ELSEVIER

Pattern Recognition Letters 16 (1995) 801–808

Pattern Recognition Letters

Pattern classification with genetic algorithms

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

Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta 700 035, India

Received 17 January 1995; revised 16 March 1995

Abstract

A method is proposed for finding decision boundaries, approximated by piecewise linear segments, for the classification of patterns in $\mathbb{R}^2$, using an elitist model of a genetic algorithm. It involves the generation and placement of a set of lines (represented by strings) in the feature space that yields minimum misclassification. The effectiveness of the algorithm is demonstrated, for different parameter values, on both artificial data and speech data having non-linear class boundaries. Its comparison with the k-NN classifier is also made.

Keywords: Elitist model; Genetic operators; Line fitting; Pattern recognition

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. GAs have applications in fields as diverse as VLSI design, pattern recognition, image processing, neural networks, etc. (Proceedings, 1991).

In this paper, an attempt is made to study the application of GAs for pattern classification in two-dimensional data space. Classification is a problem of generating decision boundaries that can successfully distinguish the various classes in the feature space. In real-life problems, the boundaries between the different classes are usually non-linear. In the present investigation, the characteristics of GAs have been exploited in searching for a number of lines which can approximate the non-linear boundaries which provide minimum misclassification.

The feature space is generally unbounded and continuous in nature. However, if bounding information can be derived from the training patterns and the space is discretized to sufficiently small intervals in each dimension, then the classification problem can be handled within the framework of Genetic Algorithms. A distinguishing feature of this approach is that the boundaries (approximated by piecewise linear segments) are generated explicitly for making decisions. Note that in the conventional methods or in multilayered perceptron-based approaches to pattern classification, the generation of boundaries is a consequence of the respective decision making processes.

* Corresponding author. Email: sankar@isical.ernet.in

SSDI 0167-8655(95)00052-6

2. Description of the methodology

We consider a fixed number of lines (say H) to denote a decision boundary in a two-dimensional feature space F_1–F_2. The value of H varies from problem to problem, depending on the number as well as the nature of the classes. These H lines need to be encoded as a single string. Note that each line provides two halfspaces – a positive halfspace and a negative halfspace, thereby yielding two regions. For H lines, the maximum number of such regions is 2^H. The methodology to find a near-optimal classifier is described below.

2.1. Search space for lines

Let us assume that there are tr training patterns available. Then in the first step the maximum and minimum values of each of the two features, F_1 and F_2, are computed. Let these be Max_1, Min_1, Max_2, Min_2 for the features F_1 and F_2, respectively. Then the rectangle enclosing the sample points is given by the vertices (Min_1, Min_2), (Min_1, Max_2), (Max_1, Min_2), (Max_1, Max_2). Fig. 1 shows an example. The rectangle represents the search space for the possible lines which may be considered as candidates for the formation of the decision boundary.

2.2. Line generation

In this section we describe how to generate a finite number of parallel lines in a finite number of directions, so that even the two closest points of the data set, which may belong to two different classes, can be separated by a line in at least one direction. For this purpose, a distance *dist* is computed as follows:

Let $\mathscr{S}$ represent the training data set. Then we define

$$dist = \min\{\text{dist}(x, y) \mid x \in \mathscr{S},\ y \in \mathscr{S},\ x \neq y\}$$

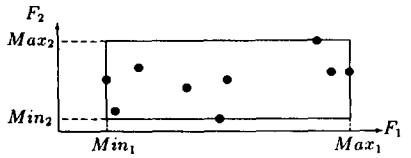

Fig. 1. Training patterns and the enclosing rectangle.

where dist(x, y) denotes the Euclidean distance between points x and y. Consequently, the separation between two consecutive parallel lines, in any direction, is taken to be $dist/2$. Thus, the maximum number of parallel lines, max_lines, which may have to be considered for searching in any particular direction within the search space, is computed as

$$max\_lines = \left\lceil \frac{diag}{dist} * 2 \right\rceil$$

where $\lceil x \rceil$ gives the smallest integer greater than or equal to x and $diag$ is the length of the diagonal of the rectangle enclosing the training points.

