APPLICATION OF A NEW GENETIC OPERATOR IN FEATURE SELECTION PROBLEMS

Nikhil R. Pal , Malay K. Kundu and Suchismita Nandi
Machine Intelligence Unit
Indian Statistical Institute
203 B. T. Road, Calcutta 700035, INDIA

ABSTRACT

Crossover is an important genetic operation that helps in random recombination of structured information to locate new points in the search space, in order to achieve a good solution to an optimization problem. The conventional crossover operation when applied on a pair of binary strings will usually not retain the total number of 1's in the offsprings to be the same as that of their parents. But there are many optimization problems which require such a constraint. In this article, we propose a new crossover technique called, "self-crossover", which satisfies this constraint as well as retains the stochastic and evolutionary characteristics of genetic algorithms. As an illustration, the effectiveness of this new technique has been demonstrated for the feature selection problem of pattern recognition.

1 Introduction

Genetic algorithms (GAs) are probabilistic heuristic search processes based on natural genetic system. They are capable of solving wide range of complex optimization problems using three simple genetic operations (selection/ reproduction, crossover and mutation) on coded solutions (strings/ chromosomes) for the parameter set (not the parameters themselves) in an iterative fashion. There are several interesting features which made GA so popular. GAs consider several points in the search space simultaneously, which reduces the chance of convergence to a local optima. GAs use only the payoff or penalty function (objective function) called, the fitness function and do not need any other auxiliary information. GAs are theoretically and empirically proven to provide robust search in complex spaces [1] even if the functions are not smooth or not continuous. Such functions are very difficult (sometimes impossible) to optimize using calculus based methods.

Normally, GAs use simple crossover which in case of binary coded solution space, may change the total number of 1's in the offsprings . Crossover and mutation are usually adequate for solving a wide class of optimization problem. However, there are families of problems which require to maintain certain constraints on the number of 1's (in case of binary coding) in the strings. Some such problems are selection of an optimal subset of features for pattern recognition application and selection of a subset of data points for designing nearest neighbour classifier. In addition there are several other combinatorial optimization problems which require some such constraints.

In this paper we show that if the genetic recombination operator is chosen properly, then GA can be competitive with the best known techniques for a large set of problems with this type of constraint. To achieve this we propose a new crossover operation. We call it "self-crossover" because of its similarity with conventional crossover. The self-crossover is performed with a single parent string instead of a pair of strings. The new operator preserves the probabilistic and evolutionary characteristics of GAs. Finally, we use GAs equipped with this new operator to solve the feature selection problem on the Mango-Leaf data. We compared the performance of GA based method with the **Fuzzy Set theoretic** feature selection method of Pal [8] and found that GA based method performs much better than the fuzzy set theoretic method.

2 Genetic Algorithms

GAs [1] are general purpose optimization algorithms which can solve problems resistant to other known optimization methods. This search technique is population based which evolves from generation to generation. The criteria of "survival of the fittest" provides evolutionary pressure for populations to grow with increasingly fit individuals. Although there are many variants, the basic mechanism of GA (conventional GA) consists of the following steps :

1. Start with an initial population (a set of strings/chromosomes).
2. Evaluation of fitness of every string and selection of appropriate candidate strings to form the mating pool.
3. Crossover and mutation.
4. Repetition of steps 2 and 3 until the system ceases to improve or some stopping criterion is reached.

Each string in the population is represented by a fixed length coded string. Although different coding schemes are possible we restrict ourselves to binary coding only. Selection or reproduction creates

0-7803-4886-9/98/ $10.00 © 1998 IEEE

the population for the next generation using a probabilistic selection process which gives a string with higher fitness a greater chance of selection. Mutation corresponds to flipping (probabilistically) one or more bits of an individual string. Mutation increases the variability of the population and ensures that the probability of attaining any point in the search space is greater than zero. The simplest implementation of crossover selects two parents (randomly) from the mating pool and then after choosing a random position each parent string exchanges its tail at that position. The resulting offsprings form the subsequent population for the next generation. Syswerda [9] presented a new crossover, *uniform crossover*, and empirically showed its superiority to one-point crossover and two-point crossover. For uniform crossover, a crossover mask is considered. The mask is a string of bits of the same size as that of the chromosomes to be crossed. If any mask bit is '1' then parents will exchange their corresponding bits, otherwise they will retain their original bits resulting in two new offsprings. Next we discuss the problem of feature ranking and a recent method for the same which we shall use for comparison.

