. (e) Clustered output of the proposed algorithm. (f) Number of clusters plotted versus number of generation.

3.2.3. Modified K-Means algorithm (MKM)

By exploiting the mutation efficiently we obtain the perturbed cluster centers and number of clusters for a particular offspring. However, to calculate the fitness value of this offspring, the input data set needs to be clustered based on these perturbed cluster centers. In addition, if the perturbed cluster centers are updated based on the clustered output, then the minimization of E takes place, and as a result, minimization of the DB-index becomes easy. We exploit the modified K-means (MKM) algorithm to accomplish this task. In each offspring, the MKM is executed for one iteration at each generation. Consequently, if an offspring survives g generations, then it passes through g iterations, and thus, the MKM is indeed iterative in nature. The steps associated with the MKM algorithm are specified below:

1. If the current generation is the first generation, follow this step else skip it. For each parent set, randomly generate the number of clusters, K where $K > 1$, and randomly determine the cluster centers within the range of the input set.
2. Distribute the N input data among the present cluster centers, using the relation

$$x \in C_k \text{ if } \|x - m_k\| \le \|x - m_j\| \quad \forall x,$$
$$k = 1, \ldots, K, \text{and } j = 1, \ldots, K, j \ne k. \qquad (14)$$

   Resolve ties arbitrarily. Moreover, if some cluster remains empty after the above grouping is over, it is eliminated.
3. Update each cluster center m_k, by

$$m_k = \frac{1}{N_k + 1}\left(\sum_{x \in C_k} x + m_k\right).$$

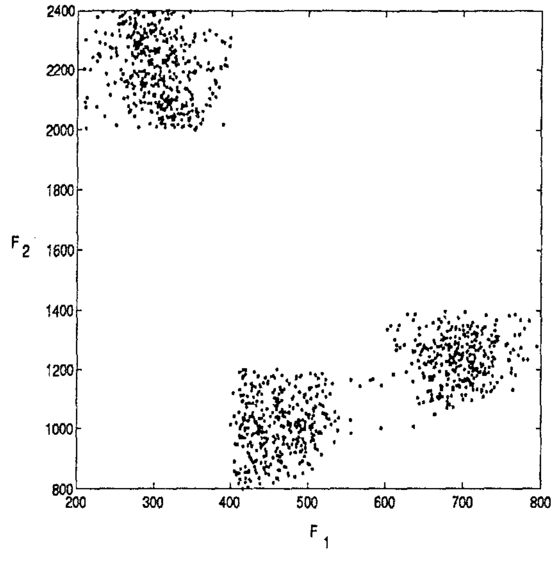
The MKM algorithm basically remembers the cluster center at the last generation, and updates the old cluster center in the current generation. This updating process, however, may get stuck in certain parametric local minima. In order to avoid this, the cluster centers of the offspring obtained from the last generation are perturbed by applying Eq. (8), and then used in the current generation for further updating. Although both MKM and K-means are iterative in nature, the difference between them is that the K-means never uses the old cluster centers in perturbed form. This difference makes the K-means a deterministic search operation, and thus vulnerable for parametric local minima.

4. Results and discussion

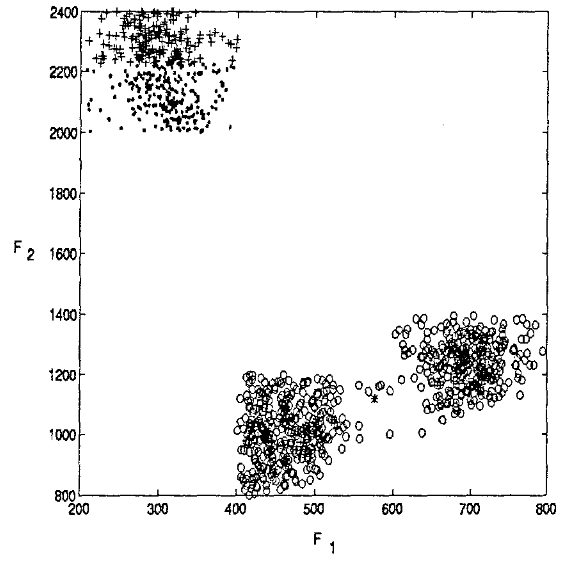
For our first experiment, we generated 350 two-dimensional data from nine Gaussian distributions (Fig. 2(a)). Even after assigning the number of clusters in the K-means algorithm as nine, it failed to cluster this data set properly (Fig. 2(b)). One possible reason may be poor initialization, which in the presence of local minima may force the search process to get stuck in one of these local minima. To avoid this initialization problem, we executed the K-means algorithm with two other random initializations (Fig. 2, (c) and (d)). The K-means algorithm performed poorly with these initializations also.

