Application of Genetic Programming for Multicategory Pattern Classification

J. K. Kishore, L. M. Patnaik, *Fellow, IEEE*, V. Mani, and V. K. Agrawal

***Abstract*—This paper explores the feasibility of applying genetic programming (GP) to multicategory pattern classification problem for the first time. GP can discover relationships among observed data and express them mathematically. Multicategory pattern classification has been done traditionally by using the maximum likelihood classifier (MLC). GP-based techniques have an advantage over statistical methods because they are distribution free, i.e., no prior knowledge is needed about the statistical distribution of the data. GP also has the ability to automatically discover the discriminant features for a class. GP has been applied for two-category (class) pattern classification. In this paper, a methodology for GP-based n-class pattern classification is developed. The given n-class problem is modeled as n two-class problems, and a genetic programming classifier expression (GPCE) is evolved as a discriminant function for each class. The GPCE is trained to recognize samples belonging to its own class and reject samples belonging to other classes. A strength of association (SA) measure is computed for each GPCE to indicate the degree to which it can recognize samples belonging to its own class. The higher the value of SA, the better is the ability of a GPCE to recognize samples belonging to its own class and reject samples belonging to other classes. The SA measure is used for uniquely assigning a class to an input feature vector. Heuristic rules are used to prevent a GPCE with a higher SA from swamping a GPCE with a lower SA. Experimental results are presented to demonstrate the applicability of GP for multicategory pattern classification, and the results obtained are found to be satisfactory, and are compared with those of the MLC. We also discuss the various issues that arise in our approach to GP-based classification, such as the creation of training sets, the role of incremental learning, and the choice of function set in the evolution of GPCEs, as well as conflict resolution for uniquely assigning a class.**

***Index Terms*—Evolutionary computation, genetic programming, pattern classification.**

I. Introduction

CLASSIFICATION plays a major role in any pattern recognition problem. In a typical n-class pattern classification problem, a large number of representative samples are available for each of the n classes. In multicategory pattern classification, an input feature vector of m dimensions is classified as belonging to one of the n classes. Classification has been done traditionally by the maximum likelihood classifier (MLC). Although MLC can be used with any likelihood function, a normal distribution is generally assumed for the input data because it leads to minimum classification error [1]. Artificial neural networks have also been applied successfully for n-class pattern classification problems in the areas of remote sensing [2] and biomedical applications [3].

Both the MLC and neural networks have the framework for a multicategory pattern classification problem. In the MLC approach, after the mean vectors and covariance matrices have been computed, the likelihood values are computed for each class for a given input vector. The given input vector is assigned to the class with maximum likelihood value [1]. The basic drawback of the maximum likelihood classification is that a distance-based approach for classification is adopted and a normal distribution is assumed for the input data. In the neural-network approach, a multilayered neural network with m inputs and n outputs is trained with a training set. Subsequently, the given input vector is applied to the network, and an n-dimensional output vector (result vector) is obtained. The given input vector is assigned the class of the maximum output [4]. In the neural-network approach, the basic drawback is that the optimal configuration of the network is not known *a priori*. Moreover, training times can be quite large, and the knowledge that is represented internally in the network weights is often opaque [4]. In a multicategory pattern classification problem, apart from assigning a class to a given input feature vector, there is a need to discover the underlying relationship among data and express it in an understandable manner.

Genetic programming (GP) is gaining attention due to its ability to discover the underlying data relationships and express them mathematically. Although GP uses the same principles as genetic algorithms (GAs) [5], it is a symbolic approach to program induction, i.e., it involves the discovery of a highly fit computer program from the space of computer programs that produces a desired output when presented with a particular input. Let

$F = f_1, f_2, \ldots, f_n$ be the set of functions

$T = X_1, X_2, \ldots, X_n$ be the set of terminals.

The functions in the function set may include

- arithmetic operations $(+, -, \times, \div)$
- mathematical functions (such as SINE, COS, EXP, LOG);
- Boolean operators (such as AND, OR, NOT);
- conditional operators [such as IF LESS THAN OR EQUAL TO (IFLTE)];
- user-defined domain-specific functions.

Manuscript received May 12, 1998; revised March 22, 1999 and September 26, 1999.

J. K. Kishore is with the Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560012, India and the ISRO Satellite Center, Bangalore 560017, India.

L. M. Patnaik is with the Microprocessor Application Laboratory, Indian Institute of Science, Bangalore 560012, India.

V. Mani is with the Department of Aerospace Engineering, Indian Institute of Science, Bangalore 560012, India.

V. K. Agrawal is with the ISRO Satellite Center, Bangalore 560017, India.

Publisher Item Identifier S 1089-778X(00)04465-9.

The set of possible structures, i.e., computer programs in GP, is the set of all possible compositions of functions that can be composed from F and T. GP begins with a population of randomly created computer programs. Each computer program represents a potential solution. GP maintains a population of solutions to a given problem. During every generation, the fitness of each solution is evaluated, and for the next generation, the solutions are selected based on their fitness. The choice of functions, terminals, and the fitness function depend upon the problem. The population evolves over a number of generations through the application of variation operators, such as crossover and mutation. The results of the two operators are placed in a new population. GP has been successfully applied to diverse problems such as optimal control, planning in artificial intelligence, discovery of game-playing strategies [6], evolution of neural networks [7], fuzzy logic production rules [8], automated synthesis of analog electrical circuits [9], and in decision support for vehicle dispatching [10].

The major considerations in applying GP to n-category pattern classification are listed below.

1) GP-based techniques are data distribution free, i.e., no *a priori* knowledge is needed about the statistical distribution of the data or no assumption is made as in MLC.
2) GP can operate directly on the data in their original form.
3) GP can detect the underlying but unknown relationship that exists among data, and express it as a mathematical LISP S expression. The generated LISP S expressions can be used directly in the application environment.
4) GP can discover the most important discriminative features of a class.

GP has been applied for a two-class pattern classification problem [6]. In a two-class problem, a single GP expression is evolved. While evaluating the GP expression, if the result of this GP expression is ≥ 0, the input data are assigned to one class; else they are assigned to the other class. Thus, the desired output d is $+1$ in the training set for one class and is -1 for the other class. Hence, the output of a GP expression is either $+1$ (indicating that the input data belong to that class) or -1 (indicating that the input sample does not belong to that class). In this paper, a methodology for applying GP to an n-category pattern classification problem is presented. When the GP paradigm is extended from a two-class problem to the n-class problem, the following questions arise.

As a typical GP expression returns a value ($+1$ or -1) for a two-class problem, how does one apply GP for the n-class pattern classification problem?

What should be the fitness function during evolution of the GP expressions?

How does the choice of a function set affect the performance of GP-based classification?

How should training sets be created for evaluating fitness during the evolution of GP expressions?

How does one improve learning of the underlying data distributions in a GP framework?

How should conflict resolution be handled before assigning a class to the input feature vector?

How does GP compare with that of the widely used MLC for an n-class pattern classification problem?

This paper addresses these questions, and demonstrates the applicability of the GP paradigm for the n-category pattern classification problem. This paper is organized as follows. Section II addresses the various questions that arise while extending GP to an n-class pattern classification problem. Section III presents various statistical measures used to validate a classifier. Section IV presents the experimental results obtained. Section V discusses the important issues in our approach to GP-based pattern classification. The conclusions are summarized in Section VI.

II. GP-Based Classification

This section addresses the questions that arise while extending GP from a two-class to an n-class pattern classification problem.

A. Formulation of the n-Class Problem as n Two-class Problems

In an earlier study [6], GP was applied to a two-class pattern classification problem. In a two-class problem, simple thresholding is sufficient for a discriminant function to divide the feature space into two regions. This means that one GPCE expression is sufficient to say whether or not the given input feature vector belongs to that class; i.e., the GP expression returns a value ($+1$ or -1). To extend GP to an n-class problem, we modify the n-class problem as n two-class problems. For the sake of illustration, consider a five-class pattern classification problem. Let n_j be the number of samples that belong to class j, and let N_j be the number of samples that do not belong to class $j\,(j = 1, \cdots, 5)$. Thus

$$\begin{aligned} N_1 &= n_2 + n_3 + n_4 + n_5 \\ N_2 &= n_1 + n_3 + n_4 + n_5 \\ N_3 &= n_1 + n_2 + n_4 + n_5 \\ N_4 &= n_1 + n_2 + n_3 + n_5 \\ N_5 &= n_1 + n_2 + n_3 + n_4. \end{aligned}$$

When the five-class problem is formulated as five two-class problems, we need five GPCEs as discriminant functions to resolve between n_1 and N_1, n_2 and N_2, n_3 and N_3, n_4 and N_4, and lastly, n_5 and N_5. Thus, each of these five two-class problems is handled as a separate two-class problem with simple thresholding. GP uses a function set that contains operators and functions to evolve a GPCE as the discriminant function for the given pair of classes present in the training set. Let O_i be the output of GPCE$_i$. Then

$$\begin{aligned} &\text{If } (\text{GPCE}_i(\boldsymbol{X}) \geq 0)\ O_i = +1\ \boldsymbol{X} \in \text{Class}_i \\ &\text{If } (\text{GPCE}_i(\boldsymbol{X}) \ngeq 0)\ O_i = -1\ \boldsymbol{X} \notin \text{Class}_i \end{aligned} \tag{1}$$

where $\boldsymbol{X}$ is the input feature vector. Each GPCE partitions the feature space differently into two regions. Thus, for an n-class problem, n GPCEs are evolved. In the following sections, we

will explain why conflicts arise in class assignment, and how conflict resolution is handled for uniquely assigning a class to a given input.

B. Fitness Measure

GP is guided by the fitness function to search for the most efficient computer program to solve a given problem. A simple measure of fitness has been adopted for the pattern classification problem

$$\text{fitness} = \frac{\text{number of samples classified correctly}}{\text{number of samples used for training during evolution}}. \quad (2)$$

C. Choice of Function Set

As GP uses a function set (FS) for evolving a GPCE, four combinations of the function set (FS) are considered here. They are

- arithmetic (A) given by $+, -, \times, \div$;
- arithmetic and logical (AL) given by $+, -, \times, \div, IFLTE$;
- arithmetic and nonlinear (ANL) given by $+, -, \times, \div, SINE$;
- arithmetic, logical, and nonlinear ($ALNL$) given by $+, -, \times, \div, IFLTE, SINE.$;

We will show in the next section how the choice of the function set affects the performance of GP for the n-class pattern classification problem.

