Using Genetic Programming for Multiclass Classification by Simultaneously Solving Component Binary Classification Problems

Will Smart and Mengjie Zhang

School of Mathematics, Statistics and Computer Sciences,
Victoria University of Wellington, P. O. Box 600, Wellington, New Zealand
{smartwill, mengjie}@mcs.vuw.ac.nz

Abstract. In this paper a new method is presented to solve a series of multiclass object classification problems using Genetic Programming (GP). All component two-class subproblems of the multiclass problem are solved in a single run, using a multi-objective fitness function. Probabilistic methods are used, with each evolved program required to solve only one subproblem. Programs gain a fitness related to their rank at the subproblem that they solve best. The new method is compared with two other GP based methods on four multiclass object classification problems of varying difficulty. The new method outperforms the other methods significantly in terms of both test classification accuracy and training time at the best validation performance in almost all experiments.

1 Introduction

Object classification problems abound in daily life, and a computer system that can solve object classification problems is very desirable. The advantages of using a computer to solve such problems, over a human expert, include lower cost, higher speed and higher reliability in the face of large throughput. However, building automatic computer systems for object and image classification tasks that are reliable and can achieve desirable performance is very difficult.

GP research has considered a variety of kinds of evolved program representations for classification tasks, including decision tree classifiers and classification rule sets [1, 2, 3]. Recently, a new form of classifier representation – numeric expression classifiers – has been developed using GP [4, 5, 6, 7]. This form has been successfully applied to real world classification problems such as detecting and recognising particular classes of objects in images [5, 6, 8], demonstrating the potential of GP as a general method for classification problems.

The output of a numeric expression classifier is a single numeric value (the program output), and problems arise when attempting to convert this value into a class label. For binary problems, one reasonable solution is to assign one class if the program output is negative, and the other otherwise [4, 5, 6, 9]. However, the problem is much more complicated when three or more classes exist (multiclass

M. Keijzer et al. (Eds.): EuroGP 2005, LNCS 3447, pp. 1–13, 2005.

problems), as multiple boundaries need to be found to divide the numeric space into three or more class regions.

Statically-assigned class boundary methods have been used widely, but are seen to unnecessarily constrain programs, leading to long search time and low final accuracy [4, 7]. In previous research a *probabilistic* method has been used to avoid the setting of class boundaries [10], however this method still constrains each program to solve the entire problem, even when the problem has many classes.

To avoid the above problems, the multiclass classification task may be decomposed into many binary tasks [4]. However, in the past each binary task requires a separate GP evolution, leading to a long total time for evolution even though each evolution is quite short [4].

The goal of this paper is to construct a method to decompose a multiclass classification problem into a number of two-class (binary) subproblems, solve all these subproblems in a single GP run, then combine the binary subproblems to solve the whole multiclass problem. A secondary goal is to evaluate the new method on a variety of problems of varying difficulty, comparing it with two other GP based methods.

This paper is organized as follows. In section 2 two existing fitness functions for comparison with the new method are presented. In section 3 the new method is described. In section 4 the data sets and settings used for experiments are given. In section 5 the results of experiments are presented and discussed. In section 6 conclusions are drawn, and some directions are given for future research.

2 Two Existing Fitness Functions

The new approach described in this paper will be compared with two existing fitness functions: Program Classification Map (PCM) [7, 11] and Probabilistic Multiclass (PM) [10].

2.1 Program Classification Map

In PCM, the floating-point output of a program is converted directly into a class label, depending on the numeric region it falls into. Thresholds are set at even spacing (one unit) on the number line from some negative number to the same positive number. For an N class problem, there will be $N - 1$ thresholds. The classes are assigned to the regions before, between and after the thresholds, in order of magnitude. For example, figure 1 shows the numeric regions of a five class problem. The fitness of a program is found by subtracting the training set accuracy from 100%.