One way of representing a line is by the equation

$$x_1 \cos \alpha + x_2 \sin \alpha = d. \tag{1}$$

Here α is the angle between the F_1-axis and the unit normal to the line; d is the perpendicular distance of the normal from the origin. The way of specifying the two variables α and d is mentioned below.

● *Angle (direction) specification.* The entire feature space will be spanned if the angle α is allowed to vary in the range from 0 to π radians. The angle space is discretized to sufficiently small intervals as

$$0, \delta\pi, 2\delta\pi, \ldots, n * \delta\pi.$$

The number of discrete angles considered is then equal to $n + 1$, and $\delta = 1/n$. An angle α can thus be specified by a number *angle* in the range $[0, n]$ such that $\alpha = angle * \delta\pi$.

● *Perpendicular distance specification.* Once the angle α is fixed, the orientation of the line becomes fixed. For a given orientation, the perpendicular distances of the two lines passing through the base points $((Min_1, Min_2)$ and $(Max_1, Min_2))$ of the enclosing rectangle, from the origin, are computed from Eq. (1). (The perpendicular distance, d, assumes a negative value if it lies in the negative halfspace of the F_2-axis.) Among these, the one with the minimum value, $d_{\min}$, is selected as the base line. This is demonstrated in Fig. 2 where the line through point 2 is the base line. The search space for lines with this orientation is restricted at one end by the base line. In other words, all lines with $d < d_{\min}$, are automatically discarded from the search space. At the other end is a parallel line at a perpendicular

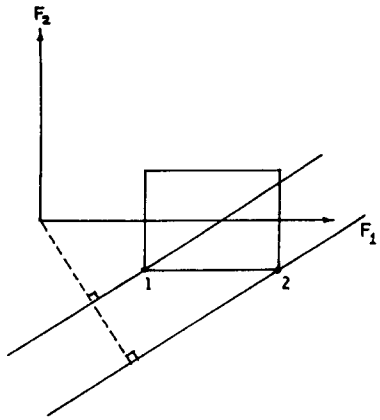

Fig. 2. Fixing the base line.

distance of $(d_{\min} + max\_lines \times (dist/2))$ from the origin.

Therefore any line at a distance *offset* from the base line can be specified by a number p in the range [0, *max_lines*] such that $offset = p * dist/2$. The perpendicular distance d of that line from the origin is therefore

$$d = d_{\min} + offset.$$

A line *ln* is now specified by the two integer variables *angle* and p.

2.3. String representation and population initialization

Each string is composed of a fixed number, H, of lines. Each line is encoded in terms of an angle variable and a perpendicular distance variable (both assume integer values). If the angle variable is represented by *b_ang* number of bits and the perpendicular distance by *b_perp* bits, then the length of each string, *str_len* equals

$$str\_len = (b\_ang + b\_perp) * H.$$

The GA generally works with a fixed population size of *Pop*. Initially, each binary string, of length *str_len*, is generated by randomly selecting two variables *angle* and p from the intervals [0, n] and [0, *max_lines*] respectively, for each of the H lines in a string.

2.4. Region identification and fitness computation

The computation of the fitness is done for each string in the population. The fitness of a string is characterized by the number of points it misclassifies. A string with the lowest misclassification is therefore considered to be the fittest among the population of strings. Note that every string str_i, $i = 1, 2, \ldots, Pop$, represents H lines denoted by ln_j^i, $j = 1, 2, \ldots, H$.

For each ln_j^i, the parameters l_1^{ij}, l_2^{ij} and d^{ij} are retrieved. For each training pattern point (x_1^k, x_2^k), $k = 1, 2, \ldots, tr$, the sign with respect to the line ln_j^i, i.e., the sign of the expression

$$\cos\ \alpha^{ij}\ x_1^k + \sin\ \alpha^{ij}\ x_2^k - d^{ij} \qquad (2)$$

is found. The sign is digitized as 1 (0) if the point lies on the positive (negative) side of the line ln_j^i. The process is repeated for each of the lines, at the end of which we have a string $sign_i^k$, subsequently to be referred to as the sign string. This string, of length H, corresponds to the classification yielded by the string str_i of the population, for the kth training pattern. The class information of the kth training point is stored along with the sign string $sign_i^k$ for str_i in a linked list. This procedure is repeated for all the tr pattern points.

