Genetic Programming for Simultaneous Feature Selection and Classifier Design

Durga Prasad Muni, Nikhil R. Pal, *Fellow, IEEE*, and Jyotirmoy Das

***Abstract*—This paper presents an online feature selection algorithm using genetic programming (GP). The proposed GP methodology simultaneously selects a good subset of features and constructs a classifier using the selected features. For a c-class problem, it provides a classifier having c trees. In this context, we introduce two new crossover operations to suit the feature selection process. As a byproduct, our algorithm produces a feature ranking scheme. We tested our method on several data sets having dimensions varying from 4 to 7129. We compared the performance of our method with results available in the literature and found that the proposed method produces consistently good results. To demonstrate the robustness of the scheme, we studied its effectiveness on data sets with known (synthetically added) redundant/bad features.**

***Index Terms*—Classification, classifier design, evolutionary algorithm, feature selection, genetic programming.**

I. Introduction

FEATURE selection (FS) is a process to select useful features to obtain an efficient and improved solution to a given problem. Ideally, the FS process should select an optimum subset of features from the set of available features which is necessary and sufficient for solving the problem. FS is important because all available features may not be useful. Some of the features may be redundant while some others may cause confusion during the learning phase. These features unnecessarily increase the complexity of the feature space which in turn demands more computation time for learning or finding a solution to the given problem.

FS algorithms can be grouped [1] based on characteristics of searching: *exhaustive, heuristic, and random*. Alternatively, they can also be categorized [1] into five groups based on the evaluation function that is used for evaluating the utility of the features: *distance* [2], [3], *information* [4], [5], *dependence* [6], *consistency* [7], [8] and *classifier error rate*. The FS techniques which use classifier error rate as the criterion are called *wrapper* type algorithms [9]. In a wrapper approach, the classifier for which features are being selected, itself is used as the evaluation function. Since the suitability of features depends upon the concerned learning algorithm or classifier, the wrapper type algorithms usually perform better (in terms of accuracy) compared to other type called *filter* type techniques. However, wrapper approaches require more computation time than filter approaches. In a filter-type method, the selection is done independent of the learning algorithms. In this case, the relevant features are filtered out from the irrelevant ones prior to the learning. Good surveys on FS include [1], [10]. Analysis and discussion of various FS techniques are available in [11]–[16], [9]. The sequential forward selection (SFS) [17] methods start from an empty set and gradually add features selected by some evaluation function while the sequential backward selection (SBS) [17] schemes start with the complete feature set and discard features one by one till an optimum feature subset is retained. However, in SFS once a feature is selected, it cannot be rejected later and reverse is true for SBS. Sequential forward floating selection (SFFS) [18] avoids this drawback. In [14] and [11], it has been shown that SFFS performs better over many conventional FS techniques. In [3], the branch and bound method is used to find an optimal feature subset with an assumption of a monotonic evaluation function. Relief [2] is a feature weight-based statistical approach whereas in [19], tabu search is used to find useful features. Some neural network based FS schemes are discussed in [20], [21], and [22]. Support vector machines (SVMs) have also been used for FS. In [23] a FS technique based on pruning analysis for SVMs is proposed. It enjoys characteristics of both filter and wrapper approaches. Filter approach reduces computational time whereas wrapper approach improves classification accuracy. Authors in [24] discuss FS and feature ranking using SVMs.

Evolutionary algorithms have also been used for FS [25]–[27], [29]–[31]. Evolutionary algorithms are random search techniques. Typically, in a genetic algorithm (GA) based FS approach [25], [26], each individual (chromosome) of the population represents a feature subset. For an n-dimensional feature space, each chromosome is encoded by an n-bit binary string $\{b_1, b_2, \ldots, b_n\}$, $b_i = 1$ if the ith feature is present in the feature subset represented by the chromosome and $b_i = 0$, otherwise. A classifier is used to evaluate each chromosome (or feature subset). Typically each chromosome is evaluated based on the classification accuracy and the dimension of the feature subset (number of 1s). In [11], it has been shown that GA based FS performs better than many conventional FS techniques for high-dimensional data. In an evolutionary paradigm, a population of candidate solutions to the problem is allowed to evolve to achieve the desired optimal or sub-optimal solution using a set of genetically motivated operations.

Foroutan and Sklansky [25] used branch and bound technique for FS using GA. Casillas *et al.* [27] devised a genetic FS scheme for fuzzy rule based classification systems. In [28], ADHOC is a genetic algorithm based FS scheme with C4.5 induction learning. Pal *et al.* [29] proposed a new genetic operator called self-crossover for FS.

Unlike GA, there have been only a few attempts to use genetic programming (GP) [32], [33] for FS. Gray *et al.* [34] analyzed GP classifiers designed for two-class problems and decided on the features to be selected. They have not considered any special step for FS during the GP evolution. Ahluwalia and Bull [30] proposed a coevolutionary approach for feature extraction and selection using *automatic defined function* (ADFs) of GP. Each ADF is assigned its own independent sub-population which co-evolves with other ADF sub-populations and a population of main program trees. They used the k-nearest neighborhood (K-NN) algorithm for classification. Sherrah *et al.* [31] used GP for feature extraction/selection. They used a generalized linear machine as the classifier. In [35], a GP-based feature extraction scheme is proposed for the classification of bearing faults.

In [20], Pal and Chintalapudi proposed a novel scheme of selecting the relevant features online while training a neural network. In the online approach, selection of features and construction of the classifier are done simultaneously producing a good system for a given problem. Chakraborty and Pal also introduced neuro-fuzzy schemes for online FS [36], [37].

This paper presents an *online* FS algorithm using GP, named, $\mathrm{GP}_{\mathrm{mtfs}}$ (multitree genetic programming based FS). For a c-class problem, a population of classifiers, each having c trees is constructed using a randomly chosen feature subset. The size of the feature subset is determined probabilistically by assigning higher probabilities to smaller sizes. The classifiers which are more accurate using a small number of features are given higher probability to pass through the GP evolutionary operations. As a result, we can achieve a good classifier using a small feature subset. We introduce two new crossover operations to suit the FS process.

Use of GP for online FS has some advantages. GP provides a mathematical representation of the classifier. As the classifier uses only a few selected features, the mathematical representation of the classifier can be easily analyzed to know more about the underlying system.

Before explaining how FS and classifier design can be integrated together, we describe how GP can be used for only classifier design.

II. Designing Classifiers Without Feature Selection

For a two-class problem a classifier can be represented by a binary tree (T). For a pattern $\mathbf{x}$, if $\mathrm{T}(\mathbf{x}) \geq 0, \mathbf{x} \in$ class 1

else $\mathbf{x} \in$ class 2.

For multicategory c-class problems, we usually require a classifier consisting of c such binary trees $(T_1, T_2, \ldots, T_c)$. For a pattern $\mathbf{x}$,

if $T_i(\mathbf{x}) \geq 0$ and $T_j(\mathbf{x}) < 0$ for all $j \neq i, i, j \in \{1, 2, \ldots, c\}$, then $\mathbf{x} \in$ class i.

However, if more than one tree show positive response $(T_i(\mathbf{x}) \geq 0)$ for a pattern $\mathbf{x}$, then a conflict resolution technique, as explained later, will be used to assign a single class to the pattern. Fig. 1 shows a typical multitree classifier.

Here we provide a brief outline of our multitree classifier design $(\mathrm{GP}_{\mathrm{mt}})$ without FS. A detailed description of a *similar* methodology can be found in our earlier publication [38].

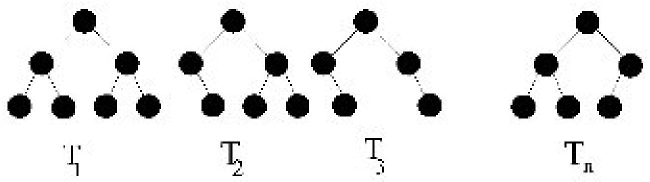

Fig. 1. Typical multitree classifier.