2.2 Probabilistic Model of Program Output

Based on the feature values of the training examples for a particular class, the mean and standard deviation of the program output values for that class can

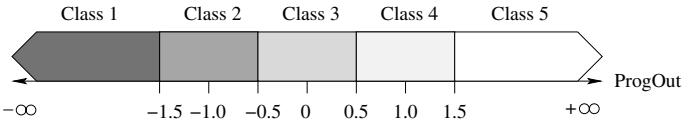

Fig. 1. Program Classification Map (PCM) for a five-class problem

be calculated. In this way, we can attain the mean and standard deviation for each class. These program statistics are compared in order to get a fitness value indicating how well the program separates the classes.

Probabilistic Binary. Figure 2 shows three examples of normal curves that may be gained by modeling three program's results on the training examples in the binary classification problems. In the figure, the leftmost program's normal curves are substantially "overlapped", so there is a high probability of misclassification. The rightmost program's normal curves are well "separated", so there is a low probability of misclassification and this program represent a good classifier.

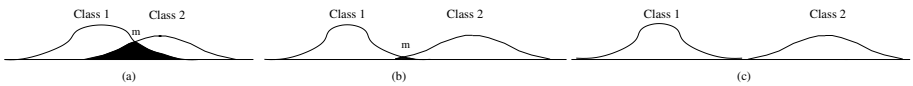

Fig. 2. Example normal distributions for a binary problem. (a) a bad discerner between the two classes, (b) an acceptable discerner, (c) a good discerner

In the binary problem case, equation 1 is used to determine the distribution distance (d) between the classes in standard deviations.

$$d = 2 \times \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{1}$$

where μ_i and σ_i are the mean and standard deviation of the program outputs for class i in the training set. For programs that distinguish between the two classes well, the distance d will be large. In such a case, the probability of misclassification would be small as the distribution overlap occurs at a high standard deviation.

To be consistent with the PCM method, we convert the distribution distance measure d to a standardised fitness measure ds, as shown in equation 2, which indicates the misclassification of a genetic program in the binary classification problem.

$$ds = \frac{1}{1 + d} \tag{2}$$

Probabilistic Multiclass (PM). In the PM method, the fitness of a program for the multiclass (N-class) classification problem is defined as the sum of the

standardised distribution distance of all binary classification problems, as shown below:

$$\text{F}itness(PM) = \sum_{i=1}^{C_N^2} ds_i \tag{3}$$

For instance, for a four class problem there are $C_4^2 = 6$ binary classification subproblems. For each subproblem, we can calculate its ds value. The fitness of the whole four class problem is the sum of all the six ds values.

3 Communal Binary Decomposition

For presentation convenience, the new approach developed in this paper is called Communal Binary Decomposition (CBD).

CBD also uses a *probabilistic* method to model the outputs of programs, as PM does. However, while a solution program in PM must separate the distributions of all classes in the multiclass problem, in CBD each program only needs to separate two classes. In CBD the program's fitness depends on its performance at separating just one pair of classes for a particular binary classification subproblem.

The separation of the problem into many two-class (binary) problems is similar to Binary Decomposition [4]. However in CBD all the problems are solved in one evolution using a multi-objective fitness function, which is why it is called "communal".

3.1 Getting Fitness

In each generation, the following steps are taken to calculate the fitness of a program:

1 For each pair of classes in a binary classification problem, calculate and store the separation distance d (equation 1).
2 For each pair of classes in a binary classification problem, sort the programs in the population based on the separation distance values in a descending order.
3 Calculate the fitness of each program based on its position in the list of programs where the program achieves the best performance (position) in all the binary classification problems.

Figure 3 shows an example of using this method to evaluate the fitness of four programs in a three class problem.

The main table in the figure lists the separation distances of all programs on all binary problems. Below the main table, the entries for each binary problem are sorted from best program to worst. For example, for the binary problem of separating class one from class two, program D was best at a distance of 2.01, so it is first in the list. Then each program is assigned a fitness to the position in the sorted list where it achieves the best performance (occurs earliest, as indicated by arrows).