It is to be noted that although $sign_i^k$ can take on at most 2^H possible values (since H lines will yield a maximum of 2^H possible classifications), all of them

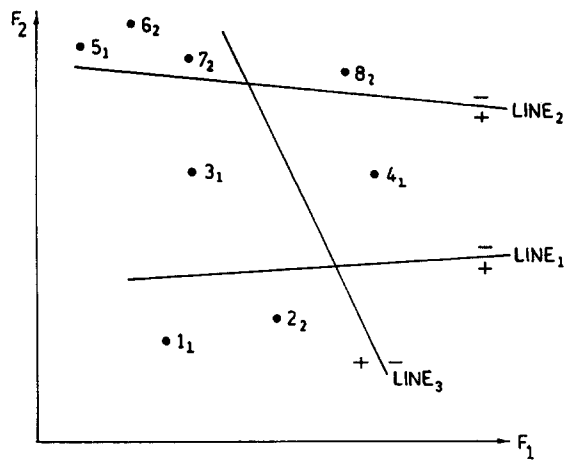

Fig. 3. An example with $H = 3$ and $tr = 8$.

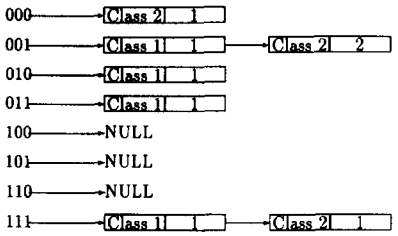

Fig. 4. Link list for the given example.

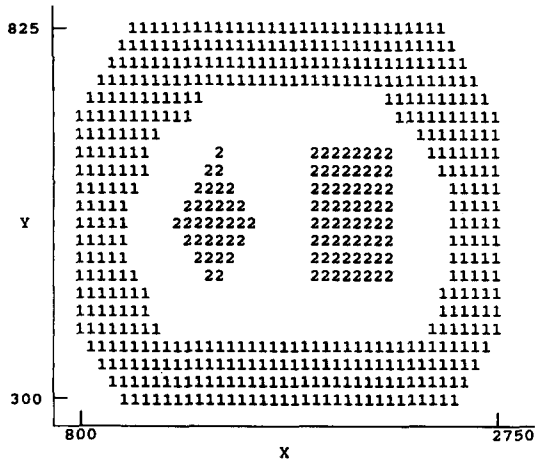

Fig. 5. Artificial data.

may not occur in practice. These sign strings, in fact, represent different regions of the search space. With each such sign string, a linked list is maintained. Each element of the list is an ordered pair indicating a class and its cardinality. The cardinality of a class denotes the number of training samples of that class which have been identified to fall into the region represented by the sign string.

The maximum class cardinality in the list for each sign string is found next. Then the region corresponding to that sign string is considered to provide the demarcation for the class possessing the maximum cardinality. All the points belonging to other classes which have been included in the same list, i.e. which lie in the same region, are considered to be misclassified. The number of misclassifications corresponding to all possible sign strings are summed up to give the resulting misclassification for the entire classifier string. It may so happen that the maximum cardinalities for two (or more) different sign strings may correspond to the same class. In that case, all these strings (correspondingly, union of all the different regions) are considered to provide the