A. Initialization

A population of N_c multitree classifiers is generated. Each tree of each individual or classifier is randomly generated using the function set $F = \{+, -, *, /\}$ and terminal set $T = \{\text{feature variables}, R\}$. R contains randomly generated constants in [0.0, 10.0].

B. Fitness Measure

The population of classifiers is allowed to learn using a set of training samples, $\mathrm{X} = X_1 \cup X_2 \cup \cdots X_k \cdots \cup X_c, X_i \cap X_{j \neq i} = \phi, |X_k| = N_k$, and $|X| = \mathrm{N}$. Each classifier classifies all training samples, $\mathbf{x_i} \in X_k, \forall i = 1, 2, \ldots, N_k$, and $k = 1, 2, \ldots, c$. The fitness function measures *how well* it performs the classification task.

We consider the following fitness function to compute the *fitness* of the lth classifier of the population

$$f_l = \sum_{k=1}^{c} \sum_{i=1}^{N_k} \sum_{j=1}^{c} h^l{}_{k,i,j}. \tag{1}$$

The two outer summations in (1) are used to consider all data points in X to compute the fitness of a classifier. $h^l{}_{k,i,j}$ is the contribution of jth tree of lth classifier $(T^l{}_j)$ for the ith data point $(\mathbf{x_i})$ of X_k. So for a training point $\mathbf{x_i} \in X_k$ we expect $T^l{}_k(\mathbf{x_i}) \geq 0$ and $T^l{}_{j \neq k}(\mathbf{x_i}) < 0$. Consequently, both $T^l{}_k(\mathbf{x_i}) \geq 0$ and $T^l{}_{j \neq k}(\mathbf{x_i}) < 0$ should increase the fitness function. To achieve this, we define

$$\begin{aligned} h^l{}_{k,i,k} &= \frac{N - N_k}{N_k} \text{ if } T^l{}_k(\mathbf{x_i}) \geq 0 \\ &\quad 0 \text{ if } T^l{}_k(\mathbf{x_i}) < 0 \\ h^l{}_{k,i,j \neq k} &= 0 \text{ if } T^l{}_{j \neq k}(\mathbf{x_i}) \geq 0 \\ &= 1 \text{ if } T^l{}_{j \neq k}(\mathbf{x_i}) < 0. \end{aligned}$$

Usually, the number of training samples in the kth class, N_k, is smaller than the total number of training samples of classes other than the kth class, $N - N_k$. The tree $T^l{}_k$ learns to discriminate between two classes, class k and all other classes taken together. Since the sizes of these two classes are highly unbalanced and the GP attempts to minimize the number of misclassifications, we give more importance to correct classification to class k. Let us illustrate it with an example. Consider a 4-class problem in which the number of training samples belonging to each class is 50. Thus, for the kth class, $N_k = 50$ and $N - N_k = 150$. That means, tree $T^l{}_k$ is trained to discriminate between 50 training samples of kth class and 150 training samples of the remaining classes. For this two class problem, we have more samples from the second class. To reduce the effect of the unbalanced sample size, we assign more weight to the correct classification of training samples $\mathbf{x_i} \in$ class k by the tree T_k. This is equivalent to copying each point in the training set of class k by $(N - N_k)/N_k$ times [39]. Thus in the stated 4-class problem, if a data point from class k is correctly classified by tree $T^i{}_k$, then the fitness is increased by $3(= (150/50))$.

If the identity of the classifier is not important, then for clarity we will ignore the superscript. So, a sample $\mathbf{x} \in X_k$ is correctly classified by the tree T_k, if $T_k(\mathbf{x}) \geq 0$ and it is correctly classified by tree $T_{j(\neq k)}$, if $T_j(\mathbf{x}) < 0$. A *classifier* correctly classifies a sample $\mathbf{x}$ if and only if all of its constituent trees correctly classify $\mathbf{x}$.

C. Selection

We select a classifier based on its fitness value [computed using (1)] for the genetic operations: reproduction, crossover and mutation. We use fitness-proportion selection scheme for reproduction operation, tournament selection scheme for crossover operation and random selection scheme for mutation operation.

A classifier is good only if all its constituent trees are good. If some of its trees are not able to classify properly, then those trees should be given more preference to take part in crossover and mutation operations for their improvement. To accomplish this, we proceed as described below.

After selecting a classifier for crossover or mutation operation, we assign a probability p_i to each tree T_i in proportion to its *unfitness*. p_i is computed as follows:

Out of N training samples, the number of training samples not correctly classified by the tree T_i is counted. Let it be k_i, $i = 1, 2, \ldots, c$. The p_i is defined as

$$p_i = \frac{k_i}{\sum_{j=1}^{c} k_j}. \tag{2}$$

This p_i is used as the probability of ith tree to be selected by the Roulette-wheel selection for crossover and mutation operations.

D. Crossover and Mutation Operations

As we give more preference to unfit trees to take part in crossover and mutation operations, the chance of evolution of weak trees of potential classifiers is increased and the chance of unwanted disruption of already fit trees is reduced. The detailed steps of our crossover operation, *Crossover_mt* and mutation operation, *mutation_mt* are given in [38]. Since the same mutation algorithm will be used here, we describe *mutation_mt* in Algorithm 1.

E. Termination of GP

The GP is terminated when all N training samples are classified correctly by a classifier (an individual of the GP population) or a predefined number, M, of generations are completed. The best classifier of the population is the required classifier, CF.

F. Interpretation of Outputs and Conflict Resolution

If only one tree of $\mathrm{CF}(\mathbf{x})$ shows a positive response then $\mathbf{x}$ is assigned the class associated with that tree. If all trees exhibit negative responses, i.e., for each tree $T_i(\mathbf{x}) < 0$, then no class is assigned to $\mathbf{x}$. This may be considered a distinct advantage over assigning a wrong class to the pattern. The classifier is able to say—I don't know—when it faces a very unfamiliar pattern $\mathbf{x}$. Moreover, if there are too many such *don't know* cases then it can be treated as an indicator of the fact that the training set is not a proper representative of the population or the designed classifier is a poor one. In this situation, we may redesign the classifier. If more than one tree of $\mathrm{CF}(\mathbf{x})$ show positive responses, then we resolve the conflict by the following weight based scheme.

1) Weight Based Scheme: For each tree T_i of the best classifier CF, we compute the total number (g_i) of correctly classified samples. The ratio $w_i = g_i/N$ is assigned as the weight of the tree T_i.

If more than one tree show positive responses for a pattern $\mathbf{x}$, then $k = \arg\max_i \{w_i\}$ is assigned to pattern $\mathbf{x}$. That means, the class corresponding to the tree with the largest weight value is assigned. For example, if T_1, T_3, and T_4 show positive responses for a pattern $\mathbf{x}$ and $w_3 > w_4 > w_1$ then class 3 will be assigned to pattern $\mathbf{x}$. However, if all N training samples are correctly classified by CF (i.e., by all trees of CF), then all trees have equal weights and hence there is no need to use the weighting scheme.

G. Validation

We validate CF using a set of test data. For a test data point $\mathbf{x}$, if a conflicting situation arises then we use a weight based scheme as mentioned above to assign a single class to the pattern $\mathbf{x}$. If d_{ts} is the number of correctly classified test data by the classifier CF and N_{ts} is the total number of test data then accuracy (A) is defined as $A = (d_{ts}/N_{ts}) \times 100\%$. Note that, d_{ts} counts only the test data points that are correctly classified. The misclassified and unclassified cases together are treated as not classified correctly by the classifier. We shall use A_{all} and A_s to denote accuracy using all features and using only selected features respectively.

