EXTENDING GENETIC PROGRAMMING FOR MULTI-CLASS CLASSIFICATION BY COMBINING K-NEAREST NEIGHBOR

Liang Zhang, Lindsay B. Jack and Asoke K. Nandi, Senior Member, IEEE

Signal Processing and Communications Group,
Department of Electrical Engineering and Electronics,
The University of Liverpool, Brownlow Hill, Liverpool L69 3GJ, UK.
liang@liv.ac.uk

ABSTRACT

Genetic programming has seldom been used for multi-class classification purposes. Previously, it was achieved by separating the output manually for different classes or expanding an n-class problem to n two-class problems. In this paper, we present a new approach to solving multi-class problem by using genetic programming for feature generation, and applying the K-nearest neighbor as a classifier. The results are comparable with other classifiers.

1. INTRODUCTION

Genetic Programming (GP) has been increasingly applied in pattern recognition scenarios, where GP is particularly suitable for binary classification, because of its single output structure. The division between negative and non-negative numbers acts as a natural boundary for a distinction between the two classes. Zhang et al. [1] extended GP for multi-class classification purpose by setting different thresholds on the output for different classes. However, the thresholds, which are set by the user, are problem-dependent, hence optimised thresholds are difficult to find; furthermore, for some problems, some classes are easier to map to different segments of the output because of the nature of the problem. Therefore, there is a need to automatically define the area for each class. Another approach to multi-class classification using GP was explored by Kishore et al. [2] and Muni et al. [3]. They modeled an n-class problem as n two-class problems, and each GP classifier is evolved as a discriminant function for a class. This method is applicable when the problem has only a few classes. However, when there are many classes, this approach will expand the problem drastically, and much more time will have to be spent in training.

The K-Nearest Neighbors (KNN) algorithm is one of the simplest pattern recognition approaches, which itself is a multi-class classifier [4]. We apply KNN to the outputs of the GPs. Using GP for feature generation, a GP output is treated as a new feature describing the problem, based on which KNN is used to separate the classes. This way, each individual is deemed a feature, and the process of training turns out to be an evolution of features. After training, some features are chosen for KNN to classify the unseen samples.

Used in conjunction with KNN, GP is able to separate n ($n > 2$) classes; on the other hand, GP selects features for KNN, so that irrelevant features can be eliminated rather than confusing the KNN, since the sample data available are insufficient for good generalisation due to their high dimensionality.

2. TEST PROBLEMS

The proposed method was evaluated on three real-world data sets. Two of them are chosen from the UCI repository [5], and the other one is of roller bearing fault detection [6]. Table 1 shows the properties of the three data sets, including one dual-class problem and two multi-class problems.

Data set	No. Features	No. Class	No. Samples
Bearing	156	6	2880
WDBC	30	2	569
Wine	13	3	178

Table 1. Properties of the test data sets

The data for the roller bearing fault detection was extracted from vibration signals that were taken from an experiment on a small test rig, which simulates an environment for running roller bearings. Six conditions were tested and recorded. There are two normal conditions – a brand new normal bearing and a worn but undamaged condition; and four fault conditions – inner race fault, outer race fault, rolling element fault, and cage fault. There are 2880 samples in the data set, equally distributed among the six classes.

The 30 features of Wisconsin Diagnostic Breast Cancer (WDBC) data are computed from a digitised image of a fine

needle aspirate of a breast mass. They describe characteristics of the cell nuclei present in the image.

The Wine recognition data represent the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

3. IMPLEMENTATIONS OF GP

3.1. Non-Destructive Approach

In order to limit code growth of the GP and keep the diversity of the population, a non-destructive approach is applied. When a modifying operator is in operation, once an offspring has been created, the offspring o and its parent p (the parent tree that shares the same root node with the offspring) are then evaluated and set with fitness f_o and f_p. If $f_o > f_p$, the offspring is added to the new generation; otherwise, if $f_o \leq f_p$, neither the offspring nor the parent is put to the new generation. After a fixed number of crossovers and mutations, some offspring (fewer than the population size) will be sent to the new generation. After that, reproduction will fill the remaining places of the new generation using the same selection method as for modifying operators, one by one until the new generation has enough members to proceed. No individual would be selected twice for reproduction, no matter how high the fitness. This mechanism helps avoid more than one copy of an individual, which may cause premature convergence.

