The Third AFSS(1998), 634－639

Variable Length Chromosomes in Genetic Algorithms for Modeling the Class Boundaries

Sanghamitra Bandyopadhyay, Sankar K. Pal and C. A. Murthy
Machine Intelligence Unit
Indian Statistical Institute
203 B.T. Road, Calcutta 700 035, INDIA.
e-mail : sankar@isical.ernet.in

Abstract

A methodology based on the concept of variable string length GA (VGA) is developed for determining automatically the number of hyperplanes and their appropriate arrangement for modeling the class boundaries of a given training data set in $\Re^N$. The genetic operators and fitness function are newly defined to take care of the variability in chromosome length. Experimental results on different artificial and real life data sets are provided.

Keywords: Optimum hyperplane fitting, speech recognition, variable string length.

1 Introduction

Genetic Algorithms (GAs) [1] 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.

In this article, the searching capability of GA is used for the placement of a number of hyperplanes, say H, for approximating the decision boundaries of a given training data set. The method involves encoding the parameters of the hyperplanes in binary strings called *chromosomes*, in the feature space that yields minimum misclassification. Since it is difficult to ascertain the the number of hyperplanes that may be required for a given data set *a priori*, the value is kept variable, using the GA to evolve an appropriate value adaptively.

For this purpose, the concept of variable length strings in GA has been adopted. Unlike the conventional GA, here the length of a string is not fixed. The chromosomes, therefore, encode the parameters of a number of hyperplanes, whose value may now vary. Crossover and mutation operators are accordingly defined. A factor has been incorporated into the fitness function that rewards a string with smaller number of misclassified samples as well as smaller number of hyperplanes. Let the classifier so designed utilizing the concept of variable string lengths be called *VGA-classifier*.

Note that in a previous investigation using fixed number of hyperplanes and consequently fixed string length, it was demonstrated in [2] that the resulting fixed string length 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 k-

NN rule (for different values of k), Bayes maximum likelihood classifier and multilayer perceptron based classifier. A comparison of the *VGA-classifier* and the previous *GA-classifier* shows that incorporation of VGA is not only able to evolve the appropriate number of hyperplanes but also provides classification performance comparable to, often better than the of the previous fixed length *GA-classifier*.

2 Variable String Length GA Classifier

Although the concept of variable string lengths in genetic algorithm (VGA) has been used earlier in [3,4,5,6], the operations defined here are new. They are described here. Note that the sequence of the different operations for GA (as shown in Fig. 1) is applicable to VGA too.

```
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
```

Figure 1: Basic steps in GA

Chromosome Representation and Population Initialization

The chromosomes are represented by strings of 1, 0 and # (don't care), encoding the parameters of variable number of hyperplanes. In $\mathcal{R}^N$, N parameters are required for representing one hyperplane. These are $N-1$ angle variables, $angle_1^i, \ldots, angle_{N-1}^i$, 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

$$l_H = (N-1) * b_1 + b_2 \quad (1)$$
$$l_{max} = H_{max} * l_H \quad (2)$$

respectively.

Let string i represent H_i hyperplanes. Then its length l_i is
$l_i = H_i * l_H$.

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.

Fitness Computation

The fitness function (which is maximized) is defined in such a way that

i : a string with smaller value of misclassifications is considered to be fitter than a string with a larger value, irrespective of the number of hyperplanes i.e., it first of all minimizes the number of misclassified points, and then

ii : 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 $M_i \leq 2^{H_i}$, 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) the said 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

$$fit_i = (n - miss_i) - \alpha H_i 1 \leq H_i \leq H_{max} \quad (3)$$
$$= 0, \text{ otherwise} \quad (4)$$

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

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

$fit_i = (n - miss_i) - \alpha H_i$,
$fit_j = (n - miss_j) - \alpha H_j$.

The aim now is to prove that $fit_i > fit_j$, or that $fit_i - fit_j > 0$. From the above equations,
$fit_i - fit_j = miss_j - miss_i - \alpha(H_i - H_j)$.