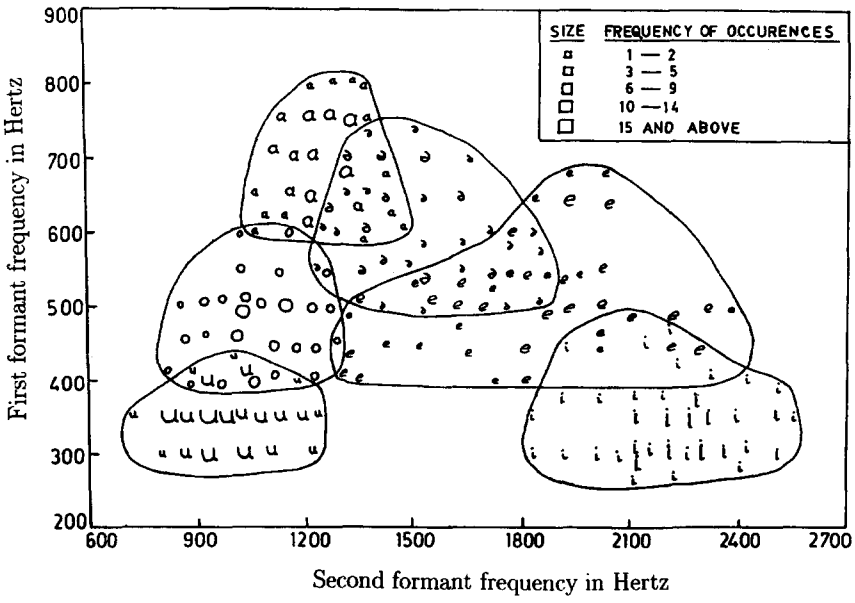

Fig. 6. Vowel data.

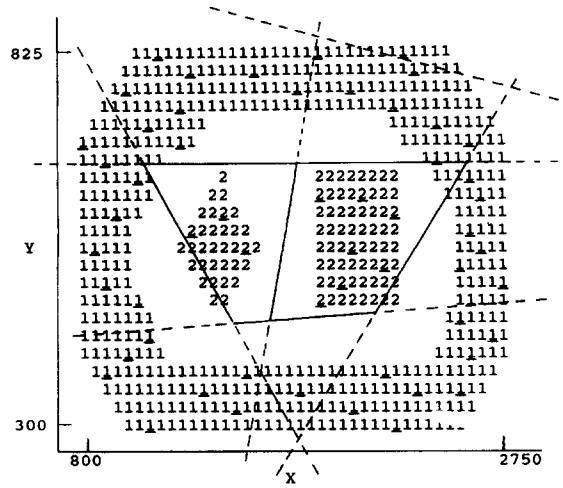

Fig. 7. Classification of 90% test data set using the final boundary generated with 10% training data set for $H = 6$. Training patterns are underscored.

region for the class. A tie is resolved arbitrarily. The example stated below will clarify this method.

Example. Let there be 8 training patterns belonging to two classes, 1 and 2, in a two-dimensional feature space F_1–F_2. Let us assume H to be 3, i.e., 3 lines will be used to classify the points. Let the training set and a set of 3 lines be as shown in Fig. 3. Each point i_j, $i = 1, 2, \ldots, 8$, $j = 1, 2$, indicates that it is the ith training point and that it belongs to class j. Let the positive and the negative sides of the lines be as shown in Fig. 3. Then, point 1_1 yields a sign string 111 since it lies on the positive side of all the three lines Line_1, Line_2, and Line_3. The corresponding linked list formed for all the eight points is shown in Fig. 4. It is to be noted that one region (denoted by sign string 110) is accidentally empty, while two regions (100 and 101) do not exist. The number of misclassifications for the example is found to be $1 + 1 = 2$, one each for sign strings 001 and 111. Note that in this example both the strings 000 and 001 are providing the regions for class 2 (assuming that the tie for region 111 is resolved in favour of class 1).

In a similar fashion, the number of misclassified cases for all the strings in the population is computed. If the number of misclassified points for a string is denoted by *miss*, then the fitness of the string is computed as $tr - miss$, where tr is the cardinality of the training set. The best string of each generation or iteration is the one which has the fewest misclassifications. This string is stored after each iteration. If the best string of the previous generation is found to be better than the best string of the current generation, then the previous best string replaces the worst string of the current generation. This is known as the *elitist strategy*, where the best string seen upto the current generation is propagated to the next generation.

2.5. Genetic operators

Selection. The *roulette wheel* selection procedure has been adopted here to implement a *proportional selection* strategy. Each string is allocated a slot of the roulette wheel subtending an angle, proportional to its fitness, at the center of the wheel. A random number in the range of 0 to 2π is generated. A copy of a string goes into the mating pool if the random number falls in the slot corresponding to the string. For a fixed population size *Pop*, this process is repeated *Pop* times, at the end of which as many strings go into the mating pool for further operations.