D. Creation of Training Sets

In a two-class problem, the pattern classification is between two classes only, and so both the classes typically have an equal number of samples, and only one GPCE is needed. As the n-class problem has been converted into n two-class problems, n GPCEs are evolved, and so n GPCE specific training sets are needed. In each GPCE specific training set, the number of samples belonging to one class (whose desired output d is $+1$) is outnumbered by the samples belonging to all other classes (whose desired output is -1). For example, in a five-class problem, let the number of samples belonging to each class be 100. Thus, in our formulation, $n_1 = 100$ and $N_1 = 400$. So in the training set for class 1, the desired output d will be $+1$ for 100 samples and d will be -1 for 400 samples. Although it is still a valid training set for a two-class problem, it results in a highly skewed training set as there are more representative samples for one category than for the other. Our experimental results show that this skewness leads to misclassification of input feature vectors. To overcome the skewness, one possible option is to use a subset of the data belonging to other classes (whose desired output is -1), so that the number of samples belonging to a class will be the same as the number of samples belonging to other classes. Although a balanced training set is created in this manner, it will lead to poor learning as the data for the other classes are not representative.

The training set should be as representative as possible for proper learning to take place to discover the underlying data relationships. So, we are proposing an interleaved data format for the training set that is used to evaluate the fitness function during evolution.

E. Interleaved Data Format

In the interleaved data format, the samples belonging to the true class are alternately placed between samples belonging to other classes, i.e., they are repeated. Table I illustrates the format of the training set for class 1 in a five-class problem [$n_1(+1)$ is repeated]. The desired output d of the GPCE is $+1$ for the samples of the true class, and is -1 for the other classes, and is shown in parentheses.

The number of samples in the training set is increased, and hence the time taken for evaluating fitness also increases. The training time is proportional to the size of the training set. Hence, the training time for evolving the GPCE of class 1 with the skewed data set is proportional to $n_1 + N_1$, and for the interleaved data format, it is proportional to $(n-1)n_1 + N_1$. In the next section, we will show the improvement in GPCEs performance due to this interleaved data format.

F. Incremental Learning

After the training set is created, it is used for driving the fitness function during evolution. Conventionally, all of the samples of a training set are fed to every member of the population to evaluate its fitness in each generation. We call this "global" learning as GP tries to learn the entire training set at every stage of the evolution.

So here we propose incremental learning for the GP paradigm. A subset of the training set is fed, and the size of the subset is increased gradually over time to cover the entire training data. For example, if there are 1000 samples in the training set, we feed a subset (say 50 samples) first, and then increases gradually this over time (in steps of 50) to cover the entire training data. The basic motivation for incremental learning is to improve learning during evolution as it is easier to learn a smaller task, and then to progress from a smaller task to a bigger task.

The fitness of the population is computed on the subset rather than on the entire training set. The subset can be a certain fraction (e.g., 5% of the training set), and can be user defined. Let

- N_t be the size of a GPCE specific training set;
- n_s be the increment for the subset (e.g., 5% of the training set);
- n_c be the number of samples classified correctly;
- n_g be the number of generations before the size of the training set is increased.

Learning is performed in a loop where the outer loop controls the number of increments needed to cover the entire training set. For example, if the size of the increment is 5%, then the outer loop has 20 iterations. In each loop, the population is evolved for n_g generations before the subset is augmented, and n_s determines the number of times j that the subset should be augmented to cover the entire training set. The fitness function can be written as

$$\text{fitness} = \frac{n_c}{n_s * j}, \qquad j = 1, 2, \ldots, (N_t/n_s). \quad (3)$$

For example, if $N_t = 1000$, $n_g = 200$, and a 5% increment is chosen, n_s is 50 samples initially. After every $n_g = 200$ generations, the size of the training set is augmented in terms of 50 samples until the entire training set is covered. After the training set is covered, it becomes global learning.

G. Conflict Resolution

Evolution is performed on the training set for each class. The fittest individual becomes the GPCE for that class. These GPCEs are then applied to the validation set for classification. For every input in the validation set, all of the GPCEs are applied. For every input, each GPCE returns either $+1$ (to indicate that the input belongs to its class) or -1 (to indicate that the input does not belong to its class). Thus, for each input, we get an n-dimensional result vector containing $+1$ or -1 as its entries. Note that, in this method of applying GP to an n-class multicategory pattern classification problem, the following three situations can arise.

1) Only one GPCE returns a value of $+1$, and the others return a value of -1. So, the input sample can be assigned uniquely to a class.
2) In the presence of overlapping data distributions, it is possible for more than one GPCE to return a value of $+1$ as its output. For example, GPCE 1 and GPCE 3 can return a value of $+1$ as their output for a given input. This means that the input sample is classified as belonging to class 1 by GPCE 1 and belonging to class 3 by GPCE 3. There is a need for conflict resolution when more than one GPCE returns a value of $+1$ as its output.
3) In the worst case, it is possible for all GPCEs to return a value of -1. This means that no GPCE is able to identify the sample as belonging to its own class. Such samples will be assigned to the reject class. In the subsequent sections, we will discuss a likely cause for such a situation.

There is a need for conflict resolution in an n-class problem unlike in a two-class problem. The conflict resolution has to assign a unique class to the input feature vector when more than one GPCE claims that the input belongs to its class. For this purpose, a strength of association (SA) measure is computed for each GPCE. The SA indicates the degree to which a GPCE can recognize samples belonging to its own class, and reject samples belonging to other classes. The higher the value of SA, the better will be the GPCE in recognizing samples belonging to its own class.

The SA measures are computed after the evolution of the n GPCEs. The training set (the GPCE specific training sets are derived from this set) is used to generate a class count matrix from which the SA measures are derived. The steps involved in obtaining the class count matrix and strength of association are as follows. The class count matrix (Table II) is of size $n \times n$, and is obtained from the training set where n is the number of classes. For obtaining the class count matrix, all of the GPCEs are applied on all of the samples in the training set, and the results are noted. Algorithm 1 gives the procedure for obtaining the class count matrix, where C_{ij} stands for the number of samples of class i for which j returns $+1$.

TABLE I
INTERLEAVED DATA FORMAT FOR TRAINING SET OF GPCE 1 IN A FIVE-CLASS PROBLEM

class#1
n_1 (+1)
n_2 (-1)
n_1 (+1)
n_3 (-1)
n_1 (+1)
n_4 (-1)
n_1 (+1)
n_5 (-1)

TABLE II
CLASS COUNT MATRIX

GPCE Class	1	2	3	-	n
1	C_{11}	C_{12}	-	-	C_{1n}
2	C_{12}	C_{22}	-	-	C_{2n}
3	-	-	-	-	-
-	-	-	-	-	-
n	C_{1n}	C_{2n}	-	-	C_{nn}

Algorithm 1: Procedure for Determining Class Count Matrix

```
Begin
  for i = 1 to n
    for j = 1 to n
      C_ij = 0;
  for i = 1 to n/*n classes
    for k = 1 to n_i
      for j = 1 to n/*n GPCEs
        if (GPCE_j[X_k] ≥ 0, then C_ij = C_ij + 1
End
```

Ideally, the class count matrix is a diagonal matrix, where each diagonal entry is equal to the number of samples present in the training set for that particular class. However, due to overlapping data distributions, GPCE$_i$ can return a value of $+1$ for samples belonging to other classes, which leads to off-diagonal elements in the class count matrix.

The strength of association is computed from the class count matrix. Let S_i be the sum of elements present in row i of the class count matrix. Algorithm 2 gives the procedure for determining the strength of association (SA$_i$) for GPCE$_i$.

Algorithm 2: Strength of Association

```
Begin
  for i = 1 to n
    { S_i = 0
    for k = 1 to n
      S_i = S_i + C_ik
    }
  for i = 1 to n
    SA_i = C_ii/S_i
End
```

The strength of association alone may not be enough for classification. It may happen that a given input feature vector is identified by its own GPCE (output $+1$), but another GPCE with a greater strength of association can also identify this sample as its own (output is $+1$). Such a situation leads to misclassification because the GPCE with a higher strength of association swamps the GPCE with a lower strength. Heuristic rules are introduced to prevent this. The heuristic rules represent an empirical means to reduce misclassification in the GP paradigm. In the next section, through experimental results, we will show one way of identifying the heuristic rules, and how these rules improve the performance of GP-based classification.

H. Automatic Discovery of Discriminant Features

As both MLC and neural networks handle the n-class pattern classification problem directly, they use all of the features to discriminate among classes. If m-dimensional feature vectors are available and only p features ($p < m$) are actually needed for discriminating a particular class, it is difficult to identify these p features manually. There is no way in which the MLC can discover these p features. It must use all of the m features for classification. In an m-input and n-output neural network, a particular feature may be important for one class, and not so for another class. So the weight associated with that input feature tries to learn the data distributions for both of the classes, and may not go to zero. Our approach to GP-based classification offers a means to discover these p features as the n-category problem is handled as n two-class problems, and a GPCE is evolved independently for each class. GP has the capability to automatically discover the underlying data relationships among these p features, and discard the remaining features during evolution. This will be illustrated by our experimental results in Section IV.

III. Statistical Measures for Validating a Classifier

After the n GPCEs are obtained for each class from the training set, the validation set is used to analyze the performance of the GP classifier. For this purpose, the GPCEs are applied to the validation set to obtain the classification matrix (CM) which is of size $n \times n$, where n is the number of classes. A typical entry q_{ij} in the classification matrix shows how many samples belonging to class i have been classified as class j. For a perfect classifier, the classification matrix is diagonal. However, in practice, due to misclassification, we get off-diagonal elements. However, in GP, as it is possible for a result vector in GP to have all entries as -1, an additional column is needed for the reject class, and this makes the dimension of the CM $n \times (n+1)$. Algorithm 3 gives the procedure for obtaining the classification matrix.

Algorithm 3: Procedure for Determining Classification Matrix

```
Begin
  for i = 1 to n/* number of classes n
    for j = 1 to n + 1
      q_ij = 0;
  for k = 1 to N_v/* size of validation set
  {
   Apply the n GPCEs on sample k, and
     obtain the result vector.
   Perform conflict resolution.
   q_ip = q_ip + 1
  }
     i is the true class of the sample
       and p is the assigned class.
     p = n + 1 for the reject class
End
```

The classification matrix is used to obtain statistical measures for both the class level as well as the global performance of a classifier. Class-level performance is indicated by percentage classification and a polarization measure. Percentage classification tells us how many samples belonging to a particular class have been classified correctly. The polarization measure looks at the total number of samples that were assigned to a particular class, and indicates the fraction of the samples that were classified correctly. These can be obtained from the classification matrix as follows:

$$\text{percentage classification (for class } i) = q_{ii}/n_i$$
$$\text{polarization measure} = q_{ii}/\sum q_{ij}$$

where n_i is the number of samples belonging to class i in the validation set.

The global indexes for a classifier [13] are the average accuracy and overall accuracy, which are defined as follows:

$$\text{average accuracy} = \sum (q_{ii}/n_i)/n$$
$$\text{overall accuracy} = \sum q_{ii}/N_v$$

where N_v is the size of the validation set, and n is the number of classes.