3 Feature Extraction

Many authors [4-6] dealt with the problem of extraction of feature vectors in q-space from data in p-space where $q << p$. Feature extraction is used for two different but somewhat related problems : dimensionality reduction and visual display. Reducing p to $q << p$ reduces the space and time complexity of computations that use the extracted data. Another issue is that some of the original p features may result in confusion in the feature space. Can we do just as well or better (w.r.t. a problem under consideration) with $q < p$ new features derived from the original features? For example, transformed features can work better than the original data for purposes such as - classifier design. We show that exactly this happens with the **Mango-Leaf** data for nearest prototype (NP) classifier. Another important use of feature extraction is to get two-dimensional (q=2) displays of data for visual exploratory data analysis which helps in guessing the clustering tendencies, distributional densities through visual inspection of scatterplots of two-dimensional extracted data.

Several techniques [4] such as principal components (PC), Sammon's algorithm (SA) [5] for extracting important feature subsets are available. Fukunaga and Koontz [4] pointed out that one of the disadvantage of principal components [3] is that this technique does not necessarily produce the best features for pattern classification. Conversely, Foley and Sammon [5] observed that the best extracted features for discrimination may not serve adequately for visual display. PC and SA extract features using different criterion. PC can be used to select features that can account for a large share of variance while SA extracts features preserving interpoint distances.

Feature selection, a special type of feature extraction, on the other hand, selects a subset of the original features so that several pattern recognition tasks can be performed with the reduced set of features to a satisfactory level. For reducing the dimension of the feature space, we should eliminate those features which are less important or redundant for *discriminating* the classes. We can achieve this through ranking of features according to their importance in discrimination. In our investigation we shall call a feature subset A better than a subset B if the performance of a nearest prototype classifier with the subset A is better than that with B. There exist a number of methods for feature ranking, each having its own merits and demerits. We describe here a recent one, which has been adopted in our investigation for comparison of results.

The feature evaluation index for feature q (FEI_q) was defined by Pal[6] as

$$FEI_q = \frac{H_{qjk}}{H_{qj} + H_{qk}} \tag{1}$$

where H_{qjk} is the value of the entropy for feature q after pooling together the classes C_j and C_k; H_{qj}, H_{qk} are those for the feature q computed for C_j or C_k respectively. The lower the value of FEI_q, the higher is, therefore, the quality of the qth feature in characterizing and discriminating classes C_j and C_k. Instead of using only one feature q, FEI can be calculated [8] even for a set of features. Pal[8] further extended his earlier work [2,6] to define the average importance of a set of features S as

$$(FEI)_S^{av} = \sum_j \sum_k W_j W_k \, (FEI)_S^{(jk)} \tag{2}$$

where $W_j = \frac{n_j}{n_t}$, $W_k = \frac{n_k}{n_t}$, $n_t = \sum_j n_j$, $j,k = 1,2,....,c; k \neq j$, are weight factors. In [8] the weights were used as it was expected that two pairs of classes having different values of FEI should not contribute equally to the $(FEI)^{av}$. We have now the following remarks about $(FEI)^{av}$. $(FEI)^{av}$ should not depend on the number of points in a class but only on the structure of the classes. Moreover, in equation 9, even when $n_j + n_k = \psi$ (a constant) for two different pairs of classes, W_jW_k could be different for the two pairs. W_jW_k attains the maximum value when $n_j = n_k = \frac{\psi}{2}$. As a result of this $(FEI)^{av}$ may not reflect the actual situation. Our experimental results reported in section 6 conform to this observation.

In this context we mention some earlier attempts to solve the feature selection problem using GA. Siedlecki and Sklansky [10] used k-NN rule to find the smallest subset of features for which the classifier's performance does not deteriorate below a certain specified level. They did this by constructing a GA chromosome consisting of a binary string whose length equaled the number of features. If a bit is '1', that feature is selected for evaluating the performance of the classifier.

Kelly and Davis [11] and Punch *et al.* [12] solved the same feature selection problem using GA. Unlike Siedlecki and Sklansky they multiplied each feature by a real-valued weight and then used that weighted feature for computing distances required for implementation of k-NN classifier. GAs have been used to learn these weights. Features for which learned weights are high are considered important features and vice-versa.