Crossover. A pair of strings is picked up at random and the single-point crossover operator is applied

Table 1
Correct classification (%) of the artificial data

Class	Lines						
	8	7	6	5	4	3	2
1	98.55	92.01	100.00	100.00	97.83	93.24	92.74
2	80.68	97.70	86.21	75.00	63.63	70.45	39.08
Overall %	95.41	93.00	97.41	95.62	91.83	89.24	83.40

Table 2
Correct classification (%) of the vowel data

Class	Lines						
	8	7	6	5	4	3	2
δ	39.06	26.56	41.54	29.69	0.00	0.00	0.00
a	86.25	80.00	86.42	82.50	75.00	93.75	0.00
i	85.71	87.66	85.81	87.66	76.62	86.36	87.66
u	71.85	62.96	88.97	87.41	97.04	66.67	0.00
e	72.04	91.93	80.21	68.28	69.89	88.71	0.00
o	83.85	83.85	70.37	78.26	67.70	79.50	100.00
Overall %	75.90	77.82	78.24	75.77	70.25	75.77	37.95

according to a fixed crossover probability. For this operation, a random number *cr_pt* in the range of 0 to *str_len* is generated. This is called the crossover point. The portion of the strings lying to the right of the crossover point are interchanged to yield two new strings.

Mutation. Mutation is done on a bit by bit basis (for binary strings) (Goldberg, 1989; Filho and Treleaven, 1994) according to some mutation probability *mut_prob*. So, theoretically, more than one bit may be flipped in the same string if *mut_prob* so permits. The mutation probability is varied with the number of iterations. Initially it has a high value, thus ensuring a lot of diversity in the population. As training progresses and the GA reaches the vicinity of an optimal solution, *mut_prob* is decreased. Finally, to ensure that the GA does not get stuck at a local optimum, *mut_prob* is increased again.

Termination. The process of fitness computation, selection, crossover and mutation is continued for a fixed number of iterations or till the termination condition (a string with misclassification number reduced to zero) is achieved.

3. Implementation and results

The effectiveness of the methodology described earlier has been demonstrated on both artificial data (Fig. 5) and real-life speech data (Fig. 6), both of which are not linearly separable. The artificial data set consists of 557 samples and has two classes, 1 and 2. X- and Y-coordinates represent the two features in a Euclidean space. The vowel data set corresponds to 871 Indian Telugu vowel sounds (Pal and Dutta Majumder, 1977). These were uttered in a consonant-vowel-consonant context by three male speakers in the age group of 30–35 years. The data set has two features corresponding to the first and second vowel formant frequencies and six classes $\{\delta, a, i, u, e, o\}$.

For our experiment, a fixed population size of 10 was chosen. The crossover probability was fixed at 0.8. A variable value of *mut_prob* was selected from the range [0.015, 0.333]. Initially it had a high value, gradually decreasing at first, and then increasing again in the later stages of the algorithm. 100 iterations were performed with each value of the mutation probability. The process was executed for a maximum of 1500 iterations in case it did not attain a solution with zero misclassification. The experimental results are described below when the size of the training set is considered to be 10%, i.e. *perc* = 10.

Table 1 shows the classwise recognition scores and the overall recognition score for the artificial data (Fig. 5) taking the values of H to be 8, 7, 6, 5, 4, 3 and 2 successively. Since this data set has a very small class totally surrounded by a larger class, the

Table 3
Comparative recognition scores (%) for the artificial data

Class	*k*-NN			GA
	$k=1$	$k=3$	$k=\sqrt{tr}$	$H=6$
1	100.00	99.76	92.75	100.00
2	90.91	87.50	63.63	86.21
Overall %	98.40	97.60	87.64	97.41