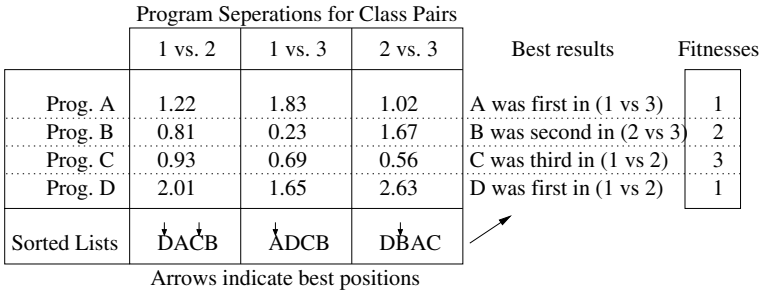

Fig. 3. Evaluating the fitness of four example programs in a three class system

With this fitness function, each program is encouraged to excel in separating one pair of classes, although the program is free to do so to any pair it chooses. Any program is free to separate more than one pair of classes, but only the pair where the program performs the best is used to calculate the fitness.

3.2 Solving the Multiclass Problem

In order that the multiclass problem is solved, a group of *expert* programs is assembled during evolution. An expert program is stored for each of the binary problems.

In each generation, for each binary problem, the program in the population that has the best separation is compared to the expert for the binary problem. If it is found to be better than the expert, it replaces the expert.

If the standardised distribution distance value ds (equation 2) for the (possibly new) expert program falls below (is better than) a parameter $solveAt$, then the binary problem is marked as solved. A solved binary problem differs from an unsolved problem in that programs are no longer encouraged to solve it. Solved binary problems are not included in the calculation of the best position for each program (discussed in section 3.1). When all binary problems have been solved, the problem is considered solved.

Note that the best program at separating classes of a solved binary problem is still found, for comparison with the current expert at the problem. As such, experts of solved binary problems can still be replaced and improved upon.

3.3 Combining CBD Expert Programs to Predict Class

To find the accuracy of the system on the test set or validation set, the experts gained for all the binary problems are combined to predict the class of each object image in the test set or validation set.

Equation 4 is used to find the probability density at points in the normal curve.

$$P_{e,c,o} = \frac{e^{\left(\frac{-(res_{e,o}-\mu_{e,c})^2}{2\sigma_{e,c}^2}\right)}}{\sigma_{e,c}\sqrt{2\pi}} \tag{4}$$

where $P_{e,c,o}$ is the probability density calculated for the expert program e using the normal distribution for class c and features for object o. $res_{e,o}$ is the output result of evaluating expert program e on object o. $\mu_{e,c}$ and $\sigma_{e,c}$ are the mean and standard deviation, respectively, of the output results of the expert program e for all training object examples for class c.

Using the function in equation 4, equation 5 shows a probability value of object o belonging to class a in the class pair (a, b) (binary classification problem), and equation 6 shows the same value expressed in a different form.

$$\frac{P_{e(a,b),a,o}}{P_{e(a,b),a,o} + P_{e(a,b),b,o}} = Pr(\mathrm{c}ls = a | cls \in \{a, b\}) \tag{5}$$

$$= \frac{Pr(\mathrm{c}ls = a \cap (cls \in \{a, b\}))}{Pr(cls \in \{a, b\})}$$

$$= \frac{Pr(\mathrm{c}ls = a)}{Pr(\mathrm{c}ls \in \{a, b\})} = \frac{Pr(\mathrm{c}ls = a)}{Pr(\mathrm{c}ls = a) + Pr(\mathrm{c}ls = b)} \tag{6}$$

where the expert for discerning class a from class b is called $e(a, b)$ and $Pr(x)$ is the probability that the condition x is true.

To obtain the probability of any object in the test set belonging to class c with the expert programs in a multiclass (N-class) problem, we consider the inverted sum of all binary classification subproblems associated with class c that the multiclass problem can be decomposed into:

$$\sum_{i=1}^{c-1} \frac{1}{Pr(cls = c | cls \in \{i, c\})} + \sum_{j=c+1}^{N} \frac{1}{Pr(\mathrm{c}ls = c | \mathrm{c}ls \in \{c, j\})}$$

$$= \sum_{i=1}^{c-1} \frac{Pr(\mathrm{c}ls = i) + Pr(\mathrm{c}ls = c)}{Pr(\mathrm{c}ls = c)} + \sum_{j=c+1}^{N} \frac{Pr(\mathrm{c}ls = c) + Pr(\mathrm{c}ls = j)}{Pr(\mathrm{c}ls = c)}$$

$$= \frac{1 + (N-2) \cdot Pr(\mathrm{c}ls = c)}{Pr(\mathrm{c}ls = c)} = N - 2 + \frac{1}{Pr(\mathrm{c}ls = c)}$$

Note that the sum of the probability of a particular object belonging to each of all the possible classes is equal to one, i.e. $\sum_{i=1}^{N} Pr(cls = i) = 1$.