For an ideal classifier, the percentage classification and polarization measure are 1.0 for each class. The average accuracy and overall accuracy are also 1.0. It is important to note the subtle difference between the class count matrix and classification matrix. The training set is used to derive the class count matrix, and the strength of association for each GPCE is computed from the class count matrix. The validation set is used to derive the classification matrix and the performance of the classifier is obtained from the classification matrix. Also, the sum of the elements of a row in the class count matrix need not be equal to the number of samples belonging to that class in the training set. In the classification matrix, the sum of the elements of a row is equal to the number of samples belonging to that class in the validation set.

IV. Experimental Results for GP-Based Classification

This section presents the results of GP for an n-class pattern classification problem by considering a data set containing five classes taken from three bands of remotely sensed satellite data. The three bands are the three feature vectors $F1$, $F2$, and $F3$. Table III shows the spatial spread for the features, along with the number of samples used for training the GPCEs (training set

TS) and for their subsequent validation (validation set VS). This data set is chosen to show the applicability of GP-based classification for a real-world problem. In Table III, there are 169 samples belonging to class 1 in the training set, to evolve GPCE 1, and 168 samples in the validation set for validating the GPCE 1. Subsequently, we will consider the well-known Fisher's iris data set [12]. We have used GPQUICK [11] software to simulate the GP paradigm for the n-category pattern classification problem. Table IV shows the choice of parameters for the GPQUICK software used in the experiments. The choice of these parameters is, however, based on empirical observation. Appendix A defines the parameters used in Table IV. In our experiments, we have used the maximum number of generations (5000) or 90% classification accuracy as the termination criterion.

TABLE III
CHARACTERISTICS OF THE DATA SET USED FOR GP-BASED CLASSIFICATION

	F1	F2	F3		
Class	Min Max	Min Max	Min Max	TS	VS
Class#1	23 35	25 45	20 63	169	168
Class#2	24 46	23 64	50 72	144	145
Class#3	22 50	21 73	22 81	214	213
Class#4	23 35	21 53	55 81	215	216
Class#5	26 34	31 55	30 86	125	124

TABLE IV
GPQUICK PARAMETERS

Parameter	Weightage
Crossover weightage	0.28
Mutation weightage	0.08
Crossover weightage annealing	0.20
Mutation weightage annealing	0.40
Copy weightage	0.04
Mutation rate(P_m)	0.1
Crossover rate(P_c)	0.7
Mutation node	0.435
Mutation constant	0.435
Mutation shrink	0.13
Selection strategy	Tournament
Tournament size	7
Termination criterion	5000 generations or 90% classification

A. Skewness in the Training Data Sets

The first set of experiments was conducted to study the effect of skewness on the training data set on the evolution of GPCEs. In the training set for GPCE 1, the number of samples belonging to class 1 is 169 (i.e., the desired output d is $+1$), and the number of samples belonging to other classes is 698 (i.e., d is -1). This results in a skewed training set.

In this interleaved data format, the 169 samples belonging to class 1 are repeated so that there are 676 samples belonging to class 1 and 698 samples belonging to other classes, thus avoiding the skewness. The GPCE for class 1 is obtained using this interleaved data format. Similarly, GPCEs are obtained for all other classes. In this manner, we can see that the training time with an interleaved data format for class 1 is increased by a factor of 1.58 over the training time with the skewed data format. Note that, in Table III, while evolving GPCE for class 2, the number of samples (training set) belonging to class 2 is 144, and when the interleaved data format is used for learning, the total number of samples belonging to class 2 is 576, and the number of samples belonging to all other classes is 723. Still, there is a slight skewness in the training set. This can be overcome by repeating the training samples belonging to class 2 once more. Thus, the general idea of this interleaved data format is to properly balance the number of samples belonging to one class with the number of samples belonging to the other classes.

B. Evolution for Different Function Sets

To study the effect of function sets on GP-based classification, we have used average accuracy as the performance measure. The function sets considered were arithmetic (*A*), arithmetic and logical (*AL*), arithmetic and nonlinear (*ANL*), and arithmetic, logical, and nonlinear (*ALNL*).

The evolution of the GPCEs was done with incremental and global learning, with percentage increment (PI) indicating the fraction of samples used for augmenting the training set after every $n_g = 200$ generations during the evolution of a GPCE. Table V shows the average accuracy for a skewed data set for various combinations of the function sets (FS) and percentage increment (PI). When PI is 100, we have global learning. For each combination of PI and FS, the GPCEs were evolved for all classes. The class count matrix was determined, and the SA measures were derived. The GPCEs were applied on the validation set to obtain the classification matrix and the average accuracy. For example, in Table V, with PI being 10% for increasing the size of the training set and the function set being arithmetic and logical (*AL*), the average accuracy of classification obtained is 0.33, and is obtained as follows.

Step 1) Obtain GPCEs for all classes from the training set (skewed) using the incremental learning procedure.
Step 2) Obtain the class count matrix, and derive the strength of association measures (using Algorithms 1 and 2).
Step 3) Apply the GPCEs on the validation set, and obtain the classification matrix (using Algorithm 3).
Step 4) Compute the average accuracy as explained in the earlier section.

The same study was conducted with the interleaved data format for the training set. The average accuracy obtained for various combinations of function sets and percentage increment is shown in Table VI. A typical entry in Table VI is obtained in the same fashion, except that in Step 1), the GPCEs for all classes are evolved with the interleaved data format of the training set.

We have conducted a number of trials to study the effect of the function set and role of incremental learning on the evolution of GPCEs. Table VII gives the experimental results of 30 trials with the arithmetic function set for different values of PI for incremental learning. Based on our trials and from the results of Tables V–VII, we conclude the following.

1) Incremental learning leads to better classification (average accuracy) than global learning. Also, the smaller the percentage increment, the better is the classification.

TABLE V
AVERAGE ACCURACY FOR VARIOUS COMBINATIONS OF FS AND PI WITH SKEWED DATA SET

PI	5	10	20	25	35	50	100
A	0.51	0.38	0.51	0.33	0.36	0.39	0.40
AL	0.22	0.33	0.33	0.34	0.20	0.34	0.48
ANL	0.40	0.33	0.39	0.20	0.20	0.37	0.22
ALNL	0.20	0.36	0.20	0.20	0.20	0.33	0.20

TABLE VI
AVERAGE ACCURACY FOR VARIOUS COMBINATIONS OF FS AND PI WITH INTERLEAVED DATA FORMAT

PI	5	10	20	25	35	50	100
A	0.75	0.57	0.54	0.54	0.48	0.47	0.53
AL	0.27	0.47	0.50	0.20	0.22	0.42	0.20
ANL	0.20	0.22	0.20	0.22	0.22	0.38	0.38
ALNL	0.22	0.22	0.20	0.38	0.20	0.20	0.35

2) With respect to the choice of function set, the average accuracy with the arithmetic function set performs better than all other function sets.
3) There is variation in the classification accuracy in GP-based classification for every trial.

Having observed the overall performance of the GP classifier, we will now see how the individual GPCEs have performed (i.e., classwise performance) in one trial. Table VIII shows the classwise behavior of the individual GPCEs for the skewed and interleaved data sets with 5% incremental learning and different combinations of the function set. The performance measures for classwise performance study are the percentage classification (PC) and polarization measure (PM) defined earlier. A perfect classifier must have both the percentage classification and polarization measure as 1.0 in every class, i.e., all samples belonging to class i must be classified as i, and no samples of other classes must be classified as i. If a classifier has a high percentage classification and low polarization measure for a class i, it means that samples belonging to other classes are also classified as class i.

For example, from Table VIII, in the skewed data set case, with the function set being arithmetic and logical (AL), the percentage classification is 1.0 for class 2 i.e., all samples belonging to class 2 were classified properly. On the other hand, the polarization measure is 0.17 $(143/866 = 0.17)$, i.e., all of the samples belonging to other classes were also classified as class 2. Thus, the entire validation set is classified as class 2. So the ratio of the number of samples that were classified correctly as class 2 to the number of samples that were classified as class 2 is very small compared to the ideal polarization measure of 1.0. The reason for this low polarization measure is due to the choice of the function set. In both the skewed and interleaved training sets, the presence of logical and nonlinear functions in the function sets can lead to polarization among the classes, i.e., one or two classes tend to dominate over other classes. The skewness in the data set can also lead to polarization. On the other hand, for the interleaved data format, when the arithmetic function set is chosen, we observe that there is high polarization for 5% incremental learning (Table VIII). In Table VIII, there are many

TABLE VII
EXPERIMENTAL RESULTS OF CLASSIFICATION ACCURACY FOR 30 TRIALS (T) WITH ARITHMETIC FUNCTION SET FOR DIFFERENT VALUES OF PI

PI	5	10	20	25	35	50	100
T1	0.75	0.57	0.54	0.47	0.48	0.47	0.53
T2	0.64	0.46	0.38	0.37	0.39	0.33	0.25
T3	0.60	0.45	0.43	0.37	0.39	0.32	0.29
T4	0.56	0.50	0.34	0.36	0.40	0.39	0.32
T5	0.64	0.42	0.39	0.30	0.37	0.34	0.33
T6	0.55	0.48	0.39	0.37	0.42	0.34	0.35
T7	0.58	0.48	0.35	0.38	0.33	0.36	0.33
T8	0.52	0.47	0.35	0.36	0.39	0.34	0.25
T9	0.57	0.46	0.34	0.30	0.36	0.35	0.32
T10	0.71	0.46	0.39	0.43	0.38	0.33	0.36
T11	0.63	0.46	0.33	0.32	0.45	0.50	0.38
T12	0.68	0.40	0.34	0.24	0.34	0.33	0.39
T13	0.59	0.42	0.34	0.33	0.30	0.32	0.33
T14	0.51	0.48	0.33	0.35	0.31	0.33	0.28
T15	0.52	0.48	0.34	0.25	0.33	0.32	0.30
T16	0.67	0.49	0.34	0.39	0.33	0.39	0.37
T17	0.64	0.40	0.39	0.47	0.35	0.33	0.35
T18	0.55	0.47	0.38	0.36	0.34	0.28	0.36
T19	0.54	0.54	0.53	0.41	0.34	0.30	0.26
T20	0.54	0.41	0.40	0.40	0.37	0.38	0.39
T21	0.57	0.44	0.39	0.34	0.32	0.32	0.30
T22	0.53	0.45	0.40	0.38	0.39	0.25	0.38
T23	0.66	0.42	0.45	0.39	0.22	0.36	0.39
T24	0.51	0.42	0.38	0.32	0.46	0.25	0.31
T25	0.59	0.49	0.33	0.39	0.39	0.40	0.37
T26	0.55	0.50	0.41	0.40	0.49	0.20	0.34
T27	0.64	0.50	0.51	0.42	0.34	0.37	0.28
T28	0.54	0.42	0.41	0.34	0.37	0.33	0.27
T29	0.60	0.51	0.32	0.38	0.39	0.37	0.31
T30	0.51	0.43	0.37	0.44	0.52	0.31	0.33

classes for which both the percentage classification and the polarization measure are zero. These classes have been swamped by the dominant classes. From Table VIII, it is clear that the arithmetic function set with an interleaved data format and incremental learning has given the best classwise performance.