3.2. Adaptive Subtree Creation

Subtree creation's role in subtree mutation is an important issue as described in [7]. In order to reduce the influence of subtree creation to the size of population, adaptive subtree creation is applied in this paper. This is achieved by keeping the average size of created subtrees as close as possible to the average size of deleted subtrees, in terms of the whole population. When subtree mutation modifys an individual tree x, the subtree deletion point is chosen randomly with no preference for input nodes. During subtree creation, the probability for growing a function node p_f is given by Equation 1, where d_x is the depth of the current growing point in the subtree being created, and s is the controlling factor. The value of s is first set to 4 initially. Assigning probabilities using this equation, the subtree to be created is limited to the depth of s. After subtree creation, the difference in size between the offspring and the parent is calculated. Such differences for every mutation operation over the entire population are added up. If this sum of differences is less than zero, indicating that mutation has brought a bias towards smaller programs, then mutation will create larger subtrees for the next generation by dividing s by 0.9, hence assigning more probability to grow function nodes in subtree creation; otherwise smaller subtrees are created by multiplying s with 0.9. Thus, the subtree creation is adapted to the size of deleted subtrees, consequently is adapted to the size of the whole population.

$$p_f = 1 - \frac{d_x}{s} \tag{1}$$

3.3. Fitness Evaluation

When calculating the fitness, the number of wrong classification of individual x (s_x) over the training set is first obtained. Then fitness f_x is given by Equation 2.

$$f_x = \frac{1}{\sqrt{s_x + 1}} - pn_x \tag{2}$$

where pn_x is the penalty term based on the program size, n_x is the number of nodes that the program contains, representing program size, and p is weighting factor, the choice of which is arbitrary. Intuitively, small positive values should be used, so that the pn_x component is negligible compared to f_x. This can be achieved by choosing p as in Equation 3 ([8]).

$$p << inf_x[\frac{f_x}{n_x}] \tag{3}$$

where infimum is taken over a set of best solutions evolved without any parsimony pressure.

3.4. GP Parameters

The non-destructive approach and the adaptive subtree creation as described in Section 3.1 and 3.2 were applied. The selection method was roulette. The best individual (elitist) of each generation was copied to the next generation. A node penalty of 0.001 was used. Twenty runs were carried out for each problem, and the average performance on the test data sets were obtained. Table 2 gives the other parameters for the three problems. The function pool is listed in Table 3.

	Bearing	WDBC	Wine
No. Generations	500	200	400
Population Size	200	500	400
No. Crossover	80	200	160
No. Mutation	80	200	160
No. Neighbors (K)	5	5	3

Table 2. GP parameters for the three problems

One-Input Operators			Two-Input Operators	
sin	tan	abs	addition	power
cos	tanh	log	subtraction	greater than
arcsin	exp	step	multiplication	less than
arccos	sqrt	ramp	division	average

Table 3. GP functions list

4. EXPERIMENTS

4.1. Data Preparation

In each experiment, the data set was first randomly shuffled, then two third of all the available data was used as a training set and the remaining third was used as the test set. This is because all the other three classifiers use two third of the data for training and validation, and one third as test set. All the data was then normalised within each feature before being used in the experiments. The normalisation is based on equation (4), where $m_f^{(1)}$ is the mean value of the feature vector f, and σ_f is the standard deviation of the feature vector f.

$$f_i^{'} = \frac{f_i - m_f}{\sigma_f} \tag{4}$$

4.2. Training

10-fold cross validation was conducted for training, because of lack of available data. For each problem, the training set was divided into 10 subsets; 9 of these were used for training and the remaining subset was used for validation. During the evaluation of each individual, 10 experiments were conducted for each subset, each sample of which was classified by KNN based on the other 9 subsets. The number of wrong classification of the 10 experiments was then summed up for calculating the fitness of the individual.

Each GP tree in the population is a newly created feature; the process of training is the evolution of these features. During training, each feature (individual) is optimised alone. After training, some GP trees are selected from the entire population to form a final classifier, i.e., some GP generated features are then sent together to KNN. In other words, during training, KNN processes one dimensional data, for the sake of simplicity in evaluation; while after training, KNN processes multi dimensional data, to search best performance and generality.

The non-destructive approach mentioned earlier in this paper makes possible of the use of several features in the population. If GP works in the original destructive way, the performance of most offspring from crossover and mutation are poorer than their parents, some of them are completely useless. Therefore in the population, there will not be many available features to choose for KNN. However in the non-destructive way, only those offspring with improved performance are kept. At the end of evolution, all the individuals have relatively high performance, which have different structures, because diversity is better kept in the non-destructive way. Hence, most GP trees in the final population are effective features. The KNN has a wide choice to find the best combination.

For the selection of the GP-created features, genetic algorithm (GA) [9] could have been used for its proved ability. However initial trials of random selection provided similar and high performance, which indicates that further search by GA will not significantly benefit. Hence GA was not in use. Instead, the way features are selected is that a large number of combinations of different numbers of features are tried, and the combination with the best performance on the training set is chosen as the final classifier. The number of features from 2 to 40 are tried, with each number of features 50 combinations have been evaluated. The method to select individuals to form a classifier is roulette, the same as for crossover and mutation. The reason why a multi-feature method is used is that for the performance on the training set, the best single GP is often equal or better than the best multi-feature classifier; however, for the performance on the test set, in most cases, the best single GP is poorer than the best multi-feature classifier. This is consistent on all the three problems, showing that the single GP is less general than the multi-feature classifier. The number of features in the final classifier varies, even for different runs of the same problem. This is due to the fact that randomicity happens everywhere in GP process and the later feature selection.