Accordingly, the probability of an object that the GP system classifies to class c is $Pr(\mathrm{c}ls = c)$ or p_c:

$$p_c = \frac{1}{\sum_{i=1}^{c-1} \frac{Pr(cls=i)+Pr(cls=c)}{Pr(cls=c)} + \sum_{j=c+1}^{N} \frac{Pr(cls=c)+Pr(cls=j)}{Pr(cls=c)} - (N-2)}$$

$$= \frac{1}{\sum_{i=1}^{c-1} \frac{1}{Pr(cls=c|cls\in\{i,c\})} + \sum_{j=c+1}^{N} \frac{1}{Pr(cls=c|cls\in\{c,j\})} - (N-2)} \tag{7}$$

Based on equations 7 and 5, we can calculate the probability of an object being of any class. The class with the highest probability for the object is used as the class the GP system classified into.

4 Data Sets and Evolutionary Settings

4.1 Image Data Sets

We used four data sets providing object classification problems of increasing difficulty in the experiments. Example images are shown in figure 4.

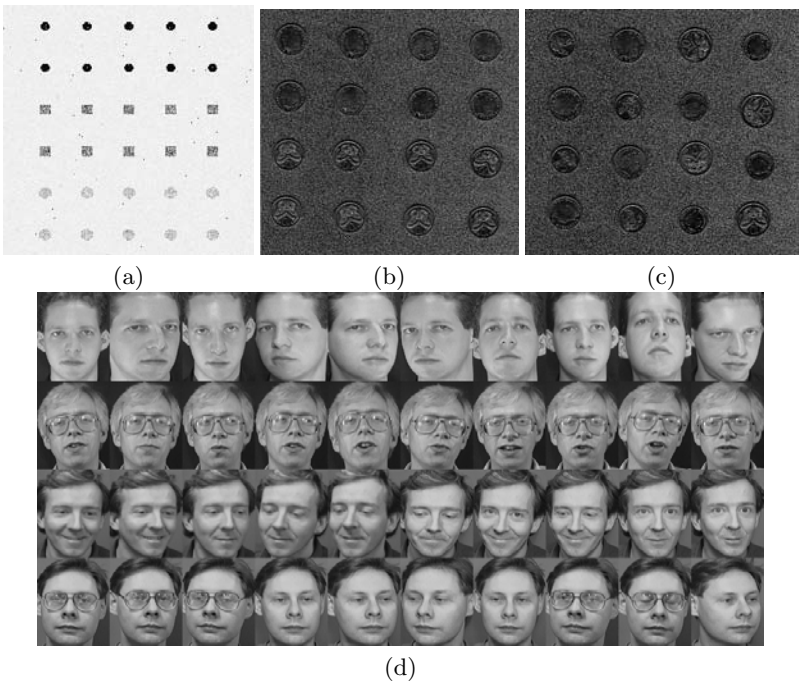

Fig. 4. Dataset examples: (a) Shapes, (b) 3-class coins, (c) 5-class coins, and (d) Faces

The first set of images (figure 4a) was generated to give well defined objects against a relatively clean background. The pixels of the objects were produced using a Gaussian generator with different means and variances for each class. Three types of shape were drawn against the light grey background: black circles, dark grey squares and light circles. The *three-class shape* data set was created by cutting out 720 objects (also called cutouts), from these images.