If $H_j = 0$, then $fit_j = 0$ (from Eq. 4) and therefore $fit_i > fit_j$. When $1 \leq H_j \leq H_{max}$, we have $\alpha(H_i - H_j) < 1$ since $(H_i - H_j) < H_{max}$. Obviously, $miss_j - miss_i \geq 1$. Therefore $fit_i - fit_j > 0$, or, $fit_i > fit_j$.

The second criterion is also fulfilled since $fit_i < fit_j$ when $miss_i = miss_j$ and $H_i > H_j$.

Genetic Operators

Among the operations of selection, crossover and mutation, the selection operation used here may be one of those used in conventional GA, 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 $l_i \leq l_j$. Then string i is padded with #s so as to make the two lengths equal. Conventional crossover like single point crossover, two point crossover (Goldberg, 1989) is now performed over these two strings with probability μ_c. The following two cases may now arise :
(a) 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.)
(b) Some hyperplanes are incomplete.

In the second case let u = number of defined bits (either 0 or 1) and t = 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 $\frac{u}{t}$. 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.

```
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
```

Figure 2: Mutation operation for string i

Note that mutation may result in some incomplete hyperplanes, and these are handled in a manner, as done for crossover operation. For example, the operation on the defined bits, i.e., when $k \leq l_i$ 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 indi-

cating that no hyperplanes are encoded in it. Consequently, this string will have fitness = 0 and will be automatically eliminated during selection. ♣

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 upto the current generations 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 put the label accordingly.

3 Implementation and Results

The experimental investigation presented in this section demonstrates the effectiveness of VGA in automatically determining the value of H of the classifier for two sets of artificial data, a speech data and Iris data. The recognition scores of the *VGA-classifier* are also compared with those of the fixed length *GA-classifier* [2].

The 2-dimensional artificial data sets, *ADS 1* (Fig. 3) and *ADS 2* [2], consist of 557 and 417 points respectively belonging to two classes. The real life speech data, *Vowel* [2], consists of 871 samples having three feature values (corresponding to the three formant frequencies) and six classes $\{\delta, a, i, u, e, o\}$. The class structures of this data set are known to be highly overlapping. 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. *Roulette wheel strategy* [1] is used to implement proportional selection. *Single point crossover* is applied with a fixed crossover probability of 0.8. A variable value of mutation probability μ_m is varied in the range [0.01, 0.333] [2]. The values of μ_{m_1} and μ_{m_2} mentioned earlier are set to 0.1. The process is executed for a maximum 3000 iterations. *Elitism* is incorporated by replacing the worst string of the present generation by the best string seen upto the previous generation.

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 aforesaid four data sets for two different values of H_{max} viz., $H_{max} = 10$ and $H_{max} = 6$, when the classifier is trained with 10% data. Tables 3 and 4 show the number of hyperplanes H_{VGA} as determined automatically by the *VGA-classifier* for modeling the class boundaries of the aforesaid four data sets for two different values of H_{max} viz., $H_{max} = 10$ and $H_{max} = 6$, when the classifier is trained with 50% data. The overall recognition scores obtained during testing of the *VGA-classifier* along with their comparison with those obtained for the fixed length version (i.e., *GA-classifier*) with $H = 6$ and 10 are also shown in the tables. (Note that $H = 6$ had been found to provide, on an average, good recognition scores in earlier experiments [2] with these data sets.) The scores provided are the average values obtained over 5 different runs of the algorithms.