Table 4
Comparative recognition scores (%) for vowel data

Class	k-NN			GA
	$k=1$	$k=3$	$k=\sqrt{tr}$	$H=6$
δ	43.08	58.46	44.61	41.54
a	59.26	72.84	83.95	86.42
i	85.16	87.10	87.10	85.81
u	78.68	79.41	90.44	88.97
e	68.98	67.38	67.38	80.21
o	82.72	78.40	69.14	70.37
Overall %	73.53	75.44	75.44	78.24

classwise recognition score in this case is of greater importance than the overall score. Thus, it is seen from Table 1, $H=2$ is not a good choice in this case since the recognition of class 2 is very poor, although the overall score is reasonably good. Fig. 7 shows the decision boundaries obtained from 10% training data (for $H=6$) and their ability in classifying the remaining 90% test data. The training pattern points are underscored in the figure. This boundary was obtained on termination of training after 538 iterations when a string with no misclassified points had been found.

Table 2 shows the classification results on the vowel data. Here too, for $H=2$, the recognition score is drastically reduced, which is as expected. Again $H=6$ provided the best recognition score as in the previous case. In each case, class δ is classified very poorly as this is the class with maximum overlap. This was also found in (Pal and Dutta Majumder, 1977) where a fuzzy set theoretic classifier and a Bayes classifier were used for this problem.

A comparison of the performance of our algorithm (for $H=6$ and $perc=10$) with the k-NN classifier (Tou and Gonzalez, 1974), which is also capable of generating piecewise linear boundaries, is shown in Tables 3 and 4, for the two data sets respectively, where the value of k is chosen as 1, 3 and $\sqrt{tr}$. It is apparent from the tables that the results of the GA-based algorithm are comparable to those of the k-NN classifier, with a better performance for the overlapping vowel data set for all values of k.

4. Conclusions and discussion

A method of determining class boundaries in $\mathbb{R}^2$ using GAs has been described along with its demonstration on both artificial and real-life non-linearly separable data sets. The results are found to be comparable to those of the k-NN classifier.

An observed feature of the methodology is that increasing the number of lines (for approximating the decision boundaries) does not necessarily result in an increase of the classification performance. The reason behind this is that increasing the number of lines means tuning more and more to the peculiarities in the training set, which is not necessarily beneficial to the test set. A point to be noted here is that although one line is redundant in Fig. 7, for $H=6$, yet assuming $H=5$ produces an inferior result as seen from Table 1. A reason behind this may be the insufficient knowledge about the termination of the algorithm.

It is known in the literature (Bhandari et al., 1994) that as the number of iterations goes towards infinity, the Elitist model of GA will certainly result in the optimal string. Thus, for the problem under consideration, for infinitely many iterations, any value of H will certainly provide the minimal misclassification for that H. However, a proper choice of H, which provides the least misclassification is of crucial importance.

Proper selection of control parameters for an application of GA is still an open issue. In this work we have taken a fixed population size and crossover probability. *mut_prob* is kept variable, having a high initial value, then decreasing and finally increasing again. Ideally, this cycle of increasing and decreasing *mut_prob* should continue for a number of times. We have terminated it after just one cycle due to practical limitations.

Acknowledgements

This work was carried out when Ms. Sanghamitra Bandyopadhyay held a fellowship awarded by the Department of Atomic Energy, Govt. of India, and Prof. S.K. Pal held the Jawaharlal Nehru Fellowship. The authors acknowledge the help of Mr. S.

Chakraborty in drawing the figures. They also acknowledge the reviewer for the valuable comments on an earlier version of the article.

References

Bhandari, D., C.A. Murthy and S.K. Pal (1994). Genetic algorithm with elitist model and its convergence. *Internat. J. Pattern Recognition and Artificial Intelligence* (communicated).

Filho, J.L. R and P.C. Treleaven (1994). Genetic algorithm programming environments. *IEEE Comput.*, June, 28–43.

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

Pal, S.K. and D. Dutta Majumder (1977). Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Trans. Syst. Man Cybernet.* 7, 625–629.

Proceedings (1991). *Proceedings of the Fourth International Conference on Genetic Algorithms*. University of California, San Diego, CA.

Tou, J.T. and R.C. Gonzalez (1974). *Pattern Recognition Principles*. Addison-Wesley, Reading, MA.