The second and the third sets of images (figure 4b and 4c) contain scanned New Zealand coins. The coins were located in different places with different orientations and appeared in different sides (head and tail). In addition, the background was quite cluttered. We need to distinguish different coins, with different sides, from the background. Two data sets were created from these images, one with three classes and one with five classes. The *three-class coin* data set has 576 cutouts that are either 10c heads, 10c tails or background. The *five-class coin* data set has 480 cutouts that are either 5c heads, 5c tails,

10c heads, 10c tails or background. The classification problem in the coin data sets is much harder than that of the shape data set. The problem is very hard due to the cluttered background and similarity of the classes as well as the low resolution of images.

The fourth data set consists of 40 images of four human faces (figure 4d) taken at different times, varying lighting slightly, with different expressions (open/closed eyes, smiling/non-smiling) and facial details (glasses/no-glasses). These images were collected from the first four directories of the ORL face database [12]. All the images were taken against a dark homogeneous background with limited orientations. The task here is to distinguish those faces into the four different people.

For the shape and the coin data sets, the objects were equally split into three separate data sets: one third for the training set used directly for learning the genetic program classifiers, one third for the validation set for controlling overfitting, and one third for the test set for measuring the performance of the learned program classifiers. For the faces data set, due to the small number of images, the standard ten-fold cross validation technique was applied.

4.2 GP Environment and Settings

In all experiments, programs used the tree-structured representation with all nodes returning single, floating-point numbers. Reproduction, mutation and crossover were used as the genetic operators. Initial programs and the subtrees created in the mutation operator were generated by the half-and-half method. A tournament selection mechanism was used with a tournament size of three.

The GP parameters used in experiments are shown in table 1.

Table 1. Parameters used for GP training for the three data sets

Parameter Names	Shapes	coins	faces	Parameter Names	Shapes	coins	faces
population-size	500	500	500	reproduction-rate	10%	10%	10%
initial-max-depth	5	5	5	cross-rate	60%	60%	60%
max-depth	7	7	7	mutation-rate	30%	30%	30%
max-generations	40	40	40	cross-term	30%	30%	30%
object-size	16×16	70×70	92×112	cross-func	70%	70%	70%

In this approach, the GP search was terminated when one of the following events occurred:

- The number of generations exceeds *max-generations*.
- The training problem was considered solved.

4.3 Terminals and Functions

Terminals. Two forms of terminals were used: numeric and feature terminals. Each numeric terminal returned a single constant value, set initially by randomly sampling a standard normal distribution.

Each feature terminal returned the value of a particular input feature. It is the feature terminals that form the inputs of the GP system from the environment. The features used are simple pixel-statistics derived from the object image. Two statistics (mean and standard deviation) for each of two regions (whole square cutout and centre square of half side-length) were used, making four features altogether.

Functions. Five functions were used, including addition, subtraction, multiplication, protected division and a continuous conditional (soft if) function. Each of the first four functions took two arguments, and applied simple mathematical operations. The soft if function returned the value r in equation 8. This allowed either the second or third argument to control the output, depending on the value of the first argument.

$$r = \frac{a_2}{1 + e^{2a_1}} + \frac{a_3}{1 + e^{-2a_1}} \tag{8}$$

where r is the output of the soft if, and a_i is the value of the i'th argument.

5 Results and Discussion

This section presents a series of results of the new method on the four object classification data sets. These results are compared with those for the PCM and PM methods.

Throughout evolution, the accuracy of the system on the validation set was monitored. The accuracy shown in results (except those for the Face data set) is the accuracy of the system on the test set, at the point of best validation set accuracy. This method is employed to avoid overfitting. For the Face data set, ten-fold cross-validation (TFCV) was used, and the accuracy reported is the maximum test set accuracy, using TFCV, found during evolution. The "run time" reported is the total time required for evolution.

For all parameter settings, the GP process was run 50 times with different random seeds. The mean results were presented, with standard deviation included for accuracy results.

5.1 Overall Results

Table 2 shows a comparison of the best classification results obtained by both the new method (CBD) and the other approaches (PCM and PM) using the same sets of features, functions and parameters. The *solveAt* parameter to the new method was set to 0.01.