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

Data set	VGA-classifier $H_{max} = 10$		Score for GA-classifier $H = 10$
	H_{VGA}	Score	
ADS 1	3	95.62	84.26
ADS 2	6	88.16	84.04
Vowel	6	73.66	69.21
Iris	2	95.56	76.29

Table 2: H_{VGA} for $H_{max} = 6$ and the comparative overall recognition scores (%) during testing (when 10% of the data set is used for training and the remaining 90% for testing)

Data set	VGA-classifier $H_{max} = 6$		Score for GA-classifier $H = 6$
	H_{VGA}	Score	
ADS 1	4	96.21	93.22
ADS 2	5	88.35	88.29
Vowel	6	71.19	71.99
Iris	2	95.81	93.33

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 an average, to be higher than that of the *GA-classifier*. There is only one exception to this for the *Vowel*

Table 3: H_{VGA} for $H_{max} = 10$ and the comparative 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 $H = 10$
	H_{VGA}	Score	
ADS 1	4	96.41	95.92
ADS 2	5	95.22	94.56
Vowel	6	78.26	77.77
Iris	2	97.60	93.33

Table 4: H_{VGA} for $H_{max} = 6$ and the comparative 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 $H_{max} = 6$		Score for GA-classifier $H = 6$
	H_{VGA}	Score	
ADS 1	4	96.83	96.05
ADS 2	3	96.26	96.17
Vowel	6	77.11	76.68
Iris	2	97.67	97.33

data when 10% of the samples is used for training and $H_{max} = 6$ (Table 2). In this case, $H_{max} = 6$ does not appear to be a high enough value for modeling the decision boundaries of *Vowel* classes with *VGA-classifier*. This is reflected in both the cases for 10% and 50% training data, 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 *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 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 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.

4 Discussion and Conclusions

The concept of variable string lengths in genetic algorithm has been used here for the purpose of placement of a number of hyperplanes in $\Re^N$ for modeling the class boundaries of a given training data set. 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 the required number of hyperplanes. It is proved that for infinitely large number of iterations the method is able to arrive at the optimal number of misclassified samples and will need 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 be able to automatically evolve an appropriate value of H for a given data set, but also result in improved performance of the classifier.

Acknowledgments : 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

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

2 S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "Pattern classification using genetic algorithms," *Patt. Recog. Lett.*, vol. 16, pp. 801–808, August 1995.

3 R. Srikanth, R. George, N. Warsi, D. Prabhu, F. Petry, and B. Buckles, "A variable-length genetic algorithms for clustering and classification," *Patt. Recog. Lett.*, vol. 16, pp. 801–808, August 1995.

4 S. F. Smith, *A Learning System Based on Genetic Algorithms.* PhD thesis, University of Pittsburg, PA, 1980.

5 S. A. Harp and T. Samad, "Genetic synthesis of neural network architecture," in *Handbook of Genetic Algorithms*, (L. Davis, ed.), pp. 202 – 221, New York: Van Nostrand Reinhold, 1992.

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

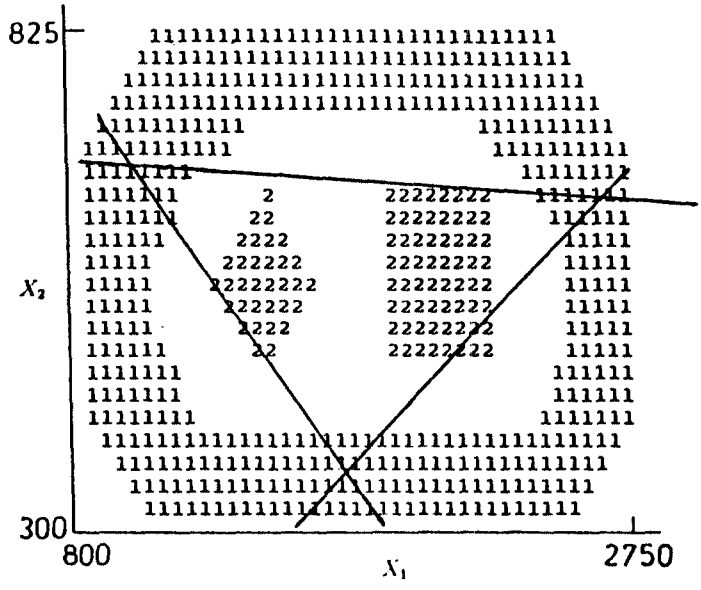

Figure 3: *ADS 1* along with VGA boundary for $H_{max} = 10$ when 10% of the data set is used for training