III. Designing Classifiers With Online Feature Selection

For simultaneous FS and classifier design, we use the following additional steps in the proposed $\mathrm{GP}_{\mathrm{mtfs}}$ scheme.

A. Selection of a Feature Subset for Each Chromosome

Let P be the population size. Before initializing each individual(classifier) $C_i, i \in \{1, 2, \ldots, P\}$, a feature subset S_i is randomly chosen from the set of all n available features. The size of the feature subset S_i for each classifier C_i is determined probabilistically. The probability to select a feature subset of size $r(\leq n)$ is taken as

$$p_r = \frac{n-r}{\sum_{j=1}^{c} n-j} = \frac{2}{n-1} - \frac{2r}{n(n-1)}. \tag{3}$$

p_r decreases linearly with increase in r. Note that $\sum_{r=1}^{n} p_r = 1$. We use Roulette wheel selection to determine the size of the feature subset,r, with probability p_r.

After deciding the number of features r, we randomly select r features to construct the feature subset S_i. The classifier is now initialized as discussed earlier with the chosen feature subset S_i instead of all n features. The chosen feature subset S_i may be represented by a vector $\mathbf{v_i}$ whose jth element $v_{ij} = 1$, if the jth feature is present in the selected feature subset, otherwise $v_{ij} = 0$. For example, if the total number of features is five and out of these five features $\{x_1, x_2, x_3, x_4, x_5\}$, if $r = 2$ features, x_1, x_3, are selected to construct a classifier, then $\mathbf{v_i}$ will be (1, 0, 1, 0, 0).

B. Fitness Function

The fitness function is required to assign higher fitness value to the classifier which can classify correctly *more* samples using *fewer* features. Thus, the fitness function is a multiobjective one, which needs to consider both correct classification and number of features used. We use the following fitness function:

$$f_s = f \times \left(1 + a e^{-\frac{r}{n}}\right). \tag{4}$$

In (4), f is the fitness value obtained by (1), r is the cardinality of the feature subset used, n is the total number of features and a is a parameter which determines the relative importance that we want to assign for correct classification and the size of the feature subset.

The factor $e^{-(r/n)}$ decreases exponentially with increase in r and so is the fitness function. Thus, if two classifiers make correct decision on the same number of training points, then the one using fewer features is assigned a higher fitness. We decrease the penalty for using larger feature subset with generations to improve the classification accuracy. So initially we use fewer features, but as learning progresses we give more importance to better classification performance. To achieve this, we decrease a as

$$a = 2a_f \left(1 - \frac{\text{gen}}{M}\right) \tag{5}$$

where a_f is a constant. M is the number of generations GP is evolved, gen is the current generation number. We have taken $a_f = 0.1$.

After each generation, to select the best chromosome, we use the fitness function f_{sg}

$$f_{sg} = f \times \left(1 + a_f e^{-\frac{r}{n}}\right). \tag{6}$$

It is similar to f_s in (4), except that here a_f does not change with generation. It is because the best chromosome of each generation should be compared with the best chromosome of the previous generation to get the best chromosome of the run.

Fig. 2 displays the fitness function (6) with $f = 0.8$, $n = 10$, and $a_f = 0.1$. It shows the effect of r, the size of the feature subset, on the fitness value.

After initialization, the population evolves using genetic operations iteratively.

C. Crossover Operation

We use two crossover operations to suit FS. These are the following.

1) Homogeneous crossover called *Crossover_hg*.
2) Heterogeneous crossover called *Crossover_ht*. The heterogeneous crossover depends on the degree of similarity between two parents.

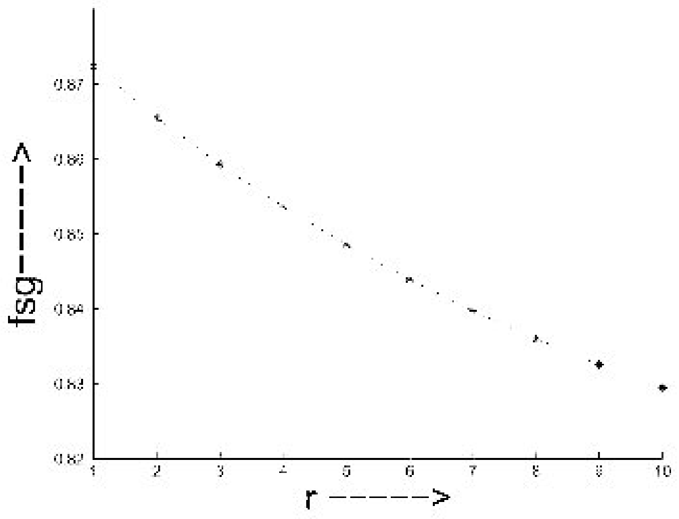

Fig. 2. Decrease of fitness (f_{sg}) value with increase in r for $a_f = 0.1$ and $n = 10$ in (6).

The probability of using homogeneous crossover is computed as

$$P_{hg} = \frac{\text{gen}}{M}. \tag{7}$$

Thus, the probability of using homogeneous crossover increases linearly with generations (gen) from 0 to 1. The *Crossover_ht* is used with probability $P_{ht} = 1 - P_{hg}$. So, P_{ht} decreases linearly from 1 to 0 with an increase in gen.

Crossover_hg restricts the crossover operation between classifiers which use the same feature subset. This crossover operation completely avoids gradual use of more and more features by the classifiers. *Crossover_ht* allows crossover between classifiers using different feature subsets. However, we shall see later it is biased toward crossover between classifiers which use more common features. Consequently, this will check the gradual use of more and more features with generations. As a result of this, with generations the classifiers are not expected to use all features.

1) Homogeneous Crossover, Crossover_hg: This crossover operation restricts the crossover between classifiers which use the same feature subset. After selecting a chromosome as the first parent C_1, we select the second parent C_2 from the group of chromosome that uses the same feature subset as used by C_1. If there is no such chromosome, then we use heterogeneous crossover. The algorithm, *Crossover_hg*, is given in Algorithm 2.

2) Heterogeneous Crossover, Crossover_ht: This is a biased crossover between two classifiers which use more common features. At first, we randomly select a set of τ classifiers. The classifier which has the highest fitness value f_s among this set of classifiers is taken as the first parent C_1 for crossover. After that, we randomly select another set of τ classifiers to select the second parent C_2. The degree of similarity s_j of the jth, $j = 1, 2, \ldots, \tau$, classifier of this second set with C_1 is calculated as

$$s_j = \frac{\sum_{k=1}^{n} v_{0k} v_{jk}}{\text{Max}\left(\sum_{k=1}^{n} v_{0k}, \sum_{k=1}^{n} v_{jk}\right)} = \frac{\mathbf{v_0^T v_j}}{\text{Max}\{\|\mathbf{v_0}\|, \|\mathbf{v_j}\|\}}. \tag{8}$$

In (8), $v_{0k} = 1$, if C_1 uses the kth feature; otherwise $v_{0k} = 0$. Similarly, $v_{jk} = 1$, if jth chromosome of the second set uses the kth feature; otherwise $v_{jk} = 0$.

Clearly, s_j lies in [0, 1]. $s_j = 0$ means jth classifier uses a completely different feature subset than that by C_1 and $s_j = 1$ implies that the jth classifier uses the same subset of features as that by C_1. The jth classifier of the second set is selected as C_2 with probability proportional to m_j

$$m_j = f_{sj} - \beta s_j. \tag{9}$$

Here, f_{sj} = fitness of the jth classifier, β = a constant that controls the effect of similarity. We have taken $\beta = 0.2$. In (9), the factor βs_j is responsible for crossover operation between classifiers using more common features.