On the shape and the first coin data set, all the three GP methods achieved good results, reflecting the fact that the classification problems are relatively

easy in these two data sets. However, the new method achieved perfect test set accuracy for all 50 runs of evolution for the shape data set where the other two existing GP methods did not. On both classification tasks, the new CBD method achieved the best classification accuracy. On the very hard five-class coin problem, the new method gained almost ideal performance (99.19% accuracy), while the PM method achieved 95.92% and PCM only 73.26%. For the face data set, new method achieved comparable results with the PM method, but greatly outperformed the PCM method.

The number of generations required to gain maximum validation set accuracy was very small with the new method. For the shape and three-class coin data sets, this number was well below one for the new method, indicating that the problems could be solved in the initial generation at most of the time due to the good fitness measure in the new CBD method.

The new method normally took a longer time to solve the training problem and terminate the run than the other methods. However, it often found a peak in the validation set accuracy in a shorter time than the PM method. It is expected that the new method would outperform the other methods if only a very limited time or number of generations was allowed for the run.

Table 2. Comparison between PCM, PM and CBD

Dataset	Classes	Method	Gens. at best Val. Acc. (s)	Time to Best Val. Acc. (s)	Run Time (s)	Test Acc. at Best Val. Acc. (%)
Shapes	3	PCM	3.02	0.41	0.45	99.82 ± 0.43
		PM	0.92	0.52	0.53	99.97 ± 0.14
		CBD	0.04	0.53	22.70	100.00 ± 0.00
Coins	3	PCM	8.74	0.94	1.29	99.44 ± 0.88
		PM	1.18	0.50	0.53	99.91 ± 0.27
		CBD	0.06	0.44	25.00	99.97 ± 0.12
	5	PCM	31.32	3.60	4.72	73.26 ± 7.45
		PM	28.82	9.85	14.23	95.92 ± 2.40
		CBD	5.06	2.82	21.69	99.19 ± 0.82
Faces	4	PCM	5.91	0.22	1.75	81.65 ± 13.49
		PM	5.63	0.44	2.78	97.75 ± 7.15
		CBD	2.12	0.36	5.93	96.45 ± 8.73

5.2 Different Problem Solving Criteria in the New Approach

Table 3 shows a comparison of the results on the four data sets using different values for the *solveAt* parameter in the new method. The *solveAt* parameter indicates the separation measure value at which to consider a binary problem solved. Five values were examined for *solveAt* over the four data sets.

Values smaller than 0.01 showed no considerable improvement in accuracy, but increased run time. Values larger than 0.01 degraded performance slightly, but did decrease the time per run. From these experiments, a value of 0.01 for the *solveAt* parameter seems a good starting point.

Table 3. Comparison between values of the *solveAt* threshold, in CBD

Dataset	Classes	*solveAt*	Gens. at best Val. Acc. (s)	Time to Best Val. Acc. (s)	Run Time (s)	Test Acc. at Best Val. Acc. (%)
Shapes	3	0.003	0.04	0.53	28.24	100.00 ± 0.00
		0.010	0.04	0.53	22.70	100.00 ± 0.00
		0.030	0.02	0.52	4.46	100.00 ± 0.00
		0.100	0.00	0.51	0.51	99.39 ± 4.26
		0.300	0.00	0.50	0.51	99.39 ± 4.26
Coins	3	0.003	0.06	0.44	24.82	99.97 ± 0.12
		0.010	0.06	0.44	25.00	99.97 ± 0.12
		0.030	0.06	0.44	23.90	99.97 ± 0.12
		0.100	0.04	0.43	4.23	99.97 ± 0.12
		0.300	0.00	0.41	0.41	99.96 ± 0.14
	5	0.003	5.72	3.16	21.99	99.16 ± 0.87
		0.010	5.06	2.82	21.69	99.19 ± 0.82
		0.030	5.22	2.88	21.32	99.17 ± 0.98
		0.100	3.78	2.08	21.69	99.29 ± 0.91
		0.300	0.70	0.65	1.71	99.04 ± 1.02
Faces	4	0.003	2.12	0.36	5.93	96.45 ± 8.73
		0.010	2.12	0.36	5.93	96.45 ± 8.73
		0.030	2.07	0.35	5.91	96.40 ± 8.78
		0.100	1.96	0.33	5.93	96.15 ± 9.02
		0.300	1.55	0.28	4.86	95.30 ± 9.77

6 Conclusions

The goal of this paper was to construct a method to decompose a multiclass classification problem into multiple two-class subproblems, solve all these subproblems in a single GP run, then combine the subproblems to solve the multiclass problem. This goal was achieved in the CBD method, which allows many component binary classification problems to be solved in one run of the GP process.