Consider the GPCE expressions obtained for class 1 for the four combinations of the function set (interleaved data format and 5% incremental learning) which are given below.

Arithmetic (A): (DIV(MUL(DIV(DIV $F3$ − 121)(ADD $F2-34$) (MUL(ADD $F2$(DIV -37 (ADD (DIV $F2$ $F2$)(SUB -35 $F1$))))(ADD 71 $F3$)))(SUB(ADD 109 $F2$)(SUB 87 (SUB -51 14)))).

Arithmetic and Logical (AL): (SUB(SUB(IFLTE 127 (MUL (DIV (ADD $F1$ 15)26)(ADD $F1$ $F2$))20 85)(ADD 75101)) (SUB(SUB -59 $F2$)(DIV $F3-40$))).

Arithmetic and NonLinear (ANL): (DIV (DIV (SUB 63 $F3$)$F1$)(SUB (SINE (SINE (DIV (ADD $F2$ (SINE -109)) -30))) (SUB (MUL (SINE $F1$)(DIV $F3$ 25))(DIV $F3$ $F2$)))).

Arithmetic, Logical, and NonLinear ($ALNL$): (MUL (IFLTE (DIV (MUL $F3$ $F3$)(SINE (SINE $F1$)))(DIV -73 -113) (SUB $F1$ $F2$) (MUL 44 82))(SINE (SINE 56))).

In a two-class problem, the GPCE is like a hypersurface that divides the entire feature space into two regions. When arithmetic operators are used, the GPCE can track the variation in

TABLE VIII
EXPERIMENTAL RESULTS FOR 5% INCREMENTAL LEARNING WITH DIFFERENT FUNCTION SETS FOR BOTH SKEWED AND INTERLEAVED DATA SETS

	GPCE	#1	#2	#3	#4	#5	#1	#2	#3	#4	#5
A	PC	0.91	0.86	0.76	0.0	0.0	0.92	0.85	0.65	0.98	0.37
	PM	0.36	0.52	0.82	0.0	0.0	0.82	0.73	0.95	0.69	0.96
AL	PC	0.0	1.0	0.08	0.0	0.0	0.0	0.97	0.40	0.0	0.0
	PM	0.0	0.17	1.00	0.0	0.0	0.0	0.21	0.49	0.0	0.0
ANL	PC	1.0	0.0	0.99	0.0	0.0	0.0	1.0	0.0	0.0	0.0
	PM	0.30	0.0	0.67	0.0	0.0	0.0	0.17	0.0	0.0	0.0
ALNL	PC	0.0	0.0	0.0	0.0	1.0	0.0	0.92	0.08	0.0	0.0
	PM	0.0	0.0	0.0	0.0	0.14	0.0	0.17	1.00	0.0	0.0

the input feature vectors. When a logical element is used, one of the subexpressions in the LISP *S* expression can return the same value for different inputs. For example, in the GPCE for class 1 with an arithmetic and logical function set (*AL*), consider the subexpression

(IFLTE 127 (MUL (DIV (ADD $F1$ 15) 26)(ADD $F1$ $F2$)) 20 85).

The IFLTE $(X1, X2, X3, X4)$ function evaluates its arguments as follows:

$$\text{If } X1 \geq X2, \text{ then return } X3; \text{ else, return } X4.$$

For example, this subexpression will return the value 20 for both of the input vectors (33, 43, 63) and (32, 41, 57). Hence, in the GPCE expression for class 1, the value obtained is -1. For the same input feature vectors, the GPCE evolved with only the arithmetic function set returns the value $+1$. Hence, GPCEs which have a logical element will not be able to track variation in the input due to such subexpressions, and lead to poor classification.

Similarly, in the GPCEs that were evolved with the function set containing a nonlinear function like the SINE function, if the SINE function appears in the subexpression followed by a MUL or DIV operator, it is possible for the sign to remain the same, although the value of the GPCE changes due to variation in the input. This leads to poor classification.

Hence, we observe that GPCEs evolved with an arithmetic (A) function set are able to track the variation in the input data, and thus can lead to higher classification accuracy than GPCEs evolved with other function sets (*AL*, *ANL*, *ALNL*). So GPCEs can be obtained for an n-class pattern classification problem by using the arithmetic function set, interleaved data format, and incremental learning.

C. Analysis of GPCEs

Consider the GPCE expressions obtained for the five-class problem discussed earlier by using the only arithmetic function set, interleaved data format, and 5% incremental learning. The evolved GPCEs that resulted in an average accuracy of 0.75 are as follows.

GPCE 1: (DIV(MUL(DIV(DIV $F3-121$)(ADD $F2-34$) (MUL(ADD $F2$(DIV -37 (ADD (DIV $F2$ $F2$)(SUB -35 $F1$))))(ADD 71 $F3$)))(SUB(ADD 109 $F2$)(SUB 87 (SUB -51 14)))).

The equivalent mathematical expression is given by

$$\frac{\dfrac{F3}{-121(F2-34)}(71+F3)\left\{F2+\dfrac{37}{(34+F1)}\right\}}{F2-43}. \quad (4)$$

From the above expression, we observe that GPCE 1 returns a value $+1$ only when $F2$ *is greater than 34 and less than 43* irrespective of the values of $F1$ and $F3$. Thus, GP has found that $F2$ is the most discriminating feature for class 1.

GPCE 2: (MUL (SUB (MUL $F2$ $-$ 49)(DIV (MUL 80 6)(MUL $F1$ 102))) (MUL (ADD (MUL -93 $F2$)(MUL $F1$ 96))(MUL (DIV $F2$ $F2$)(SUB 36 $F2$)))).

The equivalent mathematical expression is given by

$$(93F2-96F1)\left(49F2+\frac{480}{102F1}\right)(36-F2). \quad (5)$$

From this, we observe that GPCE 2 returns a value $+1$ when $F2 > 1.031F1$ and $F2 < 36$. Thus, GP discovers that $F1$ and $F2$ are discriminant features of class 2.

GPCE 3: (DIV 105 (SUB (ADD -62 $F3$)(DIV $F3$ -66))).

The equivalent mathematical expression is given by

$$\frac{105}{1.015F3-62}. \quad (6)$$

GPCE 3 returns a value $+1$ *only when* $F3$ *is greater than 61*, irrespective of the values of $F1$ and $F2$. Here, GP discovers that $F3$ is the discriminant feature for class 3.

GPCE 4: (SUB (DIV (MUL (DIV (SUB $F2-69$) 69)(SUB $F2$ (DIV -67 8))) (SUB (ADD 35 -72)$F1$))(DIV (MUL (DIV (ADD 60 -121)(ADD $F1$ 53))(DIV (SUB $F1$ $F3$)(ADD -56 $F3$))) -2)).

The equivalent mathematical expression is given by

$$\frac{30.5(F3-F1)}{(F1+53)(F3-56)}-\left(\frac{F2+69}{69}\right)\left(F2+\frac{67}{8}\right)\cdot\left(\frac{1}{F1+37}\right). \quad (7)$$

GPCE 5: (ADD (ADD (DIV (ADD -56 $F3$)(DIV (SUB (ADD (SUB $F2$ $F3$)(DIV $F2$ (DIV $F2$ (SUB $F2$ 33)))) -24)-26))(ADD (SUB $F2$ $F3$) (DIV $F2$ (SUB $F2$ 31))))(DIV (ADD 102 (ADD (SUB (MUL $F2$ 60)(DIV $F3$ $F3$))(SUB (ADD -6 $F3$)(MUL $F1$ 97))))$F3$)).

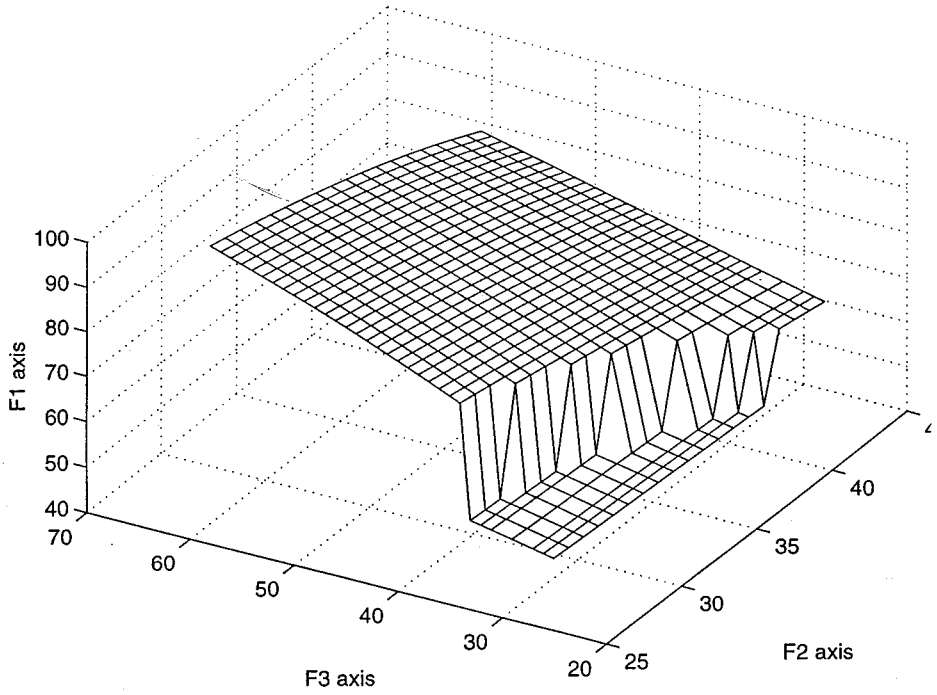


Fig. 1. Surface plot of GPCE 4 (*AZ, EL*) $= (-60, 45)$.

The equivalent mathematical expression is given by

$$\frac{26(F3-56)}{F3+9-2F2}+\frac{F2}{F2-31}+\frac{60F2+F3-97F1+95}{F3}+F2-F3. \tag{8}$$

We see that GPCE 4 and GPCE 5 are complex, and hence it was not possible to simplify them like the other GPCEs. So, we represent them pictorially.

D. Pictorial Representation of GPCEs

Fig. 1 shows the surface plot of GPCE 4, whereas Fig. 2 shows the surface plot of GPCE 5. The data for the surface plot have been obtained as follows. As GPCE is a discriminant function that gives an output of $+1$ or -1, it divides the feature space into two regions that can be viewed pictorially by plotting the surface that separates the two regions. The surface is given by the following equation:

$$\text{GPCE}\,(X)=0$$

i.e., all those points in the feature space for which the above equation is satisfied lie on the surface that divides the feature space into two regions. The data for a portion of the surface were generated as follows:

for $(F2 = 27\text{–}45)$
for $(F3 = 28\text{–}62)$.
Determine the value of $F1$ for which GPCE $(F1, F2, F3) = 0$.