To start with, since we randomly generate classifiers using different(randomly taken) feature subsets, the population uses a number of different feature subsets. If we start with homogeneous crossover, then for large dimensional data, the computation cost to search homogeneous group of classifiers will be more. Also, as discussed earlier, there is a chance that after selecting a classifier (C_1) for homogeneous crossover, we may not get another classifier (C_2) which uses the same feature subset as used by C_1. Moreover, since there are $(2^n - 1)$ possible feature subsets, for large n it is not possible to take all feature subsets to construct classifiers. So during the initial period of learning, we use *Crossover_ht* which may change some feature subsets and there by allowing GP to examine some new feature subsets. As the learning progresses, we decrease the probability of using *Crossover_ht* (and hence increase the probability of using *Crossover_hg*). As a result, with generations the good feature subsets should dominate and the population is expected to use a small number of feature subsets. In our scheme, usually FS is accomplished in the first few generations, so we do not use step-wise learning as done in [38]. Because step-wise learning uses a small subset of training samples at the beginning of the evolution and with a small subset of training samples the FS process may not be very fruitful.

Mutation, Termination criteria, conflict resolution, validation of classifier CF remain the same as discussed earlier. The complete algorithm, *Classifier*, is given in Algorithm 3 for a better understanding.

IV. Experimental Results

We have used seven data sets for validating our methodology. These data sets, named, Iris [41], WBC [42], Wine [42], Vehicle [42], WDBC [43], Sonar [44], [42], and GENE [45], [46] cover examples of small, medium and large dimensional data. Table I summerizes these data sets.

A. Data Sets

1) Iris: This is the well-known Anderson's Iris data set [41]. It is a set of measurements in four dimension taken on 150 Iris flowers, 50 each from three different species or classes. The four features are sepal length, sepal width, petal length and petal width.

2) WBC: This Wisconsin breast cancer [42] data set consists of 699 samples in 9-dimension distributed in two classes (malignant and benign). We removed 16 instances with missing values and considered the remaining 683 data points for classification.

TABLE I
Data Sets

Name of Data Set	No of classes	No of Features	Size of Data set
Iris	3	4	150 (50+50+50)
WBC	2	9	683 (444+239)
Wine	3	13	178 (59+71+48)
Vehicle	4	18	846 (212+217+218+199)
WDBC	2	30	569 (357+212)
Sonar	2	60	208 (97+111)
GENE	2	7129	72 (47+25)

3) Wine: Wine data set [42] consists of 178 points in 13-dimension distributed in three classes. These data are the results of chemical analysis of wines grown in a particular region of Italy but derived from three different cultivators. The analysis determined the quantities of 13 constituents found in each of the three types of wine.

4) Vehicle: This data set [42] has 846 data points distributed in four classes. Each data point is represented by 18 attributes. We have normalized the attribute values.

5) WDBC: This Wisconsin Diagnostic Breast Cancer (WDBC) [43] data set contains observations on 569 patients with either Malignant or Benign breast tumor. Each data point consists of 30 features, which are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features describe characteristics of the cell nuclei present in the digitized image. Out of 569 samples, 357 belong to Malignant and remaining 212 samples belong to Benign classes.

6) Sonar: This data set [42], [44] contains 208 patterns obtained by bouncing sonar signals off a *metal cylinder* and *rocks* at various angles and under various conditions. Each pattern is represented by 60 attributes. Each attribute represents the energy within a particular frequency band.

7) GENE: GENE [45], [46] is a set of *DNA microarray gene expression* levels of 72 persons having either acute lymphoblastic leukemia (ALL) or acute myeloid leukemia (AML). Each sample is represented by 7129 gene expressions. 38 samples are used for the training and the remaining 34 samples are used for the testing. Out of 38 training samples, 27 belong to ALL (class 1) and remaining 11 samples belong to AML (class 2). The test set consists of 20 ALL and 14 AML samples.

B. GP Parameters

The GP parameters which are common for all data sets are given in Table II and the GP population size(P) which differs with data sets are listed in Table III.

Note that, the parameters in Tables II and III are not specific to our algorithm. These parameters are required for any GP based classifier design.

We consider larger populations for higher dimensional data because the number of possible feature subsets increases with the number of features (dimension). Use of a large population

TABLE II
COMMON PARAMETERS FOR ALL DATA SETS

Parameters	Values
Probability of crossover operation, p_c	0.80
Probability of reproduction operation, p_r	0.05
Probability of mutation operation, p_m	0.15
Probability of selecting a function node during crossover operation, q_{fc}	0.8
Probability of selecting a terminal node during crossover operation, q_{tc}	0.2
Probability of selecting a function node during mutation operation, q_{fm}	0.7
Probability of selecting a terminal node during mutation operation, q_{tm}	0.3
Tournament size, τ	10
Total number of generations the GP is evolved, M	30
Initial height of a tree	2-6
Maximum allowed nodes for a tree, n_n	350
Maximum allowed height of a tree, n_h	12

TABLE III
POPULATION SIZE

Data set	IRIS	WBC	Wine	Vehicle	WDBC	Sonar	GENE
P	1000	1000	1500	1500	2000	2000	5000

helps GP to explore more possibilities and hence one can expect GP to evolve to a good solution without using many generations. If we allow large number of generations and large size of the classifiers then the classifiers may overfit(memorize) on training samples. Consequently, the classifiers may give better performance on the training data but poor performance on the test data. Choosing the optimal population size for a given problem is a difficult task and we do not study this problem here. The optimal number of generations can be determined using a validation set. To achieve a better generalization one should choose small values of M, n_h, and n_n. Here we allow GP to evolve only upto 30 generations for all data sets.

C. Results

We performed our experiments using lilgp [47] on Alpha server DS10. During classification by the classifier CF, we have used weight scheme for conflict resolution. We run GP ten times for each data set. Except for the GENE data set, we use the following computational protocol:

Each GP run involves a tenfold cross-validation (Thus, each GP run consists ten runs, each with one of the ten folds, so total 100 runs). We compare the average performance of the classifiers obtained by the proposed method $(\mathrm{GP}_{\mathrm{mtfs}})$ using a set of selected features with the classifiers designed using *all* features (using $\mathrm{GP}_{\mathrm{mt}}$) as described in Section II. To do this, we also run GP ten times using $\mathrm{GP}_{\mathrm{mt}}$ method on each data set. Here also we do a tenfold cross-validation.

In case of GENE data, we do not use tenfold validation to keep the results consistent with other results reported in the literature on this data set. For this data set, as used by other authors, we use a specific training and test partition as mentioned in Section IV-A7. We use the following notations in the subsequent sections.

A_s is the average percentage of correctly classified test samples by the best classifiers (CF of each run) using the selected feature subset over ten GP runs, A_{all} is the average percentage of correctly classified test samples by the best classifiers using all features (without any FS) over ten GP runs, n is the total number of features, n_s is the average number of selected features.

TABLE IV
AVERAGE PERFORMANCE

Methods	With all features		With selected features	
	n	A_{all} (%)	n_s	A_s (%)
Iris	4	98.65	1.56	98.69
WBC	9	97.42	2.23	96.84
Wine	13	95.49	4.08	94.82
Vehicle	18	78.37	5.37	78.45
WDBC	30	97.26	6.72	96.31
Sonar	60	84.74	9.45	86.26

TABLE V
MEAN RUN TIME

Data Sets	Iris	WBC	Wine	Vehicle	WDBC	Sonar	GENE
Time(min:sec)	0:30	2:20	1:10	7:30	4:10	2:30	15:40

The average classification (test) accuracy A_{all} with all features and A_s with the selected features, and the average number of selected features n_s over ten GP runs for six data sets are given in Table IV. The mean run time for each data set is shown in Table V. The mean run time includes time taken for partitioning the data set, evolving GP for simultaneous classifier design and FS, input/output file handling, post processing, and validation. Table V shows that *run time* does not necessarily increase with increase in dimension of data sets. Because, the run time also depends on factors like size of population P, number of generations M, size of classifiers, number of training samples N and the distribution of the training samples in the feature space. If all training samples are correctly classified by the best classifier during the GP run, then the GP run terminates before completion of M generations and hence it reduces the run time.