Fig. 2(e) depicts the clustered data using the proposed algorithm on the same data set. The proposed algorithm found the optimum number of clusters after twelve generations. In this method, the values of P and κ were taken as 4 and 0.6, respectively. We observed that for α equal to one, the proposed algorithm worked quite well. During structural mutation only one cluster was added or deleted at a time. Fig. 2(f) illustrates the evolution of clusters in the best set against the number of generations. Moreover, this figure exhibits the self-organization capability of the proposed algorithm, due to which it is able to cluster correctly, even though it started with the wrong number of clusters and the incorrect position of the cluster centers. This figure also shows that sets with different structural variations evolve throughout the whole process. In fact, it exhibits that search for a better set of clusters (structurally) is carried out all through the process. After the proposed algorithm converges on this data set, the value of E is calculated from Eq. (1). It is found to be 7.2% less than the value of E obtained after the K-means converges on this same data set. It again demonstrates the usefulness of the proposed method to avoid parametric local minima.

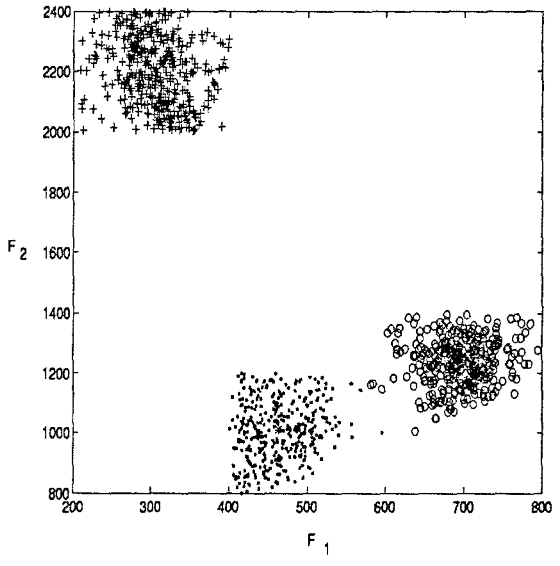
We performed a second experiment on a set of 350 English vowel sounds corresponding to the classes "a", "i" and "o". They were uttered in a Consonant-Vowel-Consonant context by three 25-35 year-old male speakers. The data set has three fea-



(a)



(b)



(c)

Fig. 3. (a) Input speech data corresponding to the classes "a", "i" and "o". Along the X-axis formant F_1 and along the Y-axis formant F_2 are shown. (b) Clustered output by K-means. Three different clusters are represented by ".", "o" and "+". (c) Clustered output by the proposed algorithm. Classes corresponding to "a", "i" and "o" are represented by the clusters consisting of symbols "o", "+" and ".", respectively. The cluster centers are represented by "$*$".