4.3. Choosing of K

Different numbers of the neighbors (k) for KNN have been initially tested, the k with the best training performance was chosen for each problem.

5. RESULTS

The proposed method was tested for 20 runs for each problem. The average performance of prediction on the test data are compared with that of three other classifiers: ANN, SVM and KNN, as shown in Table 4. The results of Wine and WDBC data for SVM and KNN are cited from [10], and the results of Bearing data for ANN and SVM are cited from [6]. Because the bearing data has a large dimension (156), a genetic algorithm (GA) was used to do feature selection for ANN and SVM. GP has derived the ability of feature selection from GA.

The results in Table 4 are similar for the four classifiers. For WDBC and Wine data sets, GP-KNN has better results

than KNN, but not as good as SVM. However for the Bearing problem, GP-KNN offers the best performance.

	Bearing	WDBC	Wine
ANN	99.1*	N/A	N/A
SVM	99.4*	96.5	96.8
KNN	N/A	95.6	94.7
GP-KNN	99.5	95.9	95.2

Table 4. The average performance of GP combined with KNN (GP-KNN) on the test data are compared with that of other classifiers: ANN, SVM and KNN. The results with * are achieved by using GA during training for feature selection. N/A: results for that algorithm were not considered in the papers that are cited.

Figure 1 shows the sample distribution on a single GP solution, with the problem of bearing fault detection (6 classes). As one can see, the six classes are located on six segments of the GP output. The location are chosen automatically, rather than manually.

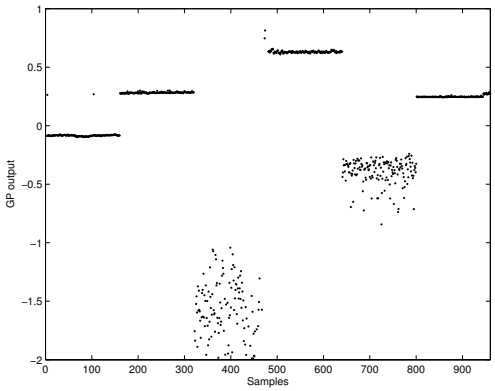

Fig. 1. Sample distribution on a GP output for the bearing fault detection problem (6 classes).

6. CONCLUSION

Genetic programming was successfully extended for multi-class classification in this work. Rather than in its traditional way as a classifier, GP was used in a manner for feature generation. KNN was applied to recognise the classes using some of the GP generated features selected from the population. GP does feature selection on the original features, and automatically generates new features for KNN. Thus it is not necessary to train the GP for each class separately, or set thresholds for different classes on the GP output, which is hard to achieve good performance for relatively difficult problems.

The non-destructive approach of GP modifying operators maintains the diversity of the population better than the traditional destructive way, and make all individuals have relatively high performance, which provides enough choices to apply more GP generated features for KNN.

Experiments have given comparable results with three other widely used classifiers, ANN, SVM and KNN, on two multi-class problems: roller bearing fault detection and Wine recognition, and one dual-class problem, Wisconsin Diagnostic Breast Cancer (WDBC). The results show that this novel classifier is successful.

Acknowledgment

Dr L. B. Jack is supported by the Biotechnology and Biological Sciences Research Council, UK.

7. REFERENCES

[1] M. Zhang, V. B. Ciesielski, and P. Andreae, "A domain-independent window approach to multiclass object detection using genetic programming," *EURASIP Journal on Applied Signal Processing*, , no. 8, pp. 841–859, July 2003.

[2] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal, "Application of genetic programming for multicategory pattern classification," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 242–258, 2000.

[3] D. P. Muni, N. R. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 183–196, 2004.

[4] W. Zheng, C. Zou, and L. Zhao, "Face recognition using two novel nearest neighbor classifiers," in *ICASSP'04*, 2004.

[5] C.L. Blake and C.J. Merz, "UCI repository of machine learning databases," 1998.

[6] L. B. Jack, *Applications of Artificial Intelligence in Machine Condition Monitoring*, Ph.D. thesis, The University of Liverpool, 2000.

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

[8] J. P. Rosca, "Analysis of complexity drift in genetic programming," in *Genetic Programming 1997: Proceedings of the Second Annual Conference*. July 1997, pp. 286–294, Morgan Kaufmann.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning*, Addison Wesley, 1989.

[10] B. Moghaddam and G. Shakhnarovich, "Boosted dyadic kernel discriminants," in *Advances in Neural Information Processing Systems (NIPS)*, December 2002, vol. 15.