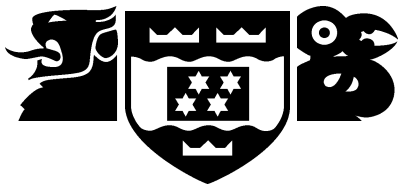
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

Using Gaussian Distribution to Construct Fitness Functions in Genetic Programming for Multiclass Object Classification

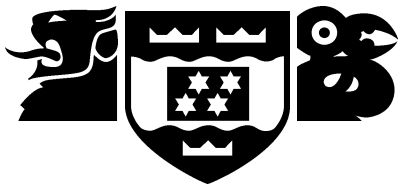
Mengjie Zhang, Will Smart

Technical Report CS-TR-05/5
December 2005

School of Mathematical and Computing Sciences
Victoria University
PO Box 600, Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341, Fax: +64 4 463 5045
Email: Tech.Reports@mcs.vuw.ac.nz
http://www.mcs.vuw.ac.nz/research

Using Gaussian Distribution to Construct Fitness Functions in Genetic Programming for Multiclass Object Classification

Mengjie Zhang, Will Smart

Technical Report CS-TR-05/5
December 2005

Abstract

This paper describes a new approach to the use of Gaussian distribution in genetic programming (GP) for multiclass object classification problems. Instead of using predefined multiple thresholds to form different regions in the program output space for different classes, this approach uses probabilities of different classes, derived from Gaussian distributions, to construct the fitness function for classification. Two fitness measures, overlap area and weighted distribution distance, have been developed. Rather than using the best evolved program in a population, this approach uses multiple programs and a voting strategy to perform classification. The approach is examined on three multiclass object classification problems of increasing difficulty and compared with a basic GP approach. The results suggest that the new approach is more effective and more efficient than the basic GP approach. Although developed for object classification, this approach is expected to be able to be applied to other classification problems.

Keywords Probability based genetic programming, object recognition, object detection, fitness function, multiclass classification

Author Information

The first author is a staff member in computer science, Victoria University of Wellington, New Zealand. The second author is postgraduate student in computer science, Victoria University of Wellington, New Zealand.

Using Gaussian Distribution to Construct Fitness Functions in Genetic Programming for Multiclass Object Classification

Mengjie Zhang, Will Smart

School of Mathematics, Statistics and Computer Science,

Victoria University of Wellington, P.O. Box 600, Wellington, New Zealand

Email: mengjie@mcs.vuw.ac.nz

Abstract

This paper describes a new approach to the use of Gaussian distribution in genetic programming (GP) for multiclass object classification problems. Instead of using predefined multiple thresholds to form different regions in the program output space for different classes, this approach uses probabilities of different classes, derived from Gaussian distributions, to construct the fitness function for classification. Two fitness measures, overlap area and weighted distribution distance, have been developed. Rather than using the best evolved program in a population, this approach uses multiple programs and a voting strategy to perform classification. The approach is examined on three multiclass object classification problems of increasing difficulty and compared with a basic GP approach. The results suggest that the new approach is more effective and more efficient than the basic GP approach. Although developed for object classification, this approach is expected to be able to be applied to other classification problems.

Keywords Probability based genetic programming, object recognition, object detection, fitness function, multiclass classification

1 Introduction

Classification tasks arise in a very wide range of applications, such as detecting faces from video images, recognising words in streams of speech, diagnosing medical conditions from the output of medical tests, and detecting fraudulent credit card transactions [1, 2, 3]. In many cases, people (possibly highly trained experts) are able to perform the classification task well, but there is either a shortage of such experts, or the cost of people is too high. Given the amount of data that needs to be classified, automated classification systems are highly desirable. However, creating automated classification systems that have sufficient accuracy and reliability turns out to be very difficult.

Genetic programming (GP) is a relatively recent and fast developing approach to automatic programming [4, 5, 6]. In GP, solutions to a problem can be represented in different forms but are usually interpreted as computer programs. Darwinian principles of natural selection and recombination are used to evolve a population of programs towards an effective solution to specific problems. The flexibility and expressiveness of computer program representation, combined with the powerful capabilities of evolutionary search, make GP an exciting new method to solve a great variety of problems.