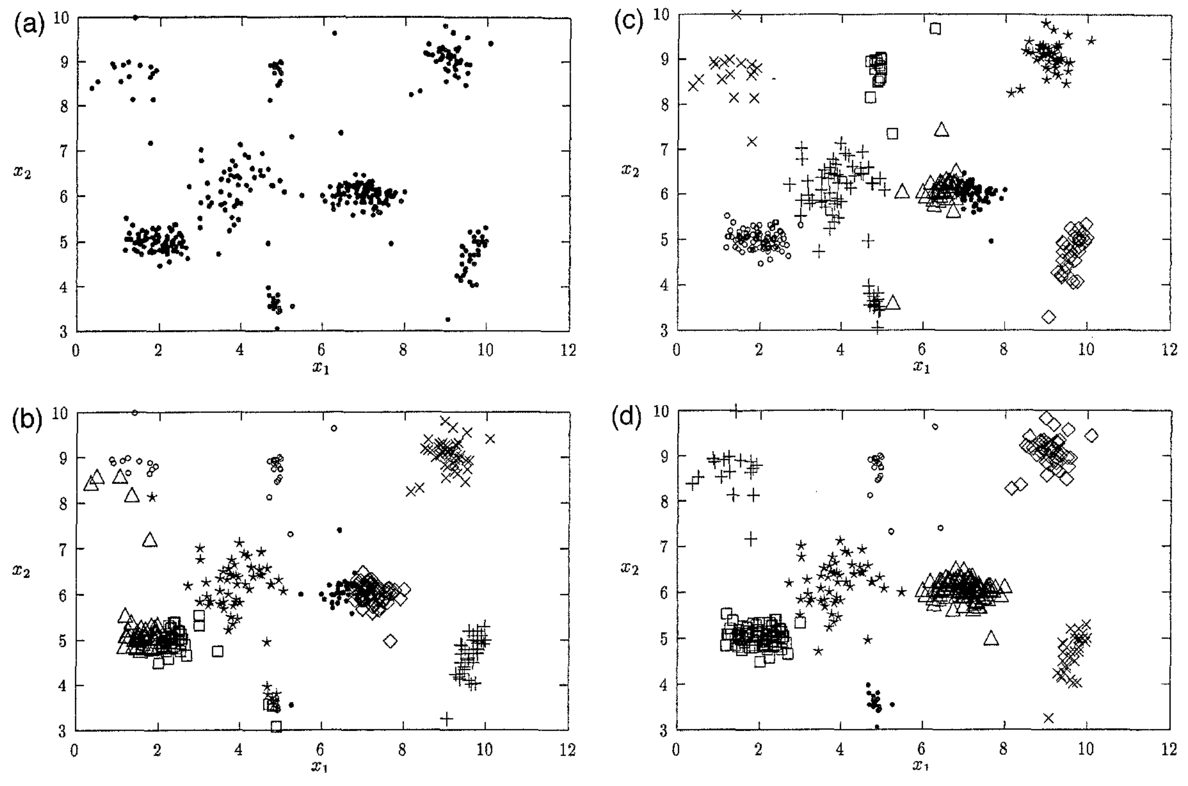


Fig. 4. (a) Eight different Gaussian distributions are used to artificially generate a set of data. (b) and (c) Clustered outputs by fuzzy K-means with two different sets of random initial cluster centers. (d) Clustered output by the proposed clustering algorithm. The proposed algorithm automatically finds the proper number of clusters and cluster centers.

tures F_1, F_2 and F_3 corresponding to first, second, and third vowel formant frequencies (obtained through spectrum analysis of the speech data). For ease of visualization, we illustrate the clustering performance on a two-dimensional plane formed by the formants F_1 and F_2. The given data set is depicted in Fig. 3(a). From Fig. 3(b), we can see that the K-means failed to cluster the input data set, even though it knew in advance the number of clusters present in the data set. In contrast, the proposed algorithm took five generations (Fig. 3(c)) to cluster the input data set after finding the proper number of clusters. The classification rate of the clustered output of the proposed algorithm, when compared to the original data, is 99.5%.

In the last experiment we generated 252 two-dimensional data from eight different Gaussian distributions (Fig. 4(a)). We applied the fuzzy K-means clustering algorithm (FKM) (Bezdek, 1981) on this data set. The number of clusters was indicated as eight a priori in the algorithm. Fig. 4(b) and Fig. 4(c) show clustered outputs of the FKM with two different random initializations. From these figures, it can be observed that the FKM has failed to cluster the data set. The reason may be that the FKM search procedure got stuck in some local minima due to incorrect initialization. Fig. 4(d) shows the clustered output after applying the proposed algorithm on the same set of data. The proposed algorithm self-organizes to find the proper number of clusters and proper cluster centers automatically. The clustered output is evidently far better than the FKM's outputs. Moreover, unlike the FKM, the proposed algorithm does not need to know the number of clusters in the data set a priori. However, it should be noted that the FKM may perform better when clustering is more fuzzy, as the proposed algorithm is not designed to take fuzzy classification into account.

5. Conclusion

It is important to note that like the K-means algorithm, the proposed clustering method depends on the Euclidean distance between input data and the cluster centers. Hence, this clustering approach can be applied only when the common property of the data of a cluster can be described by Euclidean distances between the input data and the cluster center. No attempt has been made to test the efficiency of the proposed method with other fitness functions as the purpose of this article is to illustrate the approach. The advantage here is that one can do other modifications in the given framework. In the future, this kind of technique is proposed to be extended to take care of fuzzy clustering.

Acknowledgements

This work was carried out while Manish Sarkar held a Dr K.S. Krishnan Research Fellowship from the Department of Atomic Energy, Government of India. The authors would like to thank Dr David B. Fogel, Natural Selection Inc., USA and anonymous referees for their helpful comments on earlier drafts of the paper.

