ELSEVIER

Pattern Recognition Letters 19 (1998) 1171–1181

Pattern Recognition Letters

Pattern classification using genetic algorithms: Determination of H

S. Bandyopadhyay, C.A. Murthy, Sankar K. Pal *

Indian Statistical Institute, Machine Intelligence Unit, 203 Barrackpore Trunk Road, Calcutta 700 035, India

Received 11 February 1998; received in revised form 21 July 1998

Abstract

A methodology based on the concept of a variable string length GA (VGA) is developed for determining automatically the number of hyperplanes for modeling the class boundaries in a *GA-classifier*. The genetic operators and fitness function are defined to take care of the variability in chromosome length. It is proved that the method is able to arrive at the optimal number of misclassifications after a sufficiently large number of iterations, and will need a minimal number of hyperplanes for this purpose. Experimental results on different artificial and real life data sets demonstrate that the classifier, using the concept of a variable length chromosome, can automatically determine an appropriate value of the number of hyperplanes, and also provide performance better than that of the fixed length version. Its comparison with another approach using a VGA is provided.

Keywords: Genetic algorithms; Optimum hyperplane fitting; Speech recognition; Variable string length

1. Introduction

Genetic Algorithms (GAs) (Goldberg, 1989) are randomized search and optimization techniques guided by the principles of evolution and natural genetics. They are efficient, adaptive and robust search processes, producing near optimal solutions and have a large amount of implicit parallelism. The application of GAs to various pattern recognition problems is described in (Pal and Wang, 1996; Gelsema, 1995). One such application for designing a classifier is provided in (Bandyopadhyay et al., 1995), where the searching capability of GAs is exploited for the placement of a number of hyperplanes, say H, for approximating the decision boundaries. The method involves encoding the parameters of the hyperplanes in binary strings called *chromosomes*, in the feature space that yields minimum misclassification. It was demonstrated in (Bandyopadhyay et al., 1995) that the GA based classifier, subsequently referred to as the *GA-classifier*, can be well applied to a variety of data sets having both non-overlapping, non-convex, and overlapping classes. Its recognition scores were found to be comparable to, sometimes better than, those of the k-NN rule (for different values of k), the Bayes maximum likelihood classifier and a multilayer perceptron based classifier.

Note that the estimation of a proper value of H is crucial for a good performance of the algorithm. Since this is difficult to achieve, one may

* Corresponding author. Tel.: +91 33 577 8085 3100; fax: +91 33 556 6680/6925; e-mail: sankar@isical.ac.in.

0167-8655/98/$ – see front matter
PII: S0167-8655(98)00097-X

frequently use a conservative value of H while designing the classifier. This first of all leads to the problem of an overdependence of the algorithm on the training data, especially for small sample size. In other words, since a large number of hyperplanes can readily and closely fit the classes, this may provide good performance during training but poor generalization capability. Secondly, a large value of H unnecessarily increases the computational load, and may lead to the presence of redundant hyperplanes in the final decision boundary. (A hyperplane is termed redundant if its removal has no effect on the classification capability of the GA-classifier.)

In order to overcome these limitations, a method is described here to automatically determine the value of H as a parameter of the problem. For this purpose, the concept of variable length strings in GAs has been adopted. Unlike the conventional GA, here the length of a string is not fixed. Crossover and mutation operators are accordingly defined. A factor has been incorporated into the fitness function that rewards a string with a smaller number of misclassified samples as well as a smaller number of hyperplanes. Let the classifier so designed utilizing the concept of variable string lengths be called a *VGA-classifier*. Issues of minimum misclassification error and minimum number of required hyperplanes are theoretically analyzed under limiting conditions.

One may note the difference between the proposed classification method and the one described in (Srikanth et al., 1995), also using a similar concept of variable length strings. In the latter method, the decision boundary was modeled by a variable number of ellipsoids which have a higher degree of complexity than hyperplanes. The fitness function of the string was determined from the number of misclassified samples only. Thus there was no incentive for reducing the number of ellipsoids, although a factor favoring more compact ellipsoids was introduced.

Experimental results on speech data, Iris data and two artificially generated data sets show that the proposed classifier is able to reduce the number of hyperplanes significantly, while retaining the classification performance of the previous fixed length GA-classifier. A comparison with the classifier implemented using the operators of Srikanth et al. (1995) is also provided.

2. Genetic algorithm with variable string length and the classification criteria

The concept of variable string lengths in genetic algorithms has been used earlier in (Smith, 1980) to encode sets of fixed length rules. Messy genetic algorithms (Goldberg et al., 1989) also use the concept of variable string lengths for constructing the chromosomes which may be under- or over-specified. Use of GAs with variable string length has been made in (Harp and Samad, 1992) for encoding a variable number of fixed length blocks in order to construct layers of a neural network, and in (Maniezzo, 1994) for the genetic evolution of the topology and weight distribution of neural networks.