We have the following remarks about the preceding GA based approaches : 1) These methods cannot be used to select a fixed number of best features *i.e.*, say q, best features. The algorithm may terminate at a point where the total number of '1's in the solution string may not be equal to q in [10]; while for other two methods [11-12] those q features having highest weights can be selected. But this can create another problem. Suppose, there is a feature which is more or less constant for all classes. For this feature whatever be the weight the classifier performance is not going to be changed. Since GAs are probabilistic search techniques, the algorithm might terminate at a point with high weight for this indifferent feature and there by indicating a false importance.

4 Self-crossover(SC)

Unlike conventional crossover mechanism, self-crossover mechanism alters the genetic information within a *single* potential string selected **randomly** from the mating pool to produce an offspring. This is done in such a manner that the stochastic and evolutionary characteristics of GAs are preserved.

Let

$$S = 00010010011001011011$$

be a string of length 20 selected from the mating pool. For self-crossover, first we select a random position p $(0 < p < L)$ and generate two substrings s_1 and s_2 : s_1 = bits 1 through p of S and s_2 = bits $p+1$ through L of S. Now we select two random positions p_1, $0 \leq p_1 \leq p$ and p_2, $0 \leq p_2 \leq (L-p)$. Then four substrings are generated as follows :

$$\begin{aligned} s_{11} &= \text{bits 1 through } p - p_1 \text{ of } s_1 \\ s_{12} &= \text{bits } (p - p_1 + 1) \text{ through } p \text{ of } s_1 \\ s_{21} &= \text{bits 1 through } L - p - p_2 \text{ of } s_2 \\ s_{22} &= \text{bits } (L - p_2 + 1) \text{ through } L \text{ of } s_2 \end{aligned}$$

Using operations similar to crossover we generate $S^1 = s_{11} \mid s_{22}$ and $S^2 = s_{21} \mid s_{12}$. Finally, the self-crossovered offspring of S is generated as $S_1 = S^1 \mid S^2$. It is easy to see that number of 1's in S and S_1 is the same. We now explain it with the example string S of length 20.

$$S = 00010010011001011011$$

A random position, $p = 9$, is selected for splitting the string into two substrings (s_1, s_2) as follows :

$$\begin{aligned} s_1 &= 000100100 \\ s_2 &= 11001011011 \end{aligned}$$

Now two random positions, $p_1 = 4$ and $p_2 = 7$, are selected for s_1 and s_2 respectively. After splitting s_1 and s_2 at $4th$ and $7th$ position, respectively we get,

$$\begin{aligned} s_{11} &= 00010 \\ s_{12} &= 0100 \\ s_{21} &= 1100 \\ s_{22} &= 1011011 \end{aligned}$$

The two new substrings S^1 and S^2 are then obtained as :

$$\begin{aligned} S^1 &= 000101011011 \\ S^2 &= 11000100 \end{aligned}$$

Finally, the offspring (S_1) is generated by concatenating S^1 and S^2 as :

$$S_1 = 000101011011\,11000100$$

Thus, self-crossover exchanges substrings s_{12} and s_{22}. If the parent string consists of all 0's or all 1's, the offspring generated through self-crossover will resemble its parent because of the underlying constraint on the total number of 1's in the string. It is also clear that if we do not start GA with a all '1' or all '0' string, GA with self-crossover technique, will never generate such strings as offsprings. So, self-crossover will evolve new offsprings as iterations go on.

We can see very well that mutation is not effective in producing such constrained offsprings. But self-crossover can regenerate any lost genetic information. So we may not need mutation when we use the new technique in constrained GA application.

At the first sight, it might appear that self-crossover is nothing but a parallel random search, but this is not the case because of two reasons. Self-crossover is done only on a randomly selected subset of strings and self-crossover does not alter the substring s_{11}. It exchanges, only s_{22} and s_{12}. Consequently, the evolutionary characteristics of GA are preserved. The similarity between the parents and offsprings will be more if we take $p_1 = p_2 = p'$ (say) = a random number selected between 1 and Min $(p, L-p)$; i.e., $0 < p_1 = p_2 = p' < Min\ (p, L-p)$. In this case, the bits in positions 0 through p' and bits from $p+1$ through $L-p'$ will remain unaltered. Consequently, the evolution of the process will be faster.