The secant rule was used to obtain the values of $F1$ for the above equation as the GPCE is a highly nonlinear function. After the data were obtained, the plot utilities in MATLAB were used to generate the surface.

E. SA Computation and Heuristic Rules

Table IX shows the class count matrix obtained for these GPCEs. For example, in our five-class problem, the number of samples for class 1 is 169. After the application of all GPCEs on samples of class 1, we obtain the first row of the class count matrix. The first row of the class count matrix is [155, 5, 1, 7, 6], which means that GPCE 1 returns a value of $+1$ for 155 samples belonging to class 1 out of 169 in the training set, GPCE 2 returns a value of $+1$ for 5 samples out of 169, and so on. The sum of the elements in the row vector is 174. As mentioned earlier, the sum of these elements need not be 169. The SA for GPCE 1 is $155/174 = 0.89$. In the same way, SA is computed for all of the GPCEs. Table X shows the strength of association measures for all of these GPCEs.

The classification matrix which reflects the accuracy of a classifier is obtained for this five-class problem by applying these GPCEs on the validation set. While obtaining the classification matrix, the conflict resolution can be done by using only the strength of association measures. Table XI shows the classification matrix obtained. The average accuracy and overall accuracy are 0.75 and 0.77, respectively.

The reasons for conflict and its resolution in GP-based multicategory pattern classification can be further illustrated by considering the above GPCEs with the following examples.

1) If $34 < F2 < 43$, GPCE 1 returns a value of $+1$. Similarly, if $F2 > 1.031F1$ and $F2 < 36$, GPCE 2 returns a value of $+1$. Thus, for samples in which $34 < F2 < 36$ and $F2 > 1.031F1$, both GPCE 1 and GPCE 2 return a value of $+1$. So, conflict resolution is needed to assign the true class. As mentioned earlier, this is done with the

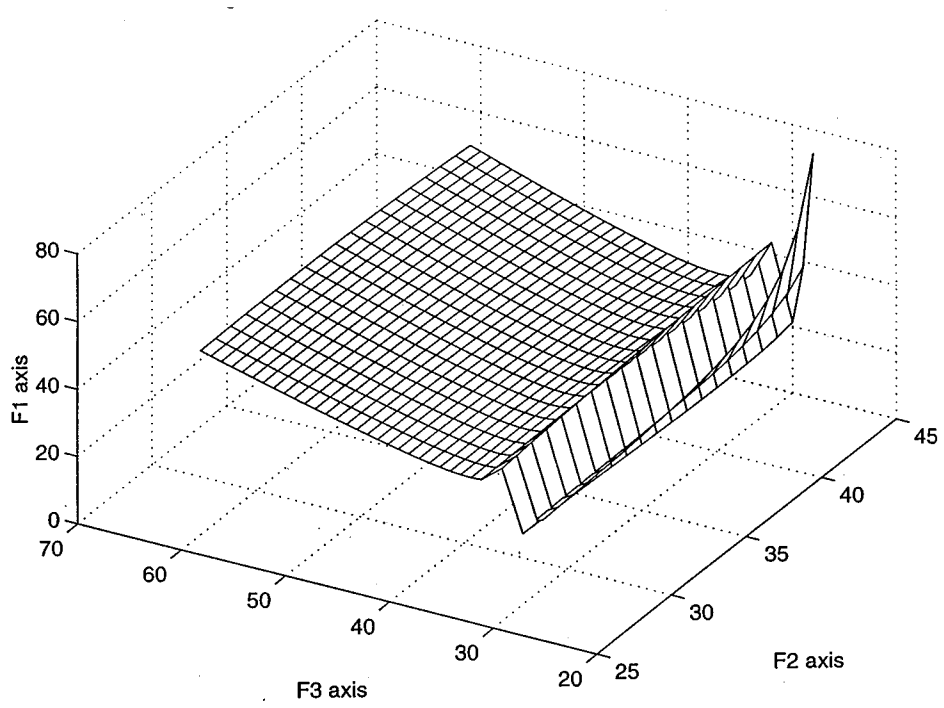


Fig. 2. Surface plot of GPCE 5 (*AZ, EL*) $= (-60, 45)$.

TABLE IX
CLASS COUNT MATRIX FOR THE GPCEs OBTAINED WITH 5% PI FOR INCREMENTAL LEARNING AND ARITHMETIC FUNCTION SET

GPCE Class	1	2	3	4	5
1	155	5	1	7	6
2	2	136	4	13	58
3	0	21	213	74	2
4	0	13	63	211	0
5	22	29	20	1	99

TABLE X
SAs FOR GPCEs OBTAINED WITH 5% PI FOR INCREMENTAL LEARNING AND ARITHMETIC FUNCTION SET

Class	1	2	3	4	5
SA	0.89	0.64	0.69	0.74	0.58

TABLE XI
CLASSIFICATION MATRIX BASED ON SA MEASURES

AC TC	1	2	3	4	5	RC	VS
1	155	10	0	1	2	0	168
2	1	123	7	14	0	0	145
3	1	0	138	74	0	0	213
4	0	0	0	211	0	5	216
5	33	36	0	5	45	5	124

Average accuracy = 0.752 Overall accuracy = 0.776
AC=Assigned Class TC=True Class RC=Reject Class

help of SA measures, and the class of the GPCE with the higher SA is assigned to the input, i.e., class 1 is assigned.

In an unlikely case, the GPCE with a higher SA swamps the GPCE with a lower SA, and this leads to misclassification. Consider the following hypothetical situation. Let the samples for which $34 < F2 < 36$ and $F2 > 1.03F1$ belong to class 2, and, let the SA of GPCE 1 be greater than that of GPCE 2. So, class 1 is assigned to these samples. To prevent misclassification, we can frame the following rule.

> If ($34 < F2 < 36$ and $F2 > 1.031F1$),
> then the true class is class 2.

However, this data relationship, which has been discovered by GP, is expressed by the following result vector $[1\ 1{-}1{-}1{-}1]$, which can be framed as the heuristic rule for preventing the misclassification of such samples.

2) Similarly, if $34 < F2 < 43$ and $F3 > 62$, we observe that GPCE 1 and GPCE 3 return a value of $+1$ for such samples. If the SA of GPCE 1 is greater than that of GPCE 3 and the true class happens to be class 3, then the following rule can be formulated to prevent misclassification.

> If ($34 < F2 < 43$ and $F3 > 62$),
> then the true class is class 3.

However, this data relationship will be expressed by the following result vector $[1 -1\ 1 -1 -1]$, and can be framed as a heuristic rule.

Algorithm 4 gives one possible means for identifying these rules. Let n_m be the number of misclassified samples in a class, and let α be the user-defined threshold for extraction of a rule. If $\alpha = 0.1$ and the number of samples in the training set for

a particular class is 100, then at least ten misclassified samples must have the same result vector. We have chosen $\alpha = 0.1$ in our experiments.

Algorithm 4 Discovery of Heuristic Rules

```
Begin
  for i = 1 to n
    {
     Analyze result vectors for n_m
       misclassified samples in
       class i.
     Let a particular result vector
       appear m_i times.
     If m_i > α_i × n_i, then the result
       vector becomes a heuristic rule;
    }
End
```

For example, if the result vector $[1, -1, 1, -1, 1]$ appears for samples of class 5 in our five-class problem, it will be misclassified as class 1 since the SA of GPCE 1 is greater than the SA of GPCE 3 and GPCE 5 . However, Algorithm 4 can identify such a result vector as a heuristic rule, and assign the true class (class 5) after overriding the class assigned by using only SA measures (class 1). Similarly, heuristic rules for other classes can be framed. The heuristic rules help in reducing misclassification. Table XII shows the two heuristic rules obtained for our five-class problem. Table XIII shows the classification matrix, obtained by using both SA and heuristic rules for conflict resolution.

Thus from Tables XI and XIII, it is clear that 21 samples out of 124 samples for class 5 and 25 samples out of 213 samples for class 3 are properly classified by the heuristic rules.

F. Performance of the MLC

Although our main objective is to show the feasibility of GP for the n-class pattern classification problem, for the sake of completeness, we will present the results of the MLC on the same five-class problem. The maximum likelihood classification is based on the assumption that the probability distribution for each class is a multivariate normal distribution [1]. The MLC is widely used for comparison. The discriminant function in MLC is given by

$$g_i(\boldsymbol{X}) = -\ln|\Sigma_i| - (\mathbf{X} - m_i)^t \Sigma_i^{-1} (\mathbf{X} - m_i), \quad i = 1, 2, \ldots, n \tag{9}$$

where

- $\boldsymbol{X}$ is the m-dimensional input feature vector;
- m_i is the mean feature vector for class i;
- Σ_i is the covariance matrix for class i, and is of size $m \times m$;
- n is the number of classes.

The same validation set was used to obtain the classification matrix for MLC, and the results obtained are presented in Table XIV. The average accuracy and overall accuracy are 0.789 and 0.809, respectively. By comparing Tables XIII and XIV, we observe that GP has a higher classification accuracy than MLC, and also has performed better for classes 1, 4, and 5, respectively. But for classes 2 and 3, MLC has a higher classification accuracy. We will further discuss the differences between GP-based classification and MLC in Section V.

TABLE XII
HEURISTIC RULES FOR REDUCING MISCLASSIFICATION

1	2	3	4	5	Class
1	-1	1	-1	1	5
-1	1	1	1	-1	3

TABLE XIII
CLASSIFICATION MATRIX BASED ON BOTH SA AND HEURISTIC RULES

AC / TC	1	2	3	4	5	RC	VS
1	155	10	0	1	2	0	168
2	1	123	8	13	0	0	145
3	1	0	163	49	0	0	213
4	0	0	3	208	0	5	216
5	12	36	0	5	66	5	124

Average accuracy = 0.806 Overall accuracy = 0.826

G. GP-Based Classification for Fisher's Iris Data Set

A second example we considered is the well-known Fisher's iris data set [11]. There are four features, namely, sepal length ($F1$), sepal width ($F2$), petal length ($F3$), and petal width ($F4$). The three classes are Iris Setosa (Class 1), Iris Versicolor (Class 2), and Iris Virginica (Class 3). The data set contains 50 instances for each of the three classes. The data set was scaled by a factor of 10, and was divided equally into a training set and a validation set. Table XV shows the characteristics of the Fisher iris data set.

The evolution of the GPCEs was done for both the skewed and interleaved data format with the arithmetic function set and 5% PI for incremental learning with the GP parameters shown in Table IV.