GP research has considered a variety of kinds of classifier programs, using different program representations, including decision tree classifiers, classification rule sets [6], and linear and graph classifiers [4]. Recently, a new form of classifier representation — numeric expression (tree-like) classifiers — has been developed using GP [7, 8, 9, 10]. This form has been successfully applied to real world classification problems such as detecting and recognising particular classes of objects in images [8, 9, 11, 12], demonstrating the potential of GP as a general method to solve classification problems.

The output of a numeric expression GP classifier is a numeric value that is typically translated into a class label. For the simple binary classification case, this translation can be based on the sign of the numeric value [8, 7, 9, 13, 14, 15, 16]; for multiclass problems, finding the appropriate boundary values to separate the different classes is more difficult.

The emphasis of previous approaches was often on separating the program output space into regions (referred to as the *basic GP approach*), with each region indicating a different class. This includes a primary static method such as object classification map or static range selection [10, 7, 12], dynamic range selection [7], centred and slotted dynamic class boundary determination methods [15, 16]. Past work has demonstrated the effectiveness of these approaches, particularly the dynamic methods, on a number of object classification problems.

In the static methods, the region boundaries of program output space were fixed and predefined. In the dynamic methods, class boundaries were automatically found during the evolutionary process. While the static methods often need a hand crafting of good boundaries, the dynamic methods usually involve a long time search to automatically find good boundaries. Both approaches usually take very long training times and often result in unnecessarily complex programs, and sometimes poor performances[12, 16].

1.1 Goals

To avoid these problems and to improve the classification accuracy and the efficiency of the GP system, this paper aims to investigate a new approach to the use of Gaussian distribution and probability for constructing the class translation rule and the fitness function in genetic programming for multiclass classification problems. Rather than setting up class region boundaries for classification, this approach uses Gaussian distribution to model the behaviour of each program based on the training examples for each class. Instead of using the single best evolved program in a population, this approach uses multiple evolved programs to perform classification. This approach is examined on three multiclass object classification tasks of increasing difficulty and compared with the basic GP approach [12] on the same tasks. Specifically, we are interested in assessing the following questions:

- How can the fitness function be constructed using the Gaussian distribution?
- How can the classification accuracy be calculated based on the Gaussian distribution

and multiple evolved programs?

- Can this approach do a good enough job on the given problems?
- Will the new approach outperform the basic GP approach for the same problems?

Note that this paper is focused on numeric expression (tree-like) genetic programming. While other representations of GP such as Linear GP and Graph GP even other methods such as Naive Bayes, decision tress and neural networks can also be used for classification problems, such an investigation is beyond the scope of this paper. However, to give an overview of these methods for related tasks, the next subsection briefly reviews the object recognition and classification tasks, with a little summary of different techniques that have been classically employed.

1.2 Related Work

Object Recognition, or called automatic object recognition or automatic target recognition, is a specific field and a challenging problem in computer vision and image understanding [17]. This task often involves *object localisation* and *object classification*. Object localisation refers to the task of identifying the positions of the objects of interest in a sequence of images either within the visual or infrared spectral bands. Object classification refers to the task of discriminating between images of different kinds of objects, where each image contains only one of the objects of interest.

Traditionally, most research on object recognition involves four stages: *preprocessing, segmentation, feature extraction* and *classification* [18, 19]. The preprocessing stage aims to remove noise or enhance edges. In the segmentation stage, a number of coherent regions and "suspicious" regions which might contain objects are usually located and separated from the entire images. The feature extraction stage extracts domain specific features from the segmented regions. Finally, the classification stage uses these features to distinguish the classes of the objects of interest. The features extracted from the images and objects are

generally domain specific such as high level relational image features. Data mining and machine learning algorithms are usually applied to object classification.

Object recognition has been of tremendous importance in many application domains. These domains include military applications[20, 21, 9], human face recognition[3, 22], agricultural product classification[23], handwritten character recognition[24, 25], medical image analysis[26], postal code recognition [27, 28], and texture classification [29].

Since the 1990s, many methods have been employed for object recognition. These include different kind of neural networks [30, 28, 31, 32], genetic algorithms [33, 34], decision trees [35], statistical methods such as Gaussian models and Naive Bayes [36, 35], support vector machines [36, 35], genetic programming [13, 37, 22, 38], and hybrid methods [39, 40, 41].

Notice that Gaussian member functions have also been used in fuzzy logic and uncertainty. In this paper, we employ the Gaussian model to genetic programming for designing and constructing the fitness function of a genetic program for multiclass object classification.