Since different feature subsets are selected in different runs, to give an idea about the importance of the features, we count the frequency of the selected features. The normalized values of those frequencies are given in Table X for Iris, WBC and Wine data sets and in Table XI for Vehicle data. These values may be (roughly) used for ranking the features, although, that is not our objective.

In [40], results of three FS algorithms, Forward Sequential Selection (FSS), Backward Sequential Selection (BSS), and relevance in context (RC), are available. RC uses a clustering like approach to select sets of locally relevant features in the feature space. In [4], results of another three FS algorithms called Information theoretic algorithm (IFN), Relief and ABB are available. Relief is a feature weight based statistical approach. ABB is a breadth first search, backward selection algorithm, with some simple pruning abilities. For comparison, we included results of these six methods on Iris, WBC and Wine data. We use results of ADHOC [28] for comparison of performance on Vehicle data. In addition, we use results of two neural network based FS

TABLE VI
COMPARISON WITH OTHER METHODS FOR IRIS DATA

Methods	FSS	BSS	RC	IFN	Relief	ABB	Fuzzy	GP_{mtfs}
$A_s(\%)$	92.6	92.6	94.4	94.0	96.0	90.0	96.58	98.69
n_f	2.2	2.6	4	1	2	1	3	1.56

TABLE VII
COMPARISON WITH OTHER METHODS FOR WBC DATA

Methods	IFN	Relief	ABB	SNR	NNFS1	NNFS2	GP_{mtfs}
$A_s(\%)$	94.0	93.6	93.6	92.53	94.15	95.77	96.84
n_s	3	2	3	1	2.7	2	2.23

schemes NNFS1 and NNFS2 given in [21] and [22] respectively and a Signal-to-Noise ratio based FS algorithm, SNRFS [48] to compare our results on WBC data. NNFS1 uses cross-validation classification errors to select features. NNFS2 uses a network pruning algorithm to remove redundant and irrelevant attributes one by one. We compare the performance of our scheme on Sonar data with results of four mutual information (MIF) based schemes [5], NNFS2 [22] and ADHOC [28].

Many methods have been used to analyze the data set GENE. Golub *et al.* [46] proposed a neighborhood analysis method to analyze it. Furey *et al.* [49] applied signal-to-noise ratio (SNR) criteria for FS and used support vector machine (SVM) for classification. Ben-Dor *et al.* [50] used a nearest neighbor method, support vector machine with quadratic kernel, and AdaBoost for classification. Principal component analysis was used by Nguyen *et al.* [51] to extract features. Then they used linear and quadratic discriminant analysis for classification. Cho and Ryu [45] considered various criteria for FS and applied multilayer perceptron (MLP) network, k-NN, SVM, self-organizing map (SOM) and decision tree to classify the samples. Cho and Ryu [45] also used ensembles of these techniques for classification.

Rowland [52] used Genetic Programming to classify this gene expression data. Rowland partitioned the training data into *train* and *validation* sets. GP was run 15 times on the *train* set and the best classifier of each run was used to classify the *validation* data. The three best classifiers which produced the minimum absolute difference between the errors on the *train* and *validation* sets, were combined by a voting scheme to classify the test samples.

1) Iris: From Table IV, we find that for Iris data, on average over ten GP runs, the best classifiers could correctly classify 98.69% test data using on average only 1.56 features. This clearly indicates that the selected features have very good discriminating power. Table VI shows that our method performs better compared to several other methods.

In most cases (GP runs), we observed that our methodology selected feature subsets $\{3\}, \{4\}$, and $\{3, 4\}$. Note that, $\{3, 4\}$ means third and fourth features. From these observations, it can be concluded that the third feature (i.e., Petal length) and fourth feature (i.e., Petal width) have good discriminating power for classification. This finding is consistent with feature analysis results represented by other researchers [36], [20]. The weights of the features are given in Table X.

As an illustration, we show the expressions corresponding to the three trees of the best classifier of a typical run:

TREE1: $x_3 - 4.8$
TREE2: $(x_3 - 2)(4.9 - x_3)$
TREE3: $2.2 - x_3$

where x_3 is the third feature.

TABLE VIII
COMPARISON FOR WINE AND VEHICLE DATA

Data	Wine				Vehicle	
Methods	IFN	Relief	ABB	GP_{mtfs}	ADHOC	GP_{mtfs}
A_c	91.7	95.0	79.0	94.82	69.6	78.45
n_c	3	3	2	4.08	7	5.37

TABLE IX
COMPARISON WITH OTHER METHODS FOR SONAR DATA

Methods	ADHOC	4 MIF schemes	NNFS1	GP_{mtfs}
$A_s(\%)$	76	76.45-81.70	93.81	86.26
n_e	16	9	3.87	9.45

After analyzing the above expressions, for a pattern $\mathbf{x}$, we find the following decision rules for the three classes.

If $x_3 \geq 4.8$ then $\mathbf{x} \in \text{Class } 1$.
If $x_3 \in [2, 4.9]$ then $\mathbf{x} \in \text{Class } 2$.
If $x_3 < 2.2$ then $\mathbf{x} \in \text{Class } 3$.

These rules suggest that we need techniques like weight based method to make decisions in the overlapping regions [2.0, 2.2] and [4.8, 4.9]. Note that, the best GP classifier may not always produce such simple rules.

2) WBC: For this data set, on average our method constructed classifiers, selecting only 2.23 features. Our method selected features 6, 2, and 3 in most of the runs. With these selected features we could achieve 96.84% test accuracy. Table IV shows the average performance.

Table VII compares our method with six other methods. In this case too, the proposed method outperforms other methods. The weight (the normalized frequency with which each feature is selected) computed for each feature is given in Table X. Table X reveals that features 2, 3, and 6 are very important.

To demonstrate the decision rules for WBC, we show one of the best outputs (best in terms of simplicity of expressions) corresponding to a GP run.

TREE1: $3.84 \quad x_2$.
TREE2: $x_2 - 3.47$.

where x_2 is the second feature. We know that for the WBC data set x_2 is an integer valued feature. So, after analyzing the above expressions, we can write the following simple decision rule:

If $x_2 \leq 3$ **then**
$\quad \mathbf{x} \in \text{Class } 1$
else
$\quad \mathbf{x} \in \text{Class } 2$.

TABLE X
WEIGHTS OF FEATURES FOR IRIS, WBC, AND WINE DATA

Features	1	2	3	4	5	6	7	8	9	10	11	12	13
Iris	0.15	0.07	0.42	0.36									
WBC	0.031	0.232	0.182	0.050	0.052	0.260	0.061	0.110	0.022				
Wine	0.108	0.044	0.063	0.056	0.054	0.092	0.103	0.043	0.066	0.118	0.043	0.091	0.119

TABLE XI
WEIGHTS OF FEATURES FOR VEHICLE DATA

Features	7	13	10	11	8	12	16	9	14
Weights	0.142	0.112	0.108	0.104	0.101	0.098	0.094	0.077	0.073

This is probably one of the simplest rule that can do the classification task with a reasonably good accuracy of 92.39%.

3) Wine: For wine data set also the average number of selected features and the classifier performance over all runs are given in Table IV. It shows that on average with only 4.08 selected features, GP is able to construct classifiers with 94.82% accuracy. The computed importance (weights) of the features are included in Table XI. Table VIII compares our method with three other methods [4]. In this case GP_{mtfs} outperforms IFN and ABB but Relief is marginally better than GP_{mtfs}.

4) Vehicle: For Vehicle data, on average, 5.37 features are selected (used) by the classifier. With about five features 78.45% test points are classified correctly as given in Table IV. The ADHOC [28] method reported a test accuracy of 69.6% using seven features while our method could achieve 78.45% test accuracy using a lesser number of features as shown in Table VIII. The classifier designed with all features classifies 78.37% of test data.

The weights of some of the features are listed in the Table XI, the rest are not included as they have negligible weights.

5) WDBC: The average performance of the best classifiers over all ten runs shows that with only six to seven features, 96.31% test data can be classified correctly (Table IV). We do not include the feature weights for WDBC in Table X because the number of features is too large to accommodate in Table X. However, the most important seven features are 28, 8, 21, 10, 13, 15, and 23.

6) Sonar: On average, we achieved 86.26% accuracy with about nine to ten features (Table IV). While using all 60 features, the average performance is 84.74%. This again emphasizes that more features are not necessarily good. Table IX shows that GP_{mtfs} performs much better than ADHOC and the four mutual information based selection schemes [5]. For this data set, NNFS1 outperforms our method both in terms of accuracy and number of features used.

7) GENE: For this data set, the average accuracy on the test data achieved by our method is 92.55% using on average 10.45 features out of 7129 features. In one of the GP runs, the best classifier could classify correctly 33 of the 34 test samples (97.1% accuracy) using 13 features. Note that, during the generation of the initial population, the algorithm may select a large number of features for some classifiers, but since the depth of initial trees are restricted to 2–6, most of the selected features will not be used. Consequently, GP will evolve using only a small number of features, which is further moderated by our crossover operations. In [46], Golub *et al.* reported correct classification of 29 test samples with high confidence. Nguyen *et al.* [51], [45] achieved 97.1% (best test) accuracy using the logistic discriminant classifier. On the other hand, the test accuracy of the classifiers studied by Cho and Ryu [45] varied from 58.8% to 100%. Rowland [52] obtained 91.1% test accuracy using Genetic Programming.

TABLE XII
AVERAGE PERFORMANCE FOR NOISY DATA

Methods	With all features		With selected features	
	n	A_{all} (%)	n_s	A_s (%)
WBC	13	94.54	3.46	95.92
Wine	17	93.86	5.27	94.33
Vehicle	22	76.32	6.12	78.16

Our experiments on Sonar and GENE suggest that the proposed method can easily do a good job for moderately, large and very large dimensional data sets also.

D. Effect of Noise Features

Here we study the sensitivity of our method on bad (noise) features that are synthetically injected to a few real data sets. To produce noisy data, we add four randomly generated values, x_{n1}, x_{n2}, x_{n3}, and x_{n4}, to each data point of WBC, Wine and Vehicle data. We generate the values of x_{n1} and x_{n2} in [0, 1] and $[-1, 1]$ respectively. To decide on the domains of x_{n3} and x_{n4}, we randomly select two features of the data set and use their domains. Once the domains are decided we randomly generate values of x_{n3} and x_{n4} from their respective domains. For example, if the domain of a selected feature is [15.5, 30.0], then for each data point we use a random number generator to obtain a value in [15.5, 30.0] and add it as a noisy feature value. To see the effect of the noise features we run our scheme with FS (GP_{mtfs}) and without FS (GP_{mt}) on the augmented noisy data sets. We run GP 5 times with twofold cross-validation using both the schemes. The average accuracies for WBC, Wine and Vehicle data are given in Table XII. The performance of our scheme on data sets with noise features almost remains the same as that on data sets without noise.

We have observed that the proposed FS scheme GP_{mtfs} only occasionally chooses any noise feature, but on average there is

TABLE XIII
$\tilde{t}$ STATISTIC FOR THE NOISY DATA SETS

Data set	WBC	Wine	Vehicle
$\tilde{t}$	2.122	2.138	2.104

a marginal increase in the number of features used. Addition of noise features causes confusion in feature space. This may lead the classifier to select more nonnoise features to counter balance the complexity in feature space caused by noise features. Moreover, if a noise feature gets selected, the system may need a few good features to balance its influence. Hence, there is a marginal increase in the number of selected features. An interesting observation is that even without FS, our scheme can find reasonably good classifiers from the noisy data. This is a very good attribute of the proposed GP-based classifier design scheme.

In addition to this comparison of average performances, we also compare the performance of these two methods over noisy data using 5×2 cross-validation paired t test [53]. Let μ_1 and μ_2 be the means of the test errors using GP_{m}, and $\mathrm{GP}_{\mathrm{mfs}}$, respectively. Let the null hypothesis be H_0: $\mu_1 = \mu_2$ and the alternative hypothesis be H_1: $\mu_1 > \mu_2$. The computed t values, $\tilde{t}$ for WBC, Wine, and Vehicle data are given in Table XIII. The cut-off value for rejecting the null hypothesis at 95% confidence level for t with five degrees of freedom [53] is 2.015. Table XIII shows that the $\tilde{t}$ value for each of these 3 data sets is greater than this critical value of t. So the null hypothesis is rejected at 95% confidence level.

Algorithm 1 *Mutation mt(Population)*

```
1: Randomly pick a classifier(C) $\subset$ *Population*
2: Compute $p_j$ of C,$\forall j = 1, 2, \cdots, c$
3: **repeat** $c$ times
4: Select a tree ($T_i$) of C with Roulette wheel selection using
   $p_i$
5: $m_1 = \frac{m}{100} \times$ total number of nodes of $T_i$ { m% of the
   total nodes, let m = 2}
6: **for** $k = 0$ to $m_1$ **do**
7:    Choose a node type *ntype*{with probability $q_{fm}$ to
      select *function* node type and $q_{tm}$ to select *terminal*
      node type}
8:    Randomly select an *ntype* node of the tree $T_i$
9:    Replace it with a randomly chosen *ntype* node
10: **end for**
11: $k_m = 0$, $k_o = 0$,{$N_i$ = number of training samples of $i_{th}$
    class}
12: **for** $j = 1$ to $N_i$ **do**
13:    **if** $T_i^m(\mathbf{x}_{ij}) \geq 0$ **then**
14:       $k_m = k_m + 1$ { $T_i^m$ = mutated $T_i$}
15:    **end if**
16:    **if** $T_i(\mathbf{x}_{ij}) \geq 0$ **then**
17:       $k_o = k_o + 1$ {$\mathbf{x}_{ij}$ is $j_{th}$ training sample of $i_{th}$ class}
18:    **end if**
19: **end for**
20: **if** $k_m \geq k_o$ **then**
21:    Accept $T_i^m$
22: **else**
23:    Retain $T_i$ with probability 0.5
24: **end if**
25: **end repeat**
```

V. CONCLUSION

We proposed a methodology for online FS and classifier design using GP. In a *single* run of GP our method automatically selects the required features while designing classifiers for a multicategory classification problem.

We generated the initial population in such a manner that the classifiers use different feature subsets. The initialization process generates classifiers using smaller feature subsets with higher probability. The fitness function assigns higher fitness values to classifiers which classify more samples using fewer features. The multiobjective fitness function helps to accomplish both FS and classifier design simultaneously. In this regard, we proposed two modified crossover operations suitable for FS. As a byproduct, we obtained a feature ranking method.

The effectiveness of our scheme is demonstrated on seven data sets having dimensionality varying between 4 and 7129. Our experimental results established that the proposed scheme is very effective in selecting a small set of features and finding useful classifiers. The obtained results (Table IV) reveal that the proposed method can achieve almost the same performance using a small set of features as that with all features. We have also demonstrated the effectiveness of our scheme on data sets with known redundant or bad features added synthetically. We have compared the performance of our methodology with results available in the literature with both filter and wrapper type approaches. Wrapper (and online) FS algorithms perform better than filter approaches, at the cost of computational time. Our proposed algorithm performed better for both two class and multiclass problems.

Some of the important characteristics of the proposed algorithm are as follows.

- It provides a mathematical description of the classifier which is easy to analyze and interpret.
- The FS process is integrated into classifier design in such a manner that the algorithm can pick up the required features and obtain the classifier using them.
- The fitness function prevents the use of more features and hence helps to achieve more readability of the trees extracted by the system.
- Since the number of features to be used is not predefined, the algorithm has more flexibility.
- Using the output of the algorithm, we can obtain a ranking of the features.
- It can deal with a c-class ($c \geq 2$) problem in a single run of GP.

We have used arithmetic functions to design classifiers. So, our methodology is applicable to numerical attributes only. For data with nominal attributes, the logical functions like AND, OR, NOT may be considered instead of arithmetic functions.

Algorithm 2 *Crossover_hg (Population)*

```
1:  $m_\tau = 0, m_1 = 1, Max_1 = 0, Max_2 = 0$
2:  for $l = 1$ to $\tau$ do
3:      Randomly take a classifier $C_j \in Population$
4:      if $f_s(C_j) \geq Max_1$ then
5:          $Max_1 = f_s(C_j),\ k = j$
6:      end if
7:  end for
8:  $P_1 = C_k$ {first parent for crossover}
9:  $C_i = C_{k+1}$ { now for second parent $P_2$}
10: while $((m_1 < P)$ and $(m_\tau < \tau))$ do
11:     if $v(C_i) = v(P_1)$ then
12:         if $f_s(C_i) \geq Max_2$ then
13:             $Max_2 = f_s(C_i),\ P_2 = C_i$ {if both use same feature subset}
14:         end if
15:         $m_\tau = m_\tau + 1$
16:     else
17:         $C_i = C_{i+1}$ {if $i = P$, then $i + 1 = 1$}
18:     end if
19:     $m_1 = m_1 + 1$
20: end while
21: if $m_\tau > 0$ then
22:     Compute $p_i^1, i = 1, 2, \cdots, c$ of $P_1$
23:     Select a tree $T_l^1$ of $P_1$, by the Roulette-wheel selection using $p_i^1$
24:     Select a node type ntype {with probability $q_{fc}$ to select function node type and $q_{tc}$ to select terminal node type}
25:     Randomly select a node of the ntype type from $T_l^1$ and $T_l^2$ (of $P_2$) independently
26:     Swap the subtrees rooted at the selected nodes of $T_l^1$ and $T_l^2$
27:     Swap $T_j^1$ with $T_j^2, \forall\ j = l+1, ..., c$
28: else
29:     Do heterogeneous crossover {as mentioned in the section III(C) }
30: end if
```

Acknowledgment

The authors express their thanks to the associate editor and anonymous referees whose valuable suggestions helped to improve this work significantly and also to their colleague Dr. A. Sharma for his help in bioinformatics-related work.

References

[1] M. Dash and H. Liu, "Feature selection for classification," *Intell. Data Anal.*, vol. 1, no. 3, pp. 131–156, 1997.

[2] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithms," in *Proc. 9th Nat. Conf. Artificial Intelligence*, 1992, pp. 129–134.

[3] P. M. Narendra and K. Fukunaga, "A branch and bound algorithm for feature selection," *IEEE Trans. Comput.*, vol. C-26, no. 9, pp. 917–922, Sep. 1977.

[4] M. Last, A. Kandel, and O. Maimon, "Information-theoretic algorithm for feature selection," *Patt. Recognit. Lett.*, vol. 22, pp. 799–811, 2001.

[5] N. Kwak and C.-H. Choi, "Input feature selection for classification problems," *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 143–159, Jan. 2002.

Algorithm 3 *Classifier*

```
1:  gen = 0, $fit_g = 0$, $fit_\tau = 0$ { $fit_g$ = highest fitness value($f_{sg}$) of that generation and $fit_\tau$ =highest fitness($f_{sg}$) till that generation}
2:  Initialize population of classifiers $\{C_j\}, \forall j = 1, 2, \cdots, P$ {max. possible fitness $f_{max} = 1 + a_f \times exp(-1/n)$}
3:  while gen < M and $fit_\tau < f_{max}$ do
4:      for $j = 1$ to $P$ do
5:          Evaluate fitness $f_s$ and $f_{sg}$ of each classifier $C_j$
6:          $fit_g = max\ f_{sg}(C_j)$
7:          k = arg $max_j\ f_{sg}(C_j)$
8:      end for
9:      if $fit_g > fit_\tau$ then
10:         $fit_\tau = fit_g$
11:         CF = $C_k$
12:     end if
13:     Perform Breeding {all genetic operations}
14:     gen = gen + 1 {go to the next generation}
15: end while
16: Compute weights $\{w_i\}$ {of the best classifier CF}
```

Algorithm 4 *Breeding(Population)*

```
1:  $P_n = 0$
2:  while $P_n \neq P$ do
3:      Select one of the operators from reproduction, crossover and mutation with a probability $p_r, p_c$ and $p_\mu$ respectively
4:      if operator = reproduction then
5:          Select a classifier using fitness-proportion selection method
6:          perform the reproduction operation
7:          $P_n = P_n + 1$
8:      end if
9:      if operator = crossover then
10:         Select two classifiers using tournament selection method
11:         perform crossover
12:         $P_n = P_n + 2$
13:     end if{Note that, if $P_n$ becomes $P + 1$, then reject the second offspring after crossover operation}
14:     if operator = mutation then
15:         Select a classifier randomly
16:         perform mutation
17:         $P_n = P_n + 1$
18:     end if
19: end while
```

[6] A. N. Mucciardi and E. E. Gose, "A comparison of seven techniques for choosing subsets of pattern recognition," *IEEE Trans. Comput.*, vol. C-20, pp. 1023–1031, Sep. 1971.

[7] M. Dash and H. Liu, "Consistency-based search in feature selection," *Artif. Intell.*, vol. 151, pp. 155–176, 2003.

[8] G. V. Lashkia and L. Anthony, "Relevant, irredundant feature selection and noisy example elimination," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 2, pp. 888–897, Apr. 2004.

[9] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artif. Intell.*, vol. 97, pp. 273–324, 1997.

[10] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell. Special Issue on Relevance*, vol. 97, pp. 245–271, 1997.

[11] M. Kudo and J. Sklansky, "Comparison of algorithms that select features for pattern classifiers," *Patt. Recognit.*, vol. 33, pp. 25–41, 2000.

[12] N. R. Pal, "Soft computing for feature analysis," *Fuzzy Sets and Syst.*, vol. 103, pp. 201–221, 1999.

[13] I. Guyon and A. Elisseeff, "An introduction to variables and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1157–1182, 2003.

[14] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 153–158, Feb. 1997.

[15] R. K. De, N. R. Pal, and S. K. Pal, "Feature analysis: Neural network and fuzzy set theoretic approaches," *Patt. Recognit.*, vol. 30, pp. 1579–1590, 1997.

[16] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *J. Mach. Res.*, vol. 3, pp. 1289–1305, 2003.

[17] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[18] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Patt. Recognit. Lett.*, vol. 15, pp. 1119–1125, 1994.

[19] H. Zhang and G. Sun, "Feature selection using tabu search method," *Patt. Recognit.*, vol. 35, pp. 701–711, 2002.

[20] N. R. Pal and K. Chintalapudi, "A connectionist system for feature selection," *Neural, Parallel, and Sci. Comput.*, vol. 5, pp. 359–381, 1997.

[21] R. Setiono and H. Liu, "Neural-network feature selector," *IEEE Trans. Neural Netw.*, vol. 8, no. 3, May 1997.

[22] A. Verikas and M. Bacauskiene, "Feature selection with neural networks," *Patt. Recognti. Lett.*, vol. 23, pp. 1323–1335, 2002.

[23] K. Z. Mao, "Feature subset selection for support vector machines through discriminative function pruning analysis," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 1, pp. 60–67, Feb. 2004.

[24] J. Bi, K. P. Bennett, M. Embrechts, C. M. Breneman, and M. Song, "Dimensionality reduction via sparse support vector machines," *J. Mach. Learn. Res.*, vol. 1, pp. 1–48, 2002.

[25] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Patt. Recognit. Lett.*, vol. 10, pp. 335–347, 1989.

[26] ——, "On automatic feature selection," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 2, no. 2, pp. 197–220, 1988.

[27] J. Casillas, O. Cordon, M. J. Del Jesus, and F. Herrera, "Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems," *Inform. Sci.*, vol. 136, pp. 135–157, 2001.

[28] M. Richeldi and P. Lanzi, "Performing effective feature selection by investigating the deep structure of the data," in *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*. Menlo Park, CA, 1996, pp. 379–383.

[29] N. R. Pal, S. Nandi, and M. K. Kundu, "Self-crossover: A new genetic operator and its application to feature selection," *Int. J. Syst. Sci.*, vol. 29, no. 2, pp. 207–212, 1998.

[30] M. Ahluwalia and L. Bull, "Coevolving functions in genetic programming," *J. Syst. Architect.*, vol. 47, no. 7, pp. 573–585, July 2001.

[31] J. Sherrah, R. E. Bogner, and A. Bouzerdoum, "Automatic selection of features for classification using genetic programming," in *Proc. 1996 Australian New Zealand Conf. Intelligent Information Systems*, pp. 284–287.

[32] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[33] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic Programming: An Introduction*. New York: Morgan Kaufmann, 1998.

[34] H. F. Gray, R. J. Maxwell, I. Martinez-Perez, C. Arus, and S. Cerdan, "Genetic programming for classification and feature selection: Analysis of ^{1}H nuclear magnetic resonance spectra from human brain tumor biopsies," *NMR Biomed.*, vol. 11, pp. 217–224, 1998.

[35] H. Guo, L. B. Jack, and A. K. Nandi, "Feature generation using genetic programming with application to fault classification," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, no. 1, pp. 89–99, 2005.

[36] D. Chakraborty and N. R. Pal, "A neuro-fuzzy scheme for simultaneous feature selection and fuzzy rule-based classification," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 110–123, 2004.

[37] ——, "Integrated feature analysis and fuzzy rule-based system identification in a neuro-fuzzy paradigm," *IEEE Trans. Syst., Man, Cybern. B*, vol. 31, no. 3, pp. 391–400, 2001.

[38] D. P. Muni, N. R. Pal, and J. Das, "A novel approach for designing classifiers using genetic programming," *IEEE Trans. Evolut. Comput.*, vol. 8, no. 2, pp. 183–196, Apr. 2004.

[39] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal, "Application of genetic programming for multicategory pattern classification," *IEEE Transaction on Evolutionary Computation*, vol. 4, pp. 242–258, Sept. 2000.

[40] P. Domingos, "Context-sensitive feature selection for lazy learners," *Artif. Intell. Rev.*, vol. 11, pp. 227–253, 1997.

[41] E. Anderson, "The irises of the gaspe peninsula," *Bulletin of the Amer. IRIS Soc.*, vol. 59, pp. 2–5, 1935.

[42] C. L. Blake and C. J. Merz, *UCI Repository of Machine Learning Databases*: Univ. of California, Dept. of Inform. Comput. Sci., 1998.

[43] O. L. Mangasarian, W. N. Street, and W. H. Wolberg, "Breast cancer diagnosis and prognosis via linear programming," *Oper. Res.*, vol. 43, no. 4, pp. 570–577, 1995.

[44] R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Netw.*, vol. 1, pp. 75–89, 1988.

[45] S.-B. Cho and J. Ryu, "Classifying gene expression data of cancer using classifier ensemble with mutually exclusive features," *Proc. IEEE*, vol. 19, no. 11, pp. 1744–1753, Nov. 2002.

[46] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531–537, 1999.

[47] D. Zongker and W. F. Punch. Lilgp 1.0 User's Manual. MSU Genetic Algorithms and Research Application Group (GARAGe). [Online]. Available: http://garage.cps.msu.edu

[48] K. W. Bauer Jr., S. G. Alsing, and K. A. Greene, "Feature screening using signal-to-noise ratios," *Neurocomputing*, vol. 31, pp. 29–44, 2000.

[49] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[50] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, and N. Yakhini, "Tissue classification with gene expression profiles," *J. Comput. Biol.*, vol. 7, pp. 559–584, 2000.

[51] D. V. Nguyen and D. M. Rocke, "Tumor classification by partial least squares using microarray gene expression data," *Bioinformatics*, vol. 18, no. 1, pp. 39–50, 2002.

[52] J. J. Rowland, "Model selection methodology in supervised learning with evolutionary computation," *Biosystems*, vol. 72, pp. 187–196, 2003.

[53] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, 1998.

Durga Prasad Muni received the B.E. degree in electronics and telecommunication engineering from Utkal University, India, in 1997 and the M.E. degree in electronics and communication engineering from the National Institute of Technology, Rourkela, India, in 2000. He is currently pursuing the Ph.D. degree in the Electronics and Communication Sciences Unit at the Indian Statistical Institute, Calcutta.

His research interest includes pattern recognition and evolutionary computation.

Nikhil R. Pal (SM'00–F'02) received the B.Sc. degree (Hons.) in physics and the Master's degree in business management from the University of Calcutta, Calcutta, India, in 1978 and 1982, respectively, and the M. Tech. and Ph.D. degrees in computer science from the Indian Statistical Institute, Calcutta, in 1984 and 1991, respectively.

Currently, he is a Professor in the Electronics and Communication Sciences Unit of the Indian Statistical Institute. He is coauthor of *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing* (Norwell, MA: Kluwer, 1999), editor of *Pattern Recognition in Soft Computing Paradigm* (Singapore: World Scientific, 2001), and has also coedited four volumes. His current research interest includes image processing, pattern recognition, fuzzy sets theory, neural networks, evolutionary computation and bioinformatics.

Dr. Pal serves on the Editorial/Advisory Board of the *International Journal of Approximate Reasoning*, *International Journal of Hybrid Intelligent Systems*, *Neural Information Processing—Letters and Reviews*, *International Journal of Knowledge-Based Intelligent Engineering Systems*, *Iranian Journal of Fuzzy Systems* and *Fuzzy Sets and Systems*. He is an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS–B and the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He was the president of and currently has been serving as a governing board member of the Asia Pacific Neural Net Assembly. He was the Program Chair of the 4th International Conference on Advances in Pattern recognition and Digital Techniques, December 1999, Calcutta. He was the General Chair of 2002 AFSS International Conference on Fuzzy Systems, Calcutta, 2002 and the 11th International Conference on Neural Information Processing, ICONIP 2004. He is a Co-Program Chair of the 2005 IEEE International Conference on Fuzzy Systems and the 2006 IEEE International Conference on Fuzzy Systems.

Jyotirmoy Das received the M.Tech. degree in radio physics and electronics and the Ph.D. degree in computer memory technology from the University of Calcutta, Calcutta, India.

He is a Senior Professor and the Head of Electronics and Communication Sciences Unit of the Indian Statistical Institute, Calcutta. His current research interest includes atmospheric science problems, computational Intelligence, and microwave/millimeterwave propagation.

Dr. Das is a Fellow of the National Science Academy (Allahabad).