Skewed Training Data Sets: The evolved GPCEs are given below.

GPCE 1: (SUB (SUB (DIV 98 $F2$)(ADD $F4$ $F1$))(SUB(MUL $F4$ $F1$)(MUL 25 $F2$))).

GPCE 2: (DIV (MUL (SUB (DIV $F1$ $F4$)(ADD 114 -30))(ADD (ADD -103 $F2$)(SUB $F1$ $-$ 11)))(DIV (SUB (ADD 103 106)(MUL $F3$ $F4$))(DIV (DIV -44 $F4$)(DIV $F3$ $F3$)))).

GPCE 3: (SUB (ADD (ADD 84 $F4$)(MUL $F2$ $F2$))(MUL (ADD $F3 - 67$)(SUB $F4$ 86))).

The class count matrix was obtained to determine the SA measures. The SA measures for the GPCEs are 0.86, 0.94, and 0.64, respectively. Table XVI shows the resulting classification matrix.

Interleaved Data Format Training Sets: The evolved GPCEs are as follows.

GPCE 1: (MUL (SUB (SUB (MUL $F3$ (SUB $F3$ $F1$)) (SUB (MUL -52 $F2$)(DIV($F4$ $F1$))) (ADD (SUB (MUL

TABLE XIV
CLASSIFICATION MATRIX FOR MAXIMUM LIKELIHOOD CLASSIFIER

AC TC	1	2	3	4	5	VS
1	147	15	0	0	6	168
2	0	132	8	3	2	145
3	0	0	175	37	1	213
4	5	0	16	191	4	216
5	0	36	21	11	56	124

Average accuracy = 0.789 Overall accuracy = 0.809

TABLE XV
DATA CHARACTERISTICS OF FISHER'S IRIS DATA SET

Class	F1	F2	F3	F4	TS	VS
	Min Max	Min Max	Min Max	Min Max		
Iris Setosa	43 58	29 42	10 19	1 5	25	25
Iris Versicolor	50 70	20 34	30 51	10 18	25	25
Iris virginica	57 79	25 34	45 69	17 25	25	25

$F1$ $F3$) (MUL $F2$ $F4$))(ADD (ADD -109 $F1$) (SUB $F1$ $F1$))))(SUB (MUL (ADD (DIV $F1-30$) (MUL 62 $F2$)) (DIV $F1$ $F2$)) $F1$)).

The equivalent mathematical expression is given by

$$\left(\left(\left(F3*(F3-F1)+\left(52F2+\frac{F4}{F1}\right)\right)-(F1*F3-F2*F4+F1-109)\right)*\left(\left(\left(\frac{F1}{-30}+62F2\right)*\left(\frac{F1}{F2}\right)\right)-F1\right)\right). \quad (10)$$

GPCE 2: (DIV (DIV (SUB $F2$ $-$ 113) (SUB 49 $F3$)) (ADD $F4$ (DIV -108 $F3$))).

The equivalent mathematical expression is given by

$$\frac{(F2+113)*F3}{(49-F3)(F3*F4-108)}. \quad (11)$$

GPCE 3: (ADD (MUL (ADD -20 $F3$) (ADD $F3$ $F4$)) (ADD (MUL $F2-58$)(SUB 120 122))).

The equivalent mathematical expression is given by

$$(F3-20)*(F3+F4)-(58F2+2). \quad (12)$$

The class count matrix was obtained and is shown in Table XVII. The resulting SA measures for the GPCEs are 1.0, 0.86, and 0.89, respectively. Table XVIII shows the classification matrix obtained by using SA measures only.

We will now explain how the GPCEs learned the data distribution during the training phase. For this purpose, consider only the features $F3$ and $F4$. GPCE 2 will return -1 only when $F4$ is less than a particular value for a given $F3$. GPCE 3 will return -1 when $F3<20$ irrespective of other features. This is shown in Fig. 3. In Fig. 3, all of the points in $F3$ and $F4$ below GPCE 2 will return -1, and all of the points on the left of GPCE 3 will also return -1. The data points for class 1 are also shown. GP discovers that only $F3$ and $F4$ are the discriminant features in the classification between class 1 and the other two classes.

TABLE XVI
CLASSIFICATION MATRIX FOR GP WITH SKEWED TRAINING SETS FOR IRIS DATA

AC TC	1	2	3	RC	VS
1	22	3	0	0	25
2	0	20	0	5	25
3	0	10	15	0	25

Average accuracy = 0.76 Overall accuracy = 0.76

TABLE XVII
CLASSIFICATION MATRIX FOR GP WITH INTERLEAVED TRAINING SETS FOR IRIS DATA SET

AC TC	1	2	3	TS
1	25	0	0	25
2	0	25	4	25
3	0	3	25	25

TABLE XVIII
CLASSIFICATION MATRIX FOR GP WITH INTERLEAVED TRAINING SETS FOR IRIS DATA SET

AC TC	1	2	3	VS
1	25	0	0	25
2	0	22	3	25
3	0	0	25	25

Average accuracy = 0.96 Overall accuracy = 0.96

This is the reason why all of the samples belonging to class 1 are classified correctly. Similarly, to classify data points between classes 2 and 3, GP discovers that $F2$ is the discriminant feature. This is shown in Fig. 4. In Fig. 4, the curve GPCE 2 has two regions. One region is $F3<49$, and the other $F3>49$. All of the points in $F3$ and $F4$ above the curve GPCE 2 will return $+1$ in the region $F3\leq 49$, and all of the points below the curve GPCE 2 will return $+1$ in the region $F3>49$. In Fig. 4, the GPCE 3 is also shown for various values of $F2(20, 25, 30, 35)$. For a given value of $F2$, all of the points on the right side of the curve will return $+1$.

The data points for classes 2 and 3 are also shown in Fig. 4 for different values of $F2$, the regions for which GPCE 3 returns $+1$ are indicated. We observe that as the value of $F2$ increases, the region for which GPCE 3 returns $+1$ shifts to the right, and hence the confusion between classes 2 and 3 decreases.

As in the earlier example, we have compared the performance of GP with that of MLC. Table XIX gives the results obtained by using MLC. The results show a good agreement.

V. SOME IMPORTANT ISSUES IN GP-BASED PATTERN CLASSIFICATION

In GP, learning takes place during evolution, and is guided by an appropriate fitness function. This evolutionary approach is different from a statistical approach like the MLC or a trainable classifier like the neural network which uses an error function for updating the weights. In this section, we will discuss some

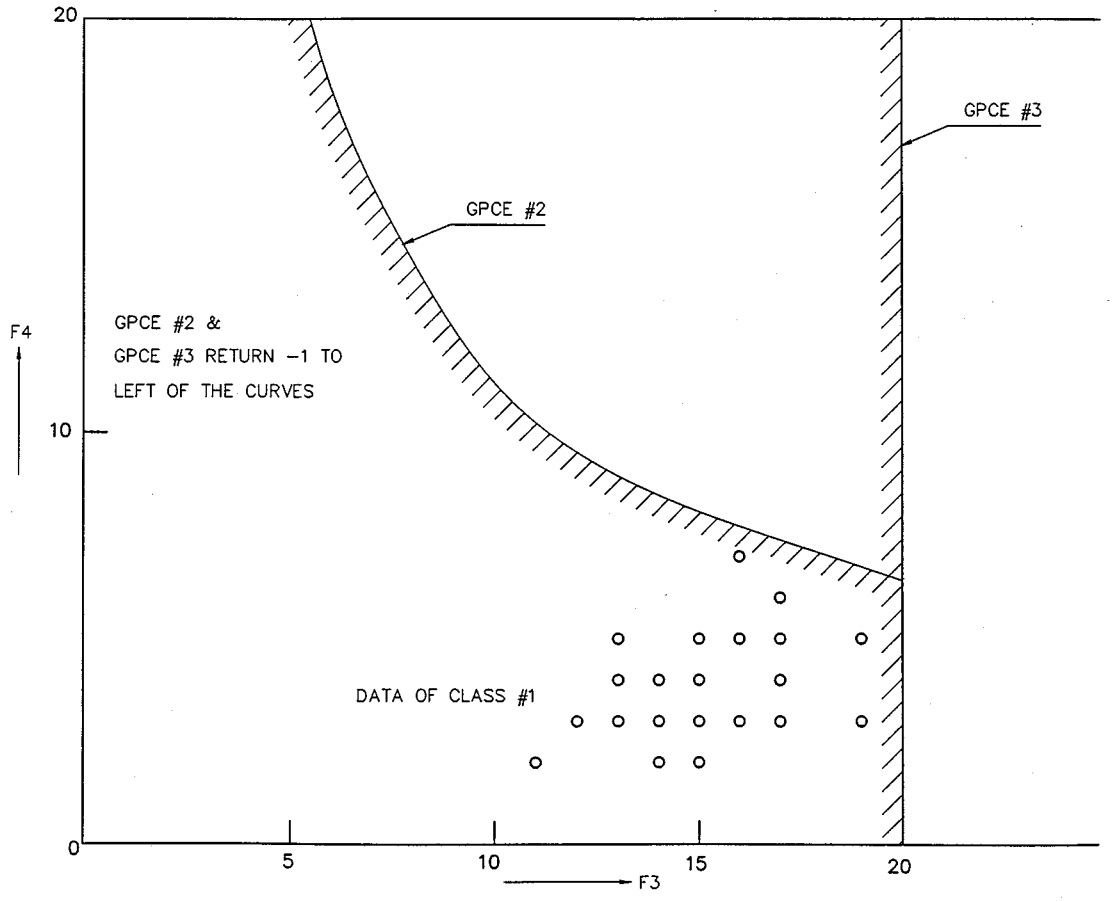


Fig. 3. Behavior of GPCE 2 and GPCE 3 for data of class 1.

of the important issues that arise in our approach to GP-based pattern classification.

A. Interleaved Data Format

Both the MLC and the neural network deal with the n-class problem directly. In the MLC, a representative mean vector and covariance matrix are computed for each class from its own set of representative samples. Thus, samples of class *A* do not influence the mean vector and covariance matrix of class *B*. For a given input, the mean vectors and the covariance matrices of all of the classes are used to determine the likelihood values, and the class with the highest likelihood value is assigned to the input. In the neural-network classification, a single neural network is trained for an m-input n-class problem. Consider a three-input and five-class problem. The desired output for samples of class 1 is [0.8 0.2 0.2 0.2 0.2], [0.2 0.8 0.2 0.2 0.2] for class 2, and so on. As samples are expected to be representative in each class and the network is trained on the entire training set during each epoch, the network is able to simultaneously learn the decision boundaries for all of the classes. So, the interleaved data format which was proposed in GP to overcome skewness as the n-class problem is converted into n two-class problems does not arise in both MLC and the neural network. The interleaved data format is an artifact that has been used to reduce skewness in the training set that is used for an evolution of a GPCE.