1.3 Structure

The rest of the paper is organised as follows. Section 2 describes the basic GP approach to classification. Section 3 describes the new fitness function for classification. Section 4 describes how to calculate the classification accuracy using multiple genetic programs. Section 5 describes the object classification problems to be applied. Section 6 presents the results and section 7 gives the conclusions.

2 GP Applied to Multiclass Classification — the Basic GP Approach

In the basic approach, we used the numeric expression (tree like) genetic programs [5]. The ramped half-and-half method was used for generating programs in the initial population and for the mutation operator [4]. The proportional selection mechanism and the reproduction,

crossover and mutation operators [6] were used in the evolutionary learning process.

2.1 Terminals and Functions

2.1.1 Terminals.

Terminals correspond to image features and form the inputs from the environment. To avoid the problem of hand-crafting of feature extraction programs for obtaining high level, domain specific image features for object classification, the basic GP approach used pixel level, domain independent statistical features as terminals and we expect the GP evolutionary process can automatically select features that are relevant to a particular domain to construct good genetic programs.

The terminals considered in this approach are the means and variances of certain regions in object cutout images. Two such regions were used, the entire object cutout image and the central square region, as shown in figure 1. This makes four feature terminals. The values of the feature terminals would remain unchanged in the evolutionary process, but different objects usually have different feature values.

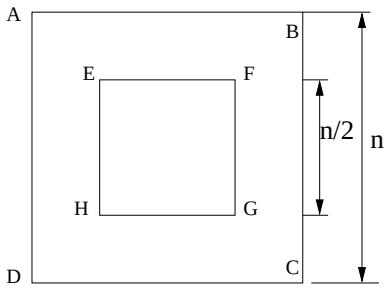


Figure 1: Region features as terminals.

Notice that these features might not be sufficient for some difficult object classification problems. However, they have been found reasonable in many problems and the selection of good features is not the goal of this paper.

In addition, we also used some constants as terminals. These constants are randomly generated using a uniform distribution at the beginning of evolution. Unlike feature terminals,

the values of this kind of terminals are the same for all object images.

2.1.2 Functions.

In the function set, the four standard arithmetic and a conditional operation were used to form the function set:

$$FuncSet = \{+, -, *, /, if\} \tag{1}$$

The $+$, $-$, $/$ and $*$ operators are addition, subtraction, multiplication and "protected" division with two arguments. The 'protected" division is the usual division operator except that a divide by zero gives a result of zero. The *if* function takes three arguments. If the first argument is negative, the *if* function returns its second argument; otherwise, it returns its third argument. The *if* function allows a program to contain a different expression in different regions of the feature space, and allows discontinuous programs, rather than insisting on smooth functions. All functions can take the result of any other functions or terminals as arguments.

2.2 Fitness Function

In the basic GP approach, we used classification accuracy on the training set as the fitness function. To calculate the accuracy, we used a variant version of the *program classification map* [12] as the class translation rule to perform object classification. This rule situates class regions sequentially on the floating point number line. The object image will be classified to the class of the region that the program output with the object image input falls into. Class region boundaries start at some negative number, and end at the same positive number. Boundaries between the starting point and the end point are allocated with an identical interval of 1.0. For example, a five class problem would have the program classification map shown in equation 2 and figure 2.

$$\mathbf{class} = \begin{cases} \text{Class 1}, & r < -1.5 \\ \text{Class 2}, & -1.5 \leq r < -0.5 \\ \text{Class 3}, & -0.5 \leq r < 0.5 \\ \text{Class 4}, & 0.5 \leq r < 1.5 \\ \text{Class 5}, & 1.5 \leq r \end{cases} \tag{2}$$

where r is the program output.

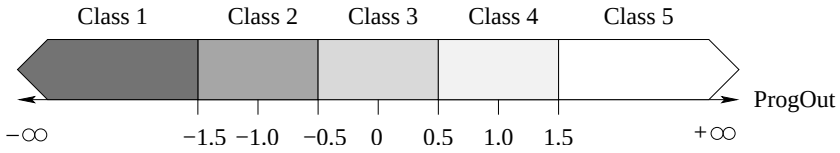


Figure 2: An example program classification map.

Although this class translation rule is easy to set and has achieved some success in several problems [42, 12], it has a number of disadvantages. Firstly, the ordering of classes is fixed. For binary classification problems, we only need one boundary value (usually zero) to separate the program output space into two regions, which is quite reasonable. For multiple class problems, fixing the ordering of different classes is clearly not good in many cases, since it is very difficult to set a good ordering of different classes without sufficient prior domain knowledge. Secondly, the subdivision of the real axis (program output space) is also fixed before evolution. In particular, this method separate the real axis into segments of the same size for some classes (except the "first" and the "last" classes), which often results in a long evolutionary learning time and unnecessarily complex genetic programs. While we can use different boundary values for different classes, it is also very hard to determine the appropriate sizes of different regions for hard classification problems without good expertise of a particular problem.

2.3 Parameters and Termination Criteria

The parameter values used in the basic GP approach are shown in table 1. The evolutionary process is run for a fixed number (*max-generations*) of generations, unless it finds a program that solves the classification perfectly or the performance on the validation set starts falling down, at which point the evolution is terminated early.

Table 1: Parameters used for GP training for the three datasets.

Parameter Names	Shapes	coins	faces	Parameter Names	Shapes	coins	faces
population-size	300	500	500	reproduction-rate	10%	10%	10%
initial-max-depth	3	3	3	cross-rate	60%	60%	60%
max-depth	5	8	8	mutation-rate	30%	30%	30%
max-generations	51	51	51	cross-term	15%	15%	15%
object-size	16×16	70×70	92×112	cross-func	85%	85%	85%

The new approach introduced in this paper used the same program representation, program generation method, genetic operators, terminal set, function set, and the same set of parameters as the basic GP approach. However, in the new approach, we employed the Gaussian distribution and the probability theory to construct a new fitness function and to calculate the final classification accuracy in order to avoid the problems of the old fitness function in the basic GP approach. The details are presented in section 3 and section 4.

3 Constructing Fitness Functions Using Gaussian Distribution Models

In our new approach, we used Gaussian distribution and probability models to construct the fitness function of each program and used multiple programs to calculate the accuracy of a genetic program classifier.

For a set of training data, we assume that the behaviour of a program classifier is modelled using multiple Gaussian distributions, each of which corresponds to a particular class. The

distribution of a class is determined by evaluating the program on the examples of the class in the training set. This was done by taking the mean and standard deviation of the program outputs for those training examples for that class.

For presentation convenience, we first use a two-class problem to describe the fitness measures, then describe the fitness function for multiclass classification problems.

3.1 Fitness Measures

Figure 3 shows three example sets of normal curves for a two class problem. If a program can successfully classify all the training examples (the ideal case), the two curves will be fairly separated (figure 3c); if the two curves are clearly overlapped, some examples will be incorrectly classified by the program (figures 3a and 3b). Clearly, the smaller the overlap, the better the program classifier. Two measures of overlap have been used: *area* and *distance*.

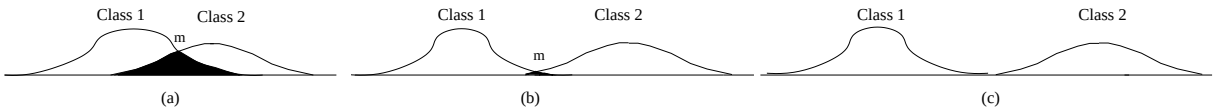


Figure 3: Example normal distributions for a two-class problem.

3.1.1 Area Measure.

In figure 3, the overlap area is shown in solid black. Assuming the intersection point of the two normal curves is m (m was found by a binary search), the area under the left distribution in the region $[m, \infty)$ plus the area under the right distribution in the region $(-\infty, m)$ form the overlap area.

At the beginning of evolution, the standard normal distribution $P(x)$ [43] (see Eq. 3) is sampled from its centre (0) to some large number (i.e. 20) at regular intervals α (i.e. 0.04). The area under the distribution at each point x is approximated using Eq. 4 and the value is stored for later use to simplify the computation.

$$P(x) = \frac{\exp(\frac{-x^2}{2})}{\sqrt{2\pi}} \tag{3}$$

$$A(x) = \sum_{i=0}^{\frac{x}{\alpha}} \alpha P(\alpha i) \tag{4}$$

When the area under a normal distribution with mean μ and standard deviation σ is to be calculated (during evolution), Eq. 5 is used.

$$A(\mu, \sigma, x) = A(\frac{x - \mu}{\sigma}) \tag{5}$$

Accordingly, the approximation of the overlap area in figure 3 (a) will be A_o:

$$A_o = 1 - A(\mu_1, \sigma_1, m) - A(\mu_2, \sigma_2, m) \tag{6}$$

The overlap area has a possible maximal approximation of 1.0 (where the distributions have the same mean), and a possible minimum of zero (where the distributions have different means, but both standard deviations are zero).

3.1.2 Distance Measure.

We also used a second measure, "weighted distribution distance" of the two distributions to evaluate the overlap, as shown in Eq. 7.

$$d = 2 \times \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{7}$$

Under this measure, the worse case is 0, when μ_1 and μ_2 are the same. In the ideal case, this distance will be very large (go to ∞).

In order to make the range of the distance measure the same as for the area measure (0 best, 1 worst), we used the following standardised distribution distance measure d_s in this approach.

$$d_s = \frac{1}{1 + d} \tag{8}$$

3.2 Fitness Function

For multiclass classification, there are three or more classes. The fitness function is determined by considering all the overlaps between every two classes. Assuming the number of classes is n, then there will be $C_n^2 = \frac{n \times (n-1) \times ... \times 2 \times 1}{2}$ overlaps of distributions. The fitness of a program is calculated based on Eq. 9.

$$fitness = \sum_{i=1}^{C_n^2} M_i \tag{9}$$

where M_i is a fitness measure of the ith overlap of distributions of two classes, which can be either the area measure (Eq. 6) or the distance measure (Eq. 8).

4 Obtaining Classification Accuracy Using Multiple Evolved Programs

Classification accuracy is calculated by the number of objects correctly classified by a GP system as a percentage of the total number of objects in a data set. To measure which class a given pattern (object example) belongs to, we used *multiple best programs* rather than the single best program in the population [35, 44]. Assuming l best programs in the population are used, the probability $Prob_c$ of a given pattern being of class c can be calculated by Eq. 10.

$$Prob_c = \prod_{i=1}^{l} P(\mu_{i,c}, \sigma_{i,c}, r_i) \tag{10}$$

where P is the normal probability density function [43] (Eq. 11), r_i is the output result of program i with the pattern to be classified, $\mu_{i,c}$ and $\sigma_{i,c}$ are the mean and standard deviation of the outputs of program i for class c.

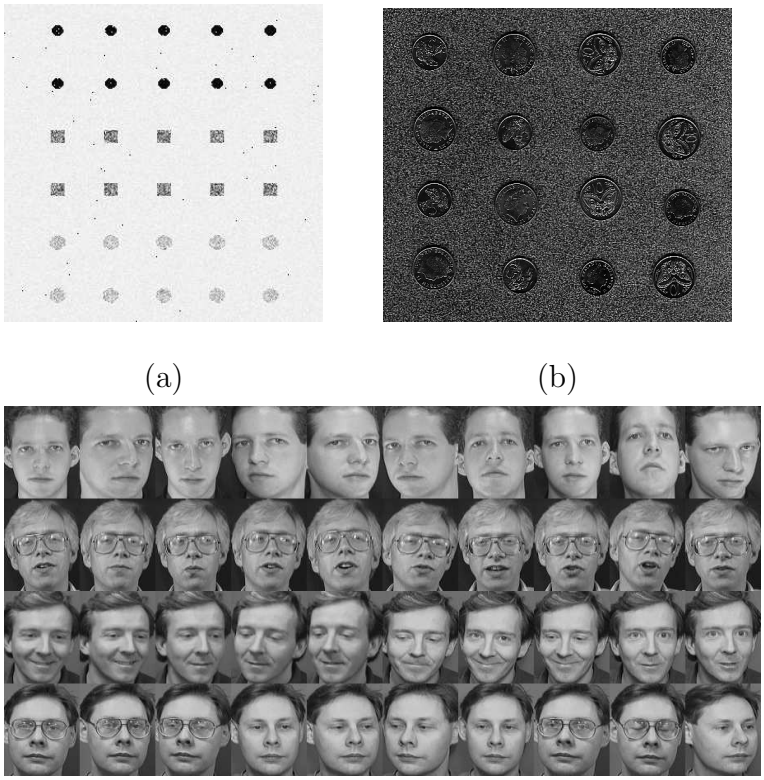
$$P(\mu, \sigma, x) = \frac{\exp(\frac{-(x-\mu)^2}{2\sigma^2})}{\sigma\sqrt{2\pi}} \tag{11}$$

Based on Eq. 10, the probability of the pattern being of each class can be calculated. The class with the largest probability is used as the class of the pattern.

It is important to note that this classification criterion based on the product of the values of the probability density function for a given class for the different classifiers (programs) is only correct when the output of the different classifiers/programs are independent random variables. In many cases, however, this is not accurate. We will consider improvement of this method using a chain rule for conditional probabilities in estimating the probability densities in the future.

5 Data Sets

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



(a) (b)

(c)

Figure 4: Dataset examples: (a) Shapes, (b) Coins, and (c)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 classes of 960 small objects were cut out from those images to form the classification data set. The three classes are: black circles, grey squares, and light circles. For presentation convenience, this dataset is referred to as *shapes*.

The second set of images (figure 4b) contains scanned 5 cent and 10 cent 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. Five classes of 576 object cutouts were created: 5 cent heads, 5 cent tails, 10 cent heads, 10 cent tails, and the cluttered background. Compared with the *shapes* data set, the classification problem in this data set is much harder. Although these are still regular, man-made objects, the problem is very hard due to the cluttered background and a low resolution.

The third data set consists of 40 human faces (figure 4c) 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 [45]. 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 shapes and the coins 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, ten-fold cross validation was applied.

6 Results and Discussion

This section presents a series of results of the new method on the three object classification data sets. For all the data sets, we run 50 times with random seeds. For the shape and the coin data sets, we used 10 different training/test/validation partitions for each run (with 10 experiments) and repeat 50 times. For the face data set, we ten-fold cross validation for each run (with 10 experiments) and repeat 50 times. Only training and test sets are used for the face data set. Accordingly, for each data set, we have 500 experiments and and the average results on the test set are presented. We firstly present the overall results of the new approach compared with the basic GP approach, then describe and analyse the corresponding results against different numbers of evolved programs and different overlap measures used to perform classification.

6.1 Overall Results

Table 2 shows a comparison of the average best classification results (over 500 experiments) obtained by both the new method and the basic GP approach using the same sets of features, functions and parameters. In the standard GP system, the (single) best evolved program was applied to the test set to achieve the final performance of the system. To make a fair comparison with the new approach developed in this paper, the basic approach also uses multiple "best" programs and a simple vote from these "best" programs is then applied to test set to calculate the final performance.

As can be seen from table 2, for the shape data set, both approaches achieved very good results, reflecting the fact that the classification problem in this data set is relatively easy. However, due to the use of a less powerful class translation rule, it took the basic GP approach 11.70 generations and 3.29 seconds on average to find "good" program classifiers to achieve an average of 98.64% accuracy on the test set. On the other hand, the new approach used less than one generation and only 0.29 second on average to achieve almost ideal performance. This reflect the fact that the classification problem in this data set is almost trivial for

Table 2: Best results of the new and the basic GP approaches.

Dataset	Strategy	Generations	Time (s)	Test Accuracy (%)
Shapes	Basic approach	11.70	3.29	98.64
	New approach	0.86	0.29	99.96
Coins	Basic approach	41.02	6.07	90.46
	New approach	14.26	2.27	98.60
Faces	Basic approach	8.60	0.38	86.75
	New approach	4.66	0.24	97.75

the new method due to the terminals and fitness function used. Clearly, the new method outperformed the basic GP approach on this data set in terms of the convergence (number of generations), evolutionary learning time and the classification accuracy on unseen test data.

The results on the coin and the face data sets show a similar pattern to those for the shape data set. For the coin set, the new method achieved an almost perfect average accuracy (98.60%), which is much better than that achieved by the basic GP approach (90.46%). The convergence and training processing in the new method are also much faster. For the face data set, the accuracy improvement of the new method over the basic GP approach is more than 10% ($(97.75 - 86.75)/86.75 = 12.7\%$) and evolutionary learning process for finding good classifiers is also much more efficient.

In summary, on all the three data sets, the new approach achieved very good results and always outperformed the basic approach, in terms of both classification accuracy and training time. This trend is particularly clear for relatively difficult problems such as in the coins and faces data sets. This indicates that the new approach is more effective than the basic GP approach and is more efficient to learn good classifiers for these problems.

6.2 Different Programs and Fitness Measures

We used two fitness measures in the new approach to evaluate the programs and used multiple programs to obtain the classification results. This section is to compare the two fitness measures and to investigate how many programs should be used.

Table 3 shows a comparison of the results on