5 Results and Discussion

We used Mango-Leaf data which is a $p = 18$ dimensional data with 166 points. It has three classes representing three kinds of mango. The feature set consists

of measurements like - area(A), perimeter(Pe), maximum length(L), maximum breadth(B), petiole(P), length + petiole(L+P), length / petiole(L/P), length / maximum breadth(L/B), (L+P)/B, A/L, A/B, A/Pe, upper midrib / lower midrib, upper Pe / lower Pe and so on. The terms upper and lower are used with respect to maximum breadth position. Table I shows the comparison of our results with results reported by Pal[8] for Mango-Leaf data. In table I the column named "match" indicates total number of data points whose derived classification and actual classification agreed *i.e.*, the value of the objective function.

Table I : Results with the Mango-leaf data

No. of features extracted	Our result with self-crossover			Results with methods in [8]	
	Feature-subset	No. of Generation	Match	Feature-subset	Match
1	9	9	103	5	91
2	14,15	24	123	11,15	91
3	7,14,15	19	123	5,11,15	96
4	2,8,12,16	19	125	-	-
5	2,8,12,16,18	19	125	-	-
6	2,4,12,14,16,17	34	125	-	-
18	(Full set)	-	118	-	-

Table I reveals several interesting things : First, more features are not always good, some features may create confusion in the decision surface. For example, with all eighteen features, the NP classifier can recognize only 118 points, while with just two features the recognition score goes upto 122 or 123. Our GA based feature selection clearly selects better features than the fuzzy set theoretic method of Pal[8]. Pal's method could get only 96 match with three features. The best recognition score is obtained with only four features and the number of iterations required for this is 19 which is much less than exhaustive search *i.e.* $3060 = \begin{pmatrix} 18 \\ 4 \end{pmatrix}$. Details of the algorithm and more results on it are available in [7].

6 Conclusions

There is a class of optimization problem which requires the number of '1's in each string to be constant. Conventional crossover / mutation does not guarantee this. We have introduced a new crossover operation named *self-crossover* which preserves this constraint. Moreover, this new operator plays the combined role of mutation and as well as of the conventional crossover. The effectiveness of GAs equipped with self-crossover is demonstrated for solving the feature selection problem. Experimental investigation shows that the new operator is very effective for the feature selection problem.

7 References

1. D. E. Goldberg, Genetic Algorithms : Search, Optimization and Machine Learning. : Addison Wesley Publishing Company, Inc. (1989).
2. Sankar K. Pal and B.Chakraborty, Intraclass and interclass ambiguities (fuzziness) in feature evaluation, *Pattern Recognition Letters*, **2**, 275-279 (1984).
3. R. A. Johnson and D. W. Wichern, Applied Multivariate Statistical Analysis, Prentice Hall, NJ, (1988).
4. K. Fukunaga and W. L. G. Koontz, Application of the Karhunen-Loeve expansion to feature selection and ordering, *IEEE Trans. Comput.*, **19**, 311-318, (1970).
5. D. H. Foley and J. W. Shammon, An optimal set of discriminant vectors, *IEEE Trans. Comput.*, **24**, 271-278, (1978).
6. S. K. Pal and B. Chakraborty, Fuzzy Set Theoretic Measure for Automatic Feature Evaluation, *IEEE Trans. Syst. Man Cybern.* SMC-16 (5) , 754-760 (1986)
7. N. R. Pal and S. Mitra and M. K. Kundu, Self-Crossover- a new genetic operator and its application to feature selection, *International Journal of Systems Science*, **29**, 207-212 (1998).
8. S. K. Pal, Fuzzy set theoretic measures for automatic feature evaluation : ii, *Information Sciences*, **64**, 165-179, (1992).
9. Gilbert Syswerda, uniform crossover in genetic algorithms,*Proceedings of the third international conference on Genetic Algorithms*, USA, 2-9, (1989).
10. W. Siedlecki and J. Sklansky, A note on genetic algorithms for large-scale feature selection, *Pattern Recognition Letters*, **10**, 335-347, (1989).
11. J. D. Kelly,Jr. and L. Davis, A hybrid Genetic Algorithm for classification , *International Joint Conference on Artificial Intelligence , Darling Harbour Australia Sydney*, **2**, 645-650, (1991).
12. Punch *et al.*, Further Research on Feature Selection and Classification using Genetic Algorithms, *Proceedings of the fifth international conference on Genetic Algorithms*, University of ILLINOIS at URBANA - CHAMPAIGN, 557-564, (1993).