B. Incremental Learning

The MLC is basically a statistical classifier. In this statistical approach, since averaging is involved, a better estimate is made for the mean vector and covariance matrices as the number of samples for a class increases. The mean vector and the covariance matrices characterize the data distribution for a class. There is no learning involved in the MLC as there is neither an error function as in a neural network nor a fitness function as in GP. In a neural network, learning can be done in a batch mode, or on a sample-by-sample basis, or even on a subset of the training set. So, incremental learning has been used in neural networks [4]. Batch mode learning is essentially global learning as the mean error for all of the samples is computed, and updating of weights is done only once in each epoch. On the other hand, when the network is trained on a sample-by-sample basis, we have incremental learning as the error for each input is determined, and the network is updated before being fed with the next input. Thus, sample-by-sample learning represents incremental learning taken to its limit in each epoch. However, in each epoch, the entire training set is used. As GP is also a learning approach like the neural network, the learning can be either global or incremental. The percentage increment for incremental learning

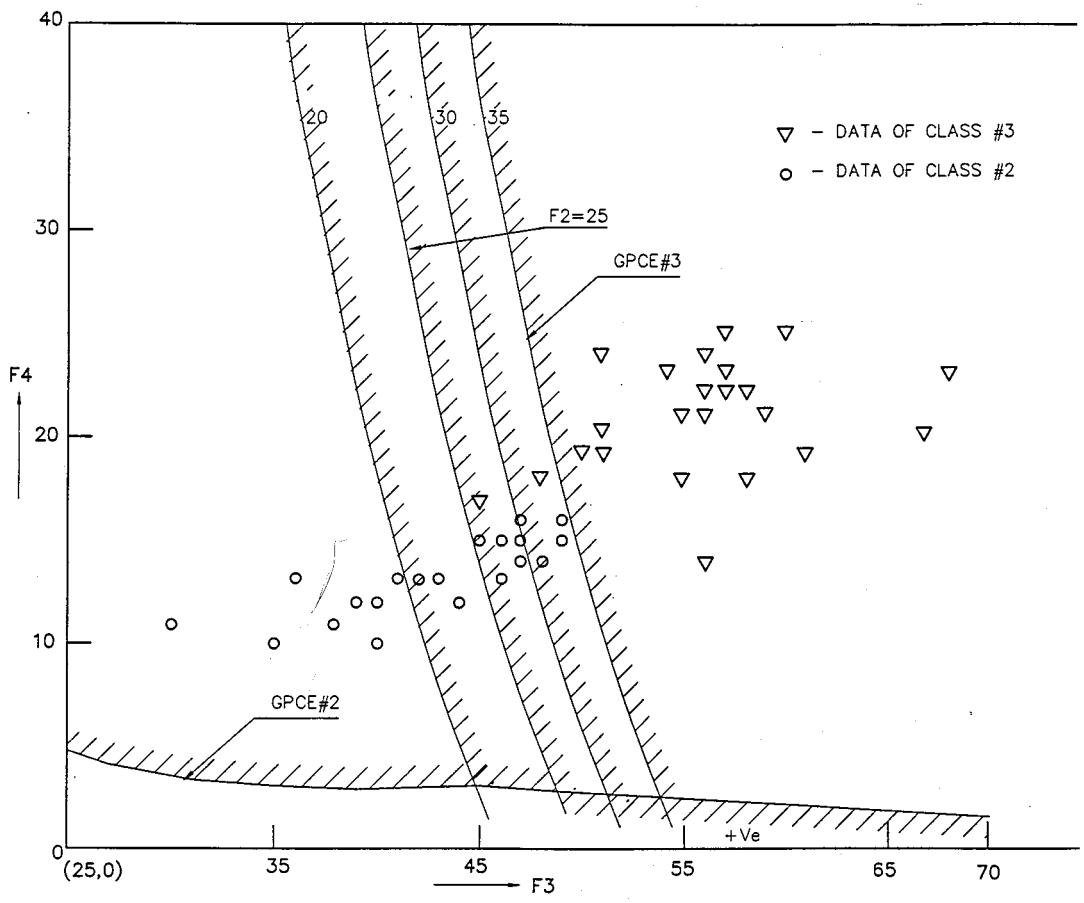


Fig. 4. Performance of GPCE3 for different values of $F2$.

TABLE XIX
CLASSIFICATION MATRIX FOR IRIS DATA SET WITH MAXIMUM LIKELIHOOD CLASSIFIER

AC / TC	1	2	3	VS
1	25	0	0	25
2	0	24	1	25
3	0	2	23	25

Average accuracy = 0.96 Overall accuracy = 0.96

can be selected by the user to gradually increase the size of the training set during evolution.

C. Conflict Resolution

Both the MLC and the neural network have a simple approach to conflict resolution. In the MLC, the likelihood value is computed for each class by using the corresponding mean vector and covariance matrix for a given input. The class of the maximum likelihood value is assigned to the input. Similarly, in the neural network, for a given input, we get an output vector. Each element in the output vector is in the range [0, 1]. For example, in a five-class problem, the output vector can be [0.1 0.2 0.8 0.4 0.5] for a given input. So, class 3 is assigned to the input. This simple approach is an outcome of the n-class problem being handled directly by the neural network. In a two-class problem, a discriminant function can return a value of $+1$ or -1 to indicate whether or not the data belong to a class. When the n-class problem is converted into n two-class problems as in GP, we get a result vector containing $+1$ or -1 as its entries. When more than one GPCE returns a value of $+1$, there is a need for conflict resolution, which is done indirectly by using SA measures. The SA measures indicate how well a GPCE can recognize samples belonging to its own class, and reject samples belonging to other classes.

D. Scope for Heuristic Rules

In both the MLC and neural network, the output vector for a given input consists of real numbers. In the MLC, the output vector consists of likelihood values which vary for each input. As the likelihood values are varying, it is not possible to identify a specific likelihood value vector as an output vector that appears for misclassified samples of a particular class. Similarly, in a neural network, the output vector has elements in the interval [0, 1]. The output vector varies even for misclassified samples of the same class. So, it is not possible to identify a specific output vector, and to assign it as a rule to overcome misclassification of certain samples belonging to a particular class. On the other hand, in GP, as the output vector consists of $+1$ or -1 as its entries, it is possible for a specific result vector to occur for misclassified samples of a particular class. This result vector can be framed as a heuristic rule to assign the true class for these

samples. As the heuristic rule represents the data relationship discovered by GP, the creation of the heuristic rules should be seen as part of GP-based classification. Thus, GP gives a scope for heuristic rules to reduce misclassification, which is a very important attribute that is not available in either the neural-network approach to classification or in the MLC.

E. Reject Class

For a particular input sample, it is possible for a result vector to contain all -1 entries. Such a sample can be assigned to the reject class. However, it does not affect the average accuracy and the overall accuracy of the classifier as they are basically dependent on the main diagonal of the classification matrix. As the samples with all -1 entries in the result vector are very small (e.g., only 10 samples out of 866 in the classification matrix shown in Table XIII), it shows that GP can be successfully applied for an n-class problem. It should be noted that it is possible to have a reject class, even in the neural network. For example, if all of the outputs of the neural network are very low, the input can be assigned to the reject class. Ideally, one of the outputs in the neural network should be high so that a class can be assigned to the input. As MLC is essentially a distance classifier, the given input is always assigned to one of the classes.

A possible explanation for the reject class in GP is as follows. The given data set is divided equally into a training set and a validation set. If the division is done in a random manner, it can lead to a situation where data points for a certain region of the feature space are not present in the training set, and are present in the validation set. In such a scenario, the GPCEs would not be trained to recognize samples belonging to that region. So, it is possible for all of the GPCEs to return a value of -1, i.e., the result vector will contain -1 as its entries. Such samples can be classified as the reject class. For example, in our experiment, five samples each in classes 4 and 5 were classified as reject class. So, we believe that, if the training set contains samples from all regions of the feature space, such a situation (i.e., the reject class) is very unlikely. The presence of a reject class does not mean that GP is lacking in generalization. But the generalization power of GP is only within the training data set. For data away from the training set, it is possible for the GPCE to reject this sample.

F. GP and AI-Based Machine Learning

The process of knowledge acquisition can be divided into two categories: symbolic and nonsymbolic. Nonsymbolic systems represent knowledge implicitly. For example, in neural networks, knowledge is distributed among the network connections. On the other hand, in symbolic systems like GP, knowledge is expressed explicitly. Both AI-based machine learning and GP have many similarities as they are learning systems that build knowledge structures by using input–output examples. However, conventional AI systems have implemented machine learning by using logic and heuristics, while GP has realized it by using the principles of natural evolution. While a heuristic is used to guide the search in AI for obtaining a solution, a fitness function is used in GP to guide the search for a solution.

VI. Conclusion

In this paper, we have demonstrated the applicability of GP to an n-class pattern classification problem by considering a real-world data set taken from remotely sensed images and the well-known Iris data set. As the n-class problem has been modeled as n two-class problems, we need n GPCEs and hence n GPCE specific training sets. If we create a training set directly, it leads to skewness (as $n_i \ll N_i$), and hence poor classification. To overcome the skewness, an interleaved data format is proposed. The experimental results show that the interleaved data format performs better than the skewed data set. We have introduced incremental learning to allow learning on a subset of the training set to simplify the task of learning during evolution. This subset is gradually increased to cover the entire training set. The performance of the GP classifier based on incremental learning is better than the performance obtained using the traditional global learning.

We have also observed that the GPCEs evolved with an arithmetic function set performed better than GPCEs evolved with other function sets containing logical and nonlinear elements. Hence, we have used the arithmetic function set, incremental learning, and interleaved data format to evolve GPCEs. Each GPCE is trained to recognize samples belonging to its own class, and to reject samples belonging to other classes. A strength of association measure is associated with each GPCE to indicate the degree to which it can recognize samples belonging to its own class. The strength of association measures are used for assigning a class to an input feature vector. Heuristic rules can be used to prevent a GPCE with a higher SA from swamping a GPCE with a lower SA, which further improves the performance of a GP classifier. For the sake of completeness, we have also presented the results of MLC. We also observe that there is variation in the performance of GP as it is essentially a nonalgorithmic approach to solving problems. However, it can automatically discover the discriminant features for a class, unlike MLC.

In our approach to GP-based classification, the choice of the GP parameters has been largely empirical. Future work should lie in the adaptive variation of these GP parameters, and in discovering any empirical relationship among the data distributions and in the selection of GP parameters for evolving the GPCEs.

Appendix A
Definition of GPQUICK Parameters

1) *Copy Weightage:* The copy operation selects a member of the population, and replaces it by randomly choosing another member in the population. There is only reproduction and no crossover and mutation.
2) *Crossover Weightage:* This indicates the probability of choosing the crossover operation.
3) *Mutation Weightage:* This indicates the probability of choosing the mutation operation.
4) *Crossover Weightage Annealing:* This indicates the probability of introducing the offspring after the crossover operation, only if it is fitter than the parent; otherwise, it is discarded.