Appendix A. Minimum fitness value

Suppose that there are K clusters $(K \geq 2)$ in the input data set of size N. Of the K clusters, we are considering any two clusters C_1 and C_2 consisting of N_1 and N_2 patterns $(N_1, N_2 < N)$, respectively. Let the centers of the clusters be $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$, and the radii be r_1 and r_2, respectively. These two clusters are such that $\text{dist}(\boldsymbol{m}_1, \boldsymbol{m}_2) = D_{12} = r_1 + r_2$, i.e., both clusters are touching each other. From Eq. (5), we can say,

$$R_{1,2} = \frac{e_1 + e_2}{D_{12}} = \frac{e_1 + e_2}{r_1 + r_2}.$$

Here, r_1 can be approximately taken as average dispersion of C_1 per pattern that belongs to C_1, i.e.,

$$r_1 = \sqrt{\frac{1}{N_1} \sum_{x \in C_1} \|(\boldsymbol{x} - \boldsymbol{c}_1)\|^2} = \frac{e_1}{\sqrt{N_1}}.$$

Similarly, r_2 can be defined. Therefore,

$$R_{1,2} = \frac{\sqrt{N_1}\, r_1 + \sqrt{N_2}\, r_2}{r_1 + r_2} = \sqrt{N_1} - \frac{\left(\sqrt{N_1} - \sqrt{N_2}\right) r_2}{r_1 + r_2} \leq \sqrt{N_1}, \quad \text{when } N_1 \geq N_2. \tag{A.1}$$

Since, $N_1 < N$, we can write, $R_{1,2} < \sqrt{N}$. The numerator of the right-hand side of Eq. (5) does not vary with position of the clusters, and for any position of the clusters the denominator of the right-hand side of Eq. (5) is larger than $r_1 + r_2$. Hence, the maximum value of $R_{1,2}$ is $\sqrt{N}$. Without loss of generality, we can say that, $\sqrt{N}$ is the maximum value of $R_{1,k}$ for $k = 1,2,\ldots,K$, i.e., $R_{1,k} < \sqrt{N}$. Hence, from Eq. (6), $R_1 < \sqrt{N}$. Obviously, this relation is true for all R_k when $k = 1,2,\ldots,K$. So,

$$\text{DB}(K) = 1/K \sum_{k=1}^{K} R_k < 1/K \sum_{k=1}^{K} \sqrt{N},$$

i.e., $\text{DB}(K) < \sqrt{N}$. Therefore, the minimum fitness value is $1/\sqrt{N}$.

References

Angeline, P.J., Saunders, G.M., Pollack, J.B., 1994. An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans. Neural Networks 5 (1), 54–64.

Bezdek, J.C., 1981. Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York.

Dubes, R., Jain, A., 1987. Algorithms that Cluster Data. Prentice-Hall, Englewood Cliffs, NJ.

Fogel, D.B., 1994a. Asymptotic convergence properties of genetic algorithms and evolutionary programming. Cybernetics and Systems 25, 389–407.

Fogel, D.B., 1994b. An introduction to simulated evolutionary optimization. IEEE Trans. Neural Networks 5 (1), 3–14.

Fogel, D.B., 1995. Evolutionary Computation: Toward a New Philosophy of Machine Learning. IEEE Press, Piscataway.

Gath, I., Geva, A.B., 1989. Unsupervised optimal fuzzy clustering. IEEE Trans. Pattern Anal. Machine Intell. 11 (7), 773–781.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Hart, W.E., 1996. A theoretical comparison of evolutionary algorithms and simulated algorithm. In: Proc. 5th Ann. Conf. on Evolutionary Programming, San Diego, pp. 147–154.

Miller, G.F., Todd, P.M., Hegde, S.U., 1989. Designing neural networks and genetic algorithms. In: Schaffer, J.D. (Ed.), Proc. 3rd Internat. Conf. on Genetic Algorithms and their Applications. Morgan Kaufmann, San Mateo, CA.

Pao, Y.H., Igelnik, B., LeClair, S.R., 1996. An approach for neural-net computing with two-objective functions. In: Proc. IEEE Internat. Conf. on Neural Networks, Washington, DC, pp. 181–186.

Porto, W., Fogel, D.B., Fogel, L.J., 1995. Alternative neural network training methods. IEEE Expert, 16–22.

Renders, J.M., Flasse, S.P., 1996. Hybrid methods using genetic algorithms for global optimization. IEEE Trans. System Man Cybernet. 26 (2), 243–258.

Saravanan, N., Fogel, D.B., 1995. Evolving neural control systems. IEEE Expert, 23–27.

Selim, S.Z., Sultan, K.A., 1991. A simulated annealing algorithm for the clustering problem. Pattern Recognition 24 (10), 1003–1008.

Srinivas, M., Patnaik, L.M., 1994. Adaptive probabilities of crossover and mutation in genetic algorithm. IEEE Trans. System Man Cybernet. 24 (4), 656–667.

Tou, J., Gonzalez, R., 1974. Pattern Recognition Principles. Addison-Wesley, Reading, MA.

Yao, X., 1993. Evolutionary artificial neural networks. Internat. J. Neural Networks 4 (3), 203–222.