A secondary goal was to evaluate the new method on a variety of problems of varying difficulty, comparing it with two other methods. This goal was achieved by a series of experiments comparing the new method with a basic GP approach (PCM) and a previous probabilistic GP method (PM). The new method was seen to outperform both methods when applied to most data sets.

Like many existing GP approaches for multiclass classification problems, the new method can solve a problem of any number of classes in a single GP run. However, unlike most of these approaches, each evolved program produced in the new method is not required to solve the entire multiclass problem. Instead, each program only needs to solve a single component binary subproblem to gain good fitness. The fitness function gives each program a fitness related to its best performance for a binary subproblem.

A mathematically derived method was found to determine the most probable class of a test example, based on the results of a number of expert programs for solving those binary classification subproblems.

The new method requires a parameter, which specifies when a component problem is considered solved. While it does not appear to have a reliable way to obtain a good value for this parameter for different problems which usually needs empirical search, our experiments suggest that 0.01 is a good starting point.

Although developed for multiclass object classification problems, this approach is expected to be able to be applied to general classification problems.

In future work, we will investigate the new method on other general classification problems and compare the performance with other long established methods such as decision trees and neural networks.

References

1. John R. Koza. *Genetic programming : on the programming of computers by means of natural selection*. Cambridge, Mass. : MIT Press, London, England, 1992.
2. John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, Mass. : MIT Press, London, England, 1994.
3. Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming: An Introduction on the Automatic Evolution of computer programs and its Applications*. Morgan Kaufmann Publishers, 1998.
4. Thomas Loveard and Victor Ciesielski. Representing classification problems in genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1070–1077. 2001. IEEE Press.
5. Andy Song, Vic Ciesielski, and Hugh Williams. Texture classifiers generated by genetic programming. In David B. Fogel, et al. editors, *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 243–248. IEEE Press, 2002.
6. Walter Alden Tackett. Genetic programming for feature discovery and image discrimination. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 303–309. 1993. Morgan Kaufmann.
7. Mengjie Zhang and Victor Ciesielski. Genetic programming for multiple class object detection. In Norman Foo, editor, *Proceedings of the 12th Australian Joint Conference on Artificial Intelligence*, pages 180–192, Sydney, Australia, December 1999. Springer-Verlag. LNAI 1747.
8. Mengjie Zhang and Will Smart. Multiclass object classification using genetic programming. In Guenther R. Raidl, et al. editors, *Applications of Evolutionary Computing*, Volume 3005, *LNCS*, pages 367–376, 2004. Springer Verlag.
9. Daniel Howard, S. C. Roberts, and R. Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.
10. Will Smart and Mengjie Zhang. Probability based genetic programming for multiclass object classification. In *Proceedings PRICAI 2004, LNAI Vol. 3157*, pages 251–261, Springer-Verlag, 2004.
11. Mengjie Zhang, Victor Ciesielski, and Peter Andreae. A domain independent window-approach to multiclass object detection using genetic programming. *EURASIP Journal on Signal Processing, Special Issue on Genetic and Evolutionary Computation for Signal Processing and Image Analysis*, 2003(8):841–859, 2003.

12. F. Samaria and A. Harter. Parameterisation of a stochastic model for human face identification. In *2nd IEEE Workshop on Applications of Computer Vision*, Sarasota (Florida), July 1994. ORL database is available at: www.cam-orl.co.uk/facedatabase.html.
13. Will Smart and Mengjie Zhang. Classification strategies for image classification in genetic programming. In Donald Bailey, editor, *Proceeding of Image and Vision Computing Conference*, pages 402–407, New Zealand, 2003.