5) *Mutation Weightage Annealing:* This indicates the probability of introducing the offspring after the mutation operation only if it is fitter than the parent; otherwise, it is discarded.

The parameters discussed above indicate the possible operations that can be performed on the members of the population. A roulette-wheel strategy is used to select one of the above operations. It is important to note that the sum of the above weightage parameters is equal to 1.

6) *Crossover Rate:* If operation 2) or 4) is selected, it is carried out with a probability given by the crossover rate. Generally, a high value is chosen for the crossover rate.
7) *Mutation Rate:* If operation 3) or 5) is selected, it is carried out with a probability given by the mutation rate. Generally, a low value is chosen for the mutation rate so that changes in the population do not take place rapidly.

Mutation can result in one of the following three actions.

- *Mutation Node:* This indicates the probability of an existing sub-tree being replaced by another subtree at a given node.
- *Mutation Constant:* This indicates the probability of a constant value being replaced by another constant value at a given node.
- *Mutation Shrink:* This indicates the probability of replacing an existing subtree by a randomly generated constant at a given node.

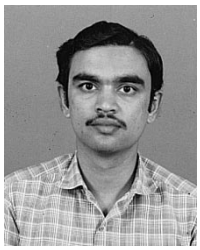
It is important to note that the sum of these probabilities is also equal to 1. The selection of these actions is done by a roulette-wheel strategy. GPQUICK uses protected division during evolution. The result of any divide-by-0 operation is equated to unity [11].

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their useful comments and suggestions on an earlier version of this paper. In particular, one of the reviewers has brought to their attention the possibility of all GPCEs returning a value of -1, which gave rise to the notion of the reject class. The authors would like to also thank Dr. D. B. Fogel, Editor-in-Chief, and also the Associate Editor for their helpful comments that improved the quality of the paper significantly.

REFERENCES

[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[2] P. D. Heerman and N. Khazenie, "Classification of multispectral remote sensing data using a back propogation neural network," *IEEE Trans. Geosci. Remote Sensing*, vol. 30, no. 1, pp. 81–88, 1992.
[3] L. C. Pretorius and C. Nel, "Feature extraction from ECG for classification by artificial neural networks," in *Proc. IEEE 5th Symp. Comput. Based Med. Syst.*, 1992, pp. 639–647.
[4] S. Haykin, *Neural Networks—A Comprehensive Foundation*. New York: Macmillan College, 1994.
[5] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.
[6] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: M.I.T. Press, 1992.
[7] S. B. Cho and K. Shimohara, "Modular neural networks evolved by genetic programming," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1996, pp. 681–684.
[8] A. N. Edmonds, "Genetic programming of fuzzy logic production rules," in *Proc. IEEE Int. Conf. Evol. Comput.*, 1995, pp. 765–770.
[9] J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Trans. Evol. Comput.*, vol. 1, no. 2, pp. 109–128, 1997.
[10] I. Benyahia and J. Y. Potvin, "Decision support for vehicle dispatching using genetic programming," *IEEE Trans. Syst., Man, Cybern. A*, vol. 28, no. 3, pp. 306–314, 1998.
[11] A. Singleton, "Genetic programming with C++," *Byte*, pp. 171–176, Feb. 1994.
[12] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, pt. II, vol. 7, pp. 179–188, 1936.
[13] T. Yoshida and S. Omatu, "Neural network approach to land cover mapping," *IEEE Trans. Geosci. Remote Sensing*, vol. 32, no. 5, pp. 1103–1108, 1994.



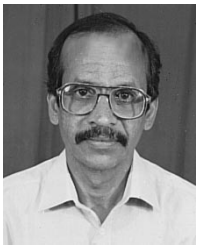
J. K. Kishore received the B.Tech. degree in electrical engineering from IIT Madras in 1987, and the M.Tech. degree in electronics from IIT Bombay in 1989. He is currently working toward the Ph.D. degree at the Department of Aerospace Engineering, Indian Institute of Science, Bangalore.

He joined ISRO Satellite Center, Bangalore, in 1990 after a brief stint at WIPRO Information Technology Ltd., also in Bangalore. He has been involved in the design and development of on-board computer systems for satellites developed at the ISRO Satellite Center, Bangalore. He is presently the Section Head for the On-Board Computers Section. His research interests involve using evolutionary computing in general and genetic programming in particular for pattern classification applications.

L. M. Patnaik (S'75–M'76-SM'86–F'92) received the Ph.D. degree in 1978 in the area of real-time systems, and the D.Sc. degree in 1989 in the areas of computer systems and architectures, both from the Indian Institute of Science, Bangalore.

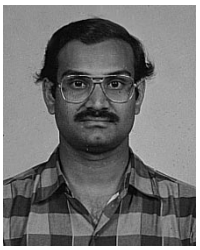
Currently, he is a Professor with the Electrical Sciences Division at the Indian Institute of Science. He directs his research group in the Microprocessor Applications Laboratory at the Institute. During the last 25 years of his service at the Institute, his teaching, research, and development interests have been in the areas of parallel and distributed computing, computer architecture, CAD of VLSI systems, computer graphics, theoretical computer science, real-time systems, neural computing, and genetic algorithms. In these areas, he has published over 140 papers in refereed international journals and over 160 papers in refereed international conference Proceedings, on the theoretical, software, and hardware aspects of the above areas of computer science and engineering. He is a Coeditor/coauthor of five books in the areas of VLSI system design and parallel computing. He has supervised 18 doctoral dissertations and over 120 Master's theses in the above areas. In the areas of parallel and distributed computing, and neural computing, he has been a Principal Investigator for a large number of Government-sponsored research projects, and a Consultant to a few industries. He is a Distinguished Lecturer of IEEE Region 10. He is a Distinguished Visitor of the IEEE Computer Society for the Asia–Pacific region. In recognition of his contributions in the areas of electronics, informatics, telematics and automation, he was awarded the Dr. Vikram Sarabhai Research Award in 1989. He was awarded the Dr. Ramlal Wadhwa Award for the year 1992 by the Institution of Electronics and Telecommunication Engineers for his contributions during the last ten years in the fields of electronics and telecommunication. He was awarded the Samanta Chandrasekhar Award for the year 1994 by the Orissa Science Academy. He was awarded Hands for Indian IT Award for his contributions to the Indian IT (information technology) industry in 1998. He was awarded the Certificate of Appreciation, "For excellent service and dedication to the objectives of the IEEE Computer Society as a Distinguished Visitor," in 1998. He was awarded the Distinguished Professional Engineer Award by the Computer Engineering Division of the Institution of Engineers (India) in 1999. He was awarded the IEEE Computer Society's 1999 Technical Achievement Award for his contributions in the field of parallel, distributed, and soft computing, and

high-performance genetic algorithms. He is Fellow of the Third World Academy of Sciences, the Computer Society of India, the Indian National Science Academy, the Indian Academy of Sciences, the National Academy of Sciences, and the Indian National Academy of Engineering. He is a Life Member of the VLSI Society of India and the Instrument Society of India, a founding Member of the Executive Committee of the Association for the Advancement of Fault-Tolerant and Autonomous Systems, and a Fellow of the Institution of Electronics and Telecommunications Engineers and the Institution of Engineers. His name appears in the 8th edition of the *Who's Who in the World* (Marquis Publications), *Reference Asia*, *Asia's Who's Who of Men and Women of Achievement*, and the *Directory of Distinguished Computer Professionals of India*. He was the Program Chair (1990, 1991, and 1996) and General Chair (1992) for the IEEE-sponsored International Conference on VLSI Design, a member of the Programme Committee for the 6th International Parallel Processing Symposium (1992) and the 22nd Annual International Symposium on Fault-Tolerant Computing (1992), both sponsored by the IEEE, Program Chair(Asia/Australia) of the IEEE Symposium on Parallel and Distributed Processing, 1993, a member of the Asian Subcommittee of the ACM Multimedia '93 Symposium, a member, of the Executive Committee and Coordinator, Asia and Pacific Rim, for the IEEE Technical Committee on Parallel Processing, Technical Cochair and member, Steering Committee, of the IEEE-sponsored 1st International Workshop on Parallel Processing (1994, 1995), a member of the Program Committee of the Technical Session on Parallel Processing Technology and Applications, IEEE Tencon'94, Program Chairman for the CSI 96 Convention and a member of the Steering Committee of the High Performance Computing Asia 97 Conference. He was a member of the IEEE CS Fellowship Committee for the year 1994. He has been the Chairman of the IEEE Computer Society Chapter of the Bangalore Section for two years. He is the founding Chairman of the Indian Transputer User Group (ITUG), presently called the Advanced Computing Society (ACS). He is a member of the Editorial Boards of the *International Journal of High Speed Computing*, *Journal of Computer-Aided Design*, *VLSI Design: An International Journal of Custom Chip Design, Simulation, and Testing*, *The Computer Journal*, *Parallel Algorithms and Applications*, *IEEE Transactions on VLSI Systems*, *Evolutionary Optimization: An International Journal*, *Microprocessors and Microsystems*, and P*arallel and Distributed Computing Practices Journal*; Editor of the *Journal of Computer Science and Informatics*. He has delivered several Invited Lectures in the United States, Canada, France, the United Kingdom, the former Yugoslavia, Hungary, Switzerland, Australia, New Zealand, Hong Kong, Singapore, Malaysia, and Japan.



V. Mani received the B.E. degree in civil engineering from Madurai University in 1974, the M.Tech. degree in aeronautical engineering from the Indian Institute of Technology, Madras, in 1976, and the Ph.D. degree in engineering from the Indian Institute of Science (IISc), Bangalore, in 1986.

From 1986 to 1988, he was a Research Associate at the School of Computer Science, University of Windsor, Windsor, ON, Canada, and from 1989 to 1990 at the Department of Aerospace Engineering, Indian Institute of Science. Since 1990, he has been with IISc, Bangalore, where he is presently an Associate Professor in the Department of Aerospace Engineering. His research interests include distributed computing, queuing networks, evolutionary computing, neural computing, and mathematical modeling. He is the coauthor of a book, *Scheduling Divisible Loads in Parallel and Distributed Systems* (Los Alamitos, CA: IEEE Computer Society Press).



V. K. Agrawal received the M.Tech. degree from IIT, Kanpur, and the Ph. D. degree from the Department of Computer Science, Indian Institute of Science, Bangalore, in 1986.

He joined the ISRO Satellite Center in 1978. He has been largely responsible for the development of microprocessor-based on-board computer systems for the satellites developed at ISRO Satellite Center, Bangalore. Presently, he is working as the Group Director for the Control Systems Group. His research interests are in the area of Petri nets and evolutionary computing.