As mentioned in Section 1, the GA-classifier (Bandyopadhyay et al., 1995) with fixed H, and consequently fixed string length is rigid, and therefore has several limitations like overfitting of the training data and the presence of redundant hyperplanes in the decision boundary when a conservative value of H is used. To overcome these limitations, the use of variable length strings representing a variable number of hyperplanes for optimally modeling the decision boundary therefore seems natural and appropriate. This would eliminate the need for fixing the value of H, evolving it adaptively instead, thereby providing an optimal value of H.

It is to be noted that in the process, if we aim at reducing the number of misclassified points only, as was the case for fixed length strings, then the algorithm may try to fit as many hyperplanes as possible for this purpose. This, in turn, would obviously be harmful with respect to the generalization capability of the classifier. Thus, the fitness function should be defined in such a way, that maximization ensures primarily the minimization of the number of misclassified samples and also the required number of hyperplanes.

While incorporating the concept of variable string lengths, one may note that it is necessary to either modify the existing genetic operators or to

introduce new ones. In order to utilize the existing operators as much as possible, a new representation scheme involving the consideration of the ternary alphabet set {0, 1, # }, where # represents the don't care position, is used. For applying the conventional crossover operator, the two strings, which may now be of unequal lengths, can be made of equal length by appropriately padding one of them with #s. However, some extra processing steps have to be defined in order to tackle the presence of #s in the strings. Similarly, the mutation operator needs to be suitably modified such that it has sufficient flexibility to change the string length while retaining the flavor of the conventional operator. (As will be evident in Section 3.3, the genetic operators are defined in such a way that the inclusion of # in the strings does not affect their binary characteristics for encoding and decoding purposes.) The classifier thus formed using a variable string length GA (or VGA) is referred to as the VGA-classifier.

Therefore, the objective of the VGA-classifier is to place an appropriate number of hyperplanes in the feature space such that it, first of all, minimizes the number of misclassified samples and then attempts to reduce the number of hyperplanes. Using variable length strings enables one to check automatically and efficiently, various decision boundaries consisting of different numbers of hyperplanes in order to attain the optimal solution. The description of such a classifier is given in Section 3.

```
Begin
        t=0
        initialize population P(t)
        compute fitness P(t)
        repeat
                t = t+1
                select P(t) from P(t-1)
                crossover P(t)
                mutate P(t)
                compute fitness P(t)
        until termination criterion is achieved
End
```

Fig. 1. Basic steps in GA.

3. Description of VGA-classifier

As is evident from the previous section, although the sequence of the different operations for GAs (as shown in Fig. 1) is applicable to VGAs too, the operators themselves are newly defined for VGA. They are described here.

3.1. Chromosome representation and population initialization

The chromosomes are represented by strings of 1, 0 and # (don't care), encoding the parameters of a variable number of hyperplanes. In $\mathbb{R}^N$, N parameters are required for representing one hyperplane. These are $N-1$ angle variables, $\text{angle}_1^i, \ldots, \text{angle}_{N-1}^i$, indicating the orientation of hyperplane i ($i = 1, 2, \ldots, H$ when H hyperplanes are encoded in the chromosome), and one perpendicular distance variable, p^i indicating its perpendicular distance from the origin. Let $H_{\max}$ represent the maximum number of hyperplanes that may be required to model the decision boundary of a given data set. It is specified a priori. Let the angle and perpendicular distance variables be represented by b_1 and b_2 bits, respectively. Then l_H, the number of bits required to represent a hyperplane and $l_{\max}$, the maximum length that a string can have are

$$l_H = (N-1) * b_1 + b_2, \tag{1}$$

$$l_{\max} = H_{\max} * l_H, \tag{2}$$

respectively.

Let string i represent H_i hyperplanes. Then its length l_i is

$$l_i = H_i * l_H.$$

An initial population is created in such a way that the first and the second strings encode the parameters of $H_{\max}$ and 1 hyperplanes, respectively, to ensure sufficient diversity in the population. For the remaining strings, the number of hyperplanes, H_i, is generated randomly in the range [1, $H_{\max}$], and the l_i bits are initialized randomly to 1s and 0s.

3.2. Fitness computation

As mentioned in Section 2, the fitness function (which is maximized) is defined in such a way that

1. a string with a smaller number of misclassifications is considered to be fitter than a string with a larger number, irrespective of the number of hyperplanes, i.e., it first of all minimizes the number of misclassified points, and then
2. among two strings providing the same number of misclassifications, the one with the smaller number of hyperplanes is considered to be fitter.

The number of misclassified points for a string i encoding H_i hyperplanes is found as follows. Let the H_i hyperplanes provide M_i distinct regions which contain at least one training data point. (Note that although $M_i \leqslant 2^{H_i}$, in reality it is upper bounded by the size of the training data set.) For each such region and from the training data points that lie in this region, the class of the majority is determined, and the region is considered to represent (or be labeled by) this class. Points of other classes that lie in this region are considered to be misclassified. The sum of the misclassifications for all the M_i regions constitutes the total misclassification miss_i associated with the string. Accordingly, the fitness of string i may be defined as

$$\text{fit}_i = (n - \text{miss}_i) - \alpha H_i, \quad 1 \leqslant H_i \leqslant H_{\max}, \tag{3}$$

$$\text{fit}_i = 0, \quad \text{otherwise}, \tag{4}$$

where n is the size of the training data set and $\alpha = 1/H_{\max}$.

Let us now explain how the first criterion is satisfied. Let two strings i and j have a number of misclassifications miss_i and miss_j, respectively, and let the number of hyperplanes encoded in them be H_i and H_j, respectively. Let $\text{miss}_i < \text{miss}_j$ and $H_i > H_j$. (Note that since the number of misclassified points can only be integers, $\text{miss}_j \geqslant \text{miss}_i + 1$.) Then

$$\text{fit}_i = (n - \text{miss}_i) - \alpha H_i,$$
$$\text{fit}_j = (n - \text{miss}_j) - \alpha H_j.$$

The aim now is to prove that $\text{fit}_i > \text{fit}_j$, or that $\text{fit}_i - \text{fit}_j > 0$. From the above equations,

$$\text{fit}_i - \text{fit}_j = \text{miss}_j - \text{miss}_i - \alpha(H_i - H_j).$$

If $H_j = 0$, then $\text{fit}_j = 0$ (from Eq. (4)) and therefore $\text{fit}_i > \text{fit}_j$. When $1 \leqslant H_j \leqslant H_{\max}$, we have $\alpha(H_i - H_j) < 1$ since $(H_i - H_j) < H_{\max}$. Obviously, $\text{miss}_j - \text{miss}_i \geqslant 1$. Therefore $\text{fit}_i - \text{fit}_j > 0$, or, $\text{fit}_i > \text{fit}_j$.

The second criterion is also fulfilled since $\text{fit}_i < \text{fit}_j$ when $\text{miss}_i = \text{miss}_j$ and $H_i > H_j$.

3.3. Genetic operators

Among the operations of selection, crossover and mutation, the selection operation used here may be one of those used in conventional GAs, while crossover and mutation need to be newly defined for VGA. These are now described in detail.

Crossover. Two strings, i and j, having lengths l_i and l_j, respectively, are selected from the mating pool. Let $l_i \leqslant l_j$. Then string i is padded with #s so as to make the two lengths equal. Conventional crossover like single point crossover, or two point crossover (Goldberg, 1989) is now performed over these two strings with probability μ_c. The following two cases may now arise:

- All the hyperplanes in the offspring are complete. (A hyperplane in a string is called *complete* if all the bits corresponding to it are either defined (i.e., 0s and 1s) or #s. Otherwise it is incomplete.)
- Some hyperplanes are incomplete.

In the second case let u be the number of defined bits (either 0 or 1) and t be the total number of bits per hyperplane $= (N-1) * b_1 + b_2$ (from Eq. (1)). Then, for each incomplete hyperplane, all the #s are set to defined bits (either 0 or 1 randomly) with probability u/t. In case this is not permitted, all the defined bits are set to #. Thus, each hyperplane in the string becomes complete. Subsequently, the string is rearranged so that all the #s are pushed to the end, or in other words all the hyperplanes are transposed to the beginning of the strings. The information about the number of hyperplanes in the strings is updated accordingly.

Mutation. In order to introduce greater flexibility in the method, the mutation operator is defined in such a way that it can both increase and

decrease the string length. For this, the strings are padded with #s such that the resultant length becomes equal to $l_{\max}$. Now for each defined bit position, it is determined whether conventional mutation (Goldberg, 1989) can be applied or not with probability μ_m. Otherwise, the position is set to # with probability μ_{m_1}. Each undefined position is set to a defined bit (randomly chosen) according to another mutation probability μ_{m_2}. These are described in Fig. 2.

Note that mutation may result in some incomplete hyperplanes, and these are handled in a manner, as done for the crossover operation. For example, the operation on the defined bits, i.e., when $k \leqslant l_i$ in Fig. 2, may result in a decrease in the string length, while the operation on #s, i.e., when $k > l_i$ in the figure, may result in an increase in the string length. Also, mutation may yield strings having all #s indicating that no hyperplanes are encoded in it. Consequently, this string will have fitness $= 0$ and will be automatically eliminated during selection.

Note that the operations defined here for designing the VGA-classifier are different from those used in (Smith, 1980; Goldberg et al., 1989; Harp and Samad, 1992; Maniezzo, 1994; Srikanth et al., 1995).

```
Begin
    l_i = length of string i
    Pad string i with # so that its length becomes l_max
    for k = 1 to l_max do
        Generate rnd, rnd1 and rnd2 randomly in [0,1]
        if k ≤ l_i do /* defined bits */
            if rnd < μ_m do /* Conventional mutation */
                flip bit k of string i
            else /* try changing to # */
                if rnd1 < μ_m1 do
                    Set bit k of string i to #
                endif
            endif
        else /* k > l_i i.e., # */
            if rnd2 < μ_m2 do /* Set to defined */
                Position k of string i set to 0 or 1 randomly
            endif
        endif
    endfor
End
```

Fig. 2. Mutation operation for string i.

As in conventional GAs, the operations of selection, crossover and mutation are performed here over a number of generations till a user specified termination condition is attained. Elitism is incorporated such that the best string seen up to the current generation is preserved in the population. The best string of the last generation, thus obtained, along with its associated labeling of regions provides the classification boundary of the n training samples. After the design is complete, the task of the classifier is to check, for an unknown pattern, the region in which it lies, and to assign a label accordingly.

4. Issues of minimum miss and H

In this section we prove that the above mentioned VGA-classifier will provide the minimal misclassification error during training, for an infinitely large number of iterations. At the same time it will require a minimum number of hyperplanes in doing so.

For proving this we use the result of (Bhandari et al., 1996), where it has been established that for an infinitely large number of iterations, an elitist model of GA will surely provide the optimal string. In order to prove this convergence they assumed that the probability of going from any string to the optimal one is always greater than zero, and the probability of going from a population containing the optimal string to one not containing the optimal one is zero. Since the mutation operation and elitism of the proposed VGA ensure that both these conditions are met, the result of (Bhandari et al., 1996) regarding the convergence to the optimal string is valid for VGA as well.

Let us now consider the fitness function for string i (Eq. (3)). Maximization of the fitness function means minimization of

$$\text{miss}_i + \alpha H_i,$$

where $\alpha = 1/H_{\max}$. Let us call this the error function (err_i).

Let for any size of the training data set (n), the minimum value of the error function as obtained by the VGA-classifier be

$$\mathrm{err}_{\min} = \mathrm{miss}^0 + \alpha H^0$$

after it has been executed for an infinitely large number of iterations. Then according to Bhandari et al. (1996), this corresponds to the optimal string. Therefore we may write

$$\mathrm{miss}^0 + \alpha H^0 \leqslant \mathrm{miss} + \alpha H, \quad \forall \mathrm{miss},\ H. \tag{5}$$

Theorem 1. *For any value of* H, $1 \leqslant H \leqslant H_{\max}$, *the minimal number of misclassified points is* miss^0.

Proof. The proof is trivial and follows from the definition of the fitness function (Eq. (3)) and the fact that $\mathrm{miss}^0 + \alpha H^0 \leqslant \mathrm{miss} + \alpha H, \quad \forall \mathrm{miss},\ H$ (Eq. (5)).

Theorem 2. H^0 *is the minimal number of hyperplanes required for providing* miss^0 *misclassified points.*

Proof. Let the converse be true, i.e., there exists some H', $H' < H^0$, that provides miss^0 misclassified points. In that case, the corresponding fitness value would be $\mathrm{miss}^0 + \alpha H'$. Note that now $\mathrm{miss}^0 + \alpha H^0 > \mathrm{miss}^0 + \alpha H'$. This violates Eq. (5). Hence $H' \not< H^0$, and therefore H^0 is the minimal number of hyperplanes required for providing miss^0 misclassified points. □

From Theorems 1 and 2, it is proved that for any value of n, the VGA-classifier provides the minimum number of misclassified points for an infinitely large number of iterations, and it requires a minimum number of hyperplanes in doing so.

5. Implementation and results

The experimental investigation presented in this section has two parts. In the first part, the effectiveness of VGA in automatically determining the value of H of the classifier is demonstrated for two sets of artificial data, a speech data set and the Iris data. The recognition scores of the VGA-classifier are also compared with those of the fixed length GA-classifier. Secondly, we compare our concept of using variable string lengths in GA with another similar approach (Srikanth et al., 1995). For this purpose we have implemented their different operators in our classification algorithm for the above mentioned four data sets.

The 2-dimensional artificial data sets, ADS 1 (Fig. 3) and ADS 2 (Fig. 4), consist of 557 and 417 points, respectively, belonging to two classes. The real life speech data, *Vowel* (Pal and Majumdar, 1977), consists of 871 samples having three feature values (corresponding to the three formant fre-

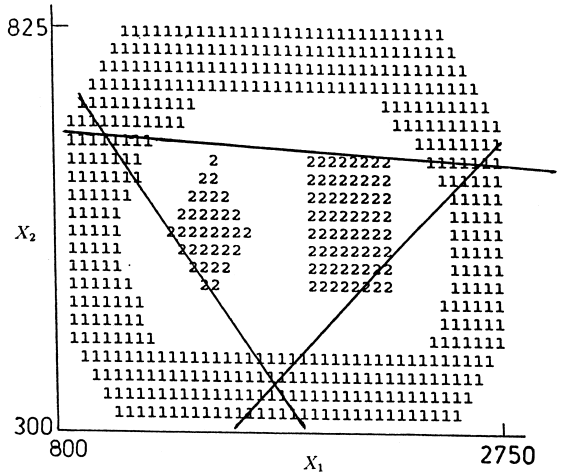

Fig. 3. ADS 1 along with VGA boundary for $H_{\max} = 10$ when 10% of the data set is used for training.

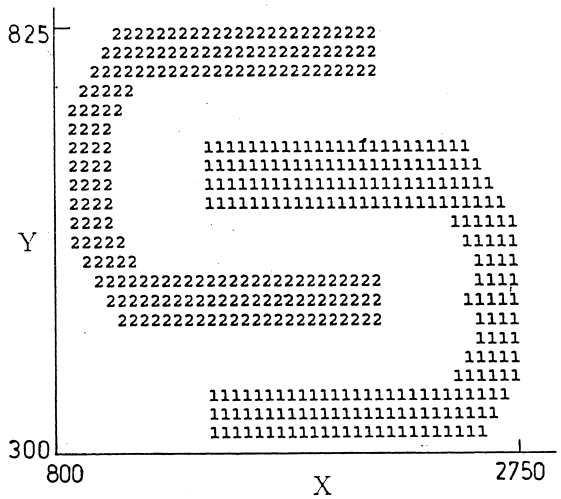

Fig. 4. ADS 2.

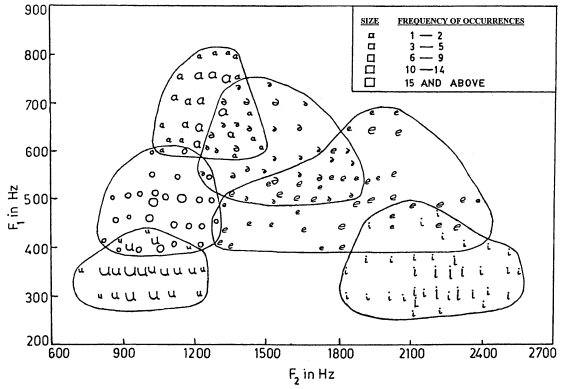

Fig. 5. Vowel data in the F_1–F_2 plane.

quencies) and six classes $\{\delta, a, i, u, e, o\}$. Fig. 5 shows the overlapping class structures in the first and second formant frequency plane. The Iris data comprises 150 samples having four features and three classes with 50 points in each class.

A fixed population size of 20 is chosen. The *roulette wheel strategy* (Goldberg, 1989) is used to implement proportional selection. As in an earlier investigation (Bandyopadhyay et al., 1995), *single point crossover* is applied with a fixed crossover probability of 0.8. A variable value of the mutation probability μ_m is selected from the range [0.01, 0.333]. Initially it assumes a high value, gradually decreasing at first, and then increasing again in the later stages of the algorithm. 200 iterations are performed with each mutation probability value. The values of μ_{m_1} and μ_{m_2} mentioned in Section 3.3 are set to 0.1. The process is executed for a maximum of 3000 iterations. *Elitism* is incorporated by replacing the worst string of the present generation by the best string seen upto the previous generation.

5.1. Performance of the VGA-classifier

Tables 1 and 2 show the number of hyperplanes H_{VGA} as determined automatically by the VGA-classifier for modeling the class boundaries of the four data sets when the classifier is trained with 10% and 50% samples, respectively. Two different values of $H_{\max}$ are used for this purpose viz., $H_{\max} = 6$ and $H_{\max} = 10$. The overall recognition scores obtained during testing of the VGA-classifier along with those obtained for the fixed length version (i.e., GA-classifier) with $H = 6$ and 10 are also shown. (Note that $H = 6$ had been found to provide, on an average, good recognition scores in

Table 1
H_{VGA} and the overall recognition scores (%) during testing (when 10% of data set is used for training and the remaining 90% for testing)

Data set	VGA-classifier $H_{\max} = 10$		Score for GA-classifier $H = 10$	VGA-classifier $H_{\max} = 6$		Score for GA-classifier $H = 6$
	H_{VGA}	Score		H_{VGA}	Score	
ADS 1	3	95.62	84.26	4	96.21	93.22
ADS 2	6	88.16	84.04	5	88.35	88.29
Vowel	6	73.66	69.21	6	71.19	71.99
Iris	2	95.56	76.29	2	95.81	93.33

Table 2
H_{VGA} and the overall recognition scores (%) during testing (when 50% of the data set is used for training and the remaining 50% for testing)

Data set	VGA-classifier $H_{\max} = 10$		Score for GA-classifier $H = 10$	VGA-classifier $H_{\max} = 6$		Score for GA-classifier $H = 6$
	H_{VGA}	Score		H_{VGA}	Score	
ADS 1	4	96.41	95.92	4	96.83	96.05
ADS 2	5	95.22	94.56	3	96.26	96.17
Vowel	6	78.26	77.77	6	77.11	76.68
Iris	2	97.60	93.33	2	97.67	97.33

earlier experiments (Bandyopadhyay et al., 1995) with these data sets.) The scores provided are the average values obtained over five different runs of the algorithms.

The results demonstrate that in all the cases, the VGA-classifier is able to evolve an appropriate value of H_{VGA} from H_{max}. In addition, its recognition score on the test data set is found, on average, to be comparable to if not higher than that of the GA-classifier. There is only one exception to this for the Vowel data when 10% of the samples is used for training (Table 1). In this case, $H_{max} = 6$ does not appear to be a high enough value for modeling the decision boundaries of the *Vowel* classes with the VGA-classifier. This is reflected in both the tables, where the scores for VGA-classifier with $H_{max} = 6$ are less than those with $H_{max} = 10$.

In all the cases where the number of hyperplanes for modeling the class boundaries is less than 6, the scores of the VGA-classifier with $H_{max} = 6$ are found to be superior to those with $H_{max} = 10$. This is so because with $H_{max} = 10$, the search space is larger as compared to that for $H_{max} = 6$, which makes it difficult for the classifier to arrive at the optimum arrangement quickly or within the maximum number of iterations considered here. (Note that it may have been possible to further improve the scores and also reduce the number of hyperplanes, if more iterations of the VGA were executed.)

In general, the scores of the GA-classifier (fixed length version) with $H = 10$ are seen to be lower than those with $H = 6$ because of two reasons; overfitting of the training data and difficulty of searching a larger space. The only exception is with Vowel for training with 50% data, where the score for $H = 10$ is larger than that for $H = 6$. This is expected, in view of the overlapping classes of the data set and the significantly large size of the training data. One must note in this context that the detrimental effect of overfitting on the generalization performance increases with a decrease in the size of the training data.

As an illustration, the decision boundary obtained by the VGA-classifier for ADS 1 when 10% of the data set is chosen for training is shown in Fig. 3.

5.2. Comparison with the method in (Srikanth et al., 1995)

In this section an investigation is made to compare the performance of our concept of using variable string length in GA with that of another similar approach (Srikanth et al., 1995). For this purpose the operators used in (Srikanth et al., 1995) are implemented here for the same problem of pattern classification using hyperplanes, and the resulting performance is compared to that of our VGA-classifier for the four data sets. Before providing the results, let us describe in brief the method of incorporating variable string lengths in GAs as proposed in (Srikanth et al., 1995).

The initial population is created randomly such that each string encodes the parameters of only one hyperplane. The fitness of a string is characterized by just the number of training points it classifies correctly, irrespective of the number of hyperplanes encoded in it. Among the genetic operators, traditional selection and mutation are used. A new form of crossover, called *modulo crossover*, is used which keeps the sum of the lengths of the two chromosomes constant both before and after crossover.

Two other operators are used in conjunction with the modulo crossover for the purpose of faster recombination and juxtaposition. These are the *insertion* and *deletion* operators. During insertion, a portion of the genetic material from one chromosome is inserted at a random *insert-location* in the other chromosome. Conversely, during deletion, a portion of a chromosome is deleted to result in a shorter chromosome.

Tables 3 and 4 show the comparative overall recognition scores during both training and testing of the VGA-classifier for the above mentioned four data sets when our approach of incorporating variable string length is compared with that adopted in (Srikanth et al., 1995) for 10% and 50% training data, respectively. Other parameters are kept the same as before. Results shown are the average values taken over five different runs. For keeping parity, the VGA of Srikanth et al. is implemented such that no more than 10 hyperplanes are used for modeling the decision boundary of the data sets. The table also shows the number of

Table 3
Comparative classification performance of VGA-classifier for $H_{max} = 10$ using two types of variable string lengths (when 10% of the data set is used for training and the remaining 90% for testing)

Data set	Proposed VGA			VGA (Srikanth et al.)		
	Training score (%)	Test score (%)	H_{VGA}	Training score (%)	Test score (%)	H_{VGA}
ADS 1	100	95.62	3	100	93.16	6
ADS 2	92.68	88.16	6	99.10	90.50	6
Vowel	80.00	73.66	6	97.36	70.22	9
Iris	100	95.56	2	100	94.98	2

Table 4
Comparative classification performance of VGA-classifier for $H_{max} = 10$ using two types of variable string lengths (when 50% of the data set is used for training and the remaining 50% for testing)

Data set	Proposed VGA			VGA (Srikanth et al.)		
	Training score (%)	Test score (%)	H_{VGA}	Training score (%)	Test score (%)	H_{VGA}
ADS 1	98.18	96.41	4	100.00	96.01	9
ADS 2	97.21	95.22	5	100.00	94.85	7
Vowel	79.73	78.26	6	85.48	78.37	9
Iris	100	97.60	2	100.00	94.67	5

hyperplanes, H_{VGA}, generated by the two methods for one particular run. Since the VGA of Srikanth et al. does not take care of the minimization of the number of hyperplanes while maximizing the fitness function, the H_{VGA} is usually higher than that of our method.

As is evident from the tables, the performance of the classifier during training is better for the VGA of Srikanth et al. than the proposed one for all the data sets. In fact, the former is found to converge quickly to an arrangement of hyperplanes that provides a high value of the classification accuracy. The VGA-classifier, on the other hand, attempts to reduce both the number of misclassified points and the number of hyperplanes. It therefore takes longer to attain a particular level of accuracy. In order to demonstrate this let us consider the Vowel data as an example. Fig. 6 shows the variation of the best recognition scores with the number of generations obtained by the two methods for Vowel with 10% training data. As is obvious from the figure, except at the initial stages of the algorithms, given any recognition score, the method of Srikanth et al. attains it earlier than our VGA-classifier.

One may note that the method of Srikanth et al., in general, uses more hyperplanes (of which many were found to be redundant on investigation), which results in an increase in the execution time for fitness computation. This is evident from Fig. 7, which shows the total time taken for fitness computation by the two methods as a function of H_{max}. Here too, we consider Vowel with 10% training data, as an illustration.

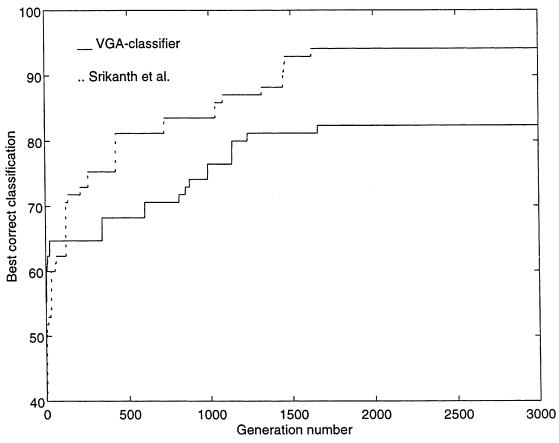

Fig. 6. Variation of the best correct classification during training with the number of generations for Vowel when $H_{max} = 10$ and 10% of the data set is used for training.

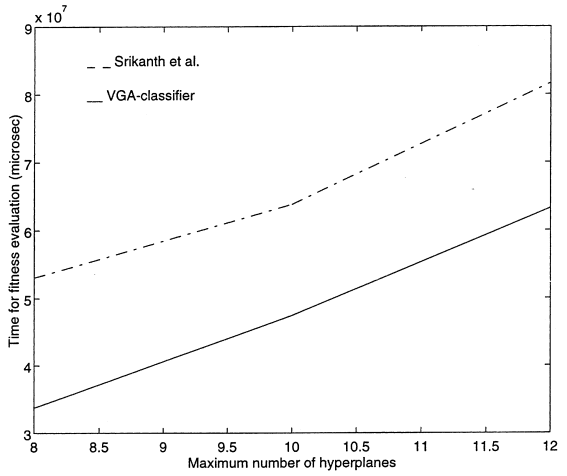

Fig. 7. Variation of the time required for fitness evaluation with H_{max} for Vowel when 10% of the data set is used for training.

From the training performance, it appears that the operators used by Srikanth et al., are better able to recombine the subsolution blocks into larger blocks. However this is seen, in general, to result in comparatively poorer scores during testing. To consider a typical example in one of the cases for the Vowel data set when 10% data is used for training, 10 hyperplanes were used to provide a training recognition score of 97.47%, while the recognition score during testing fell to 68.95%.

It is also found that with an increase in the size of the training data, the number of hyperplanes for modeling the class boundaries increase for the algorithm of Srikanth et al. Furthermore, as expected, the performance of all the classifiers is improved with an increase in the size of the training data from 10% to 50%.

6. Conclusions

The problem of fixing the appropriate value of H a priori of the GA-classifier (Bandyopadhyay et al., 1995) has been resolved by using the concept of variable string lengths in genetic algorithms. New genetic operators are defined to deal with the concept of variable string lengths for formulating the classifier. The fitness function has been defined so that its maximization indicates minimization of the number of misclassified samples as well as of the required number of hyperplanes. It is proved that for an infinitely large number of iterations, the method is able to arrive at the optimal number of misclassified samples and will need an optimal number of hyperplanes for this purpose.

Experimental evidence for different percentages of training and test data indicates that, given a value of H_{max}, the algorithm can not only automatically evolve an appropriate value of H for a given data set, but also retain the performance of the GA-classifier. In Tables 1 and 2 we considered two typical values of H, namely, 6 and 10 to demonstrate this fact. However, for proper comparison, one needs to run the GA-classifier for several values of H. This would, obviously, increase the computational complexity of the GA-classifier manifold. Also, increasing the value of H in the GA-classifier will always provide improved training performance, by closely fitting the training data. This may result in decreased generalization capability of the classifier. Thus it becomes difficult to decide on the proper value of H for the GA-classifier based on its performance during training. The VGA-classifier, on the other hand, attempts to balance both the training performance and the generalization capability by reducing the number of misclassified points and the number of hyperplanes at the same time.

The method of using variable string length in the algorithm of Srikanth et al. is also implemented in our VGA-classifier for comparison. The former method is found to attain any level of accuracy faster than the latter during training, and at the same time uses more hyperplanes for constituting the decision boundary. This results in better training performance, mostly at the cost of reduced generalization capability. Additionally, the execution time for fitness computation is also larger, since no explicit effort is made to decrease the number of hyperplanes.

In this connection one may also note that the genetic operators and processing steps of the VGA described in this article entail very little disruption of those in the conventional GA. On the other hand this is not true for the method of Srikanth et al. which introduces two new processing steps viz., insertion and deletion, besides

using a significantly different crossover operator. Furthermore, the former method requires the specification of H_{max}, whereas such a constraint is not required for the latter one.

Acknowledgements

This work was carried out when Ms. Sanghamitra Bandyopadhyay held the Dr. K.S. Krishnan fellowship awarded by the Department of Atomic Energy, Govt. of India.

References

Bandyopadhyay, S., Murthy, C.A., Pal, S.K., 1995. Pattern classification using genetic algorithms. Pattern Recognition Letters 16, 801–808.

Bhandari, D., Murthy, C.A., Pal, S.K., 1996. Genetic Algorithm with elitist model and its convergence. Internat. J. Patt. Recog. and Art. Intell. 10, 731–747.

Gelsema, E.S., 1995. Pattern Recognition Letters 16 (8), Special Issue on Genetic Algorithms.

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

Goldberg, D.E., Deb, K., Korb, B., 1989. Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems 3, 493–530.

Harp, S.A., Samad, T., 1992. Genetic synthesis of neural network architecture. In: L. Davis (Ed.), Handbook of Genetic Algorithms. New York, pp. 202–221.

Maniezzo, V., 1994. Genetic evolution of the topology and weight distribution of neural networks. IEEE Trans. Neural Networks 5, 39–53.

Pal, S.K., Majumdar, D.D., 1977. Fuzzy sets and decision making approaches in vowel and speaker recognition. IEEE Trans. Syst. Man Cybern. SMC7, 625–629.

Pal, S.K., Wang, P.P., 1996. Genetic Algorithms for Pattern Recognition. CRC Press, Boca Raton.

Smith, S.F., 1980. A learning system based on genetic algorithms. Ph.D. Dissertation, University of Pittsburgh, PA.

Srikanth, R., George, R., Warsi, N., Prabhu, D., Petry, F.E., Buckles, B.P., 1995. A variable-length genetic algorithm for clustering and classification. Pattern Recognition Letters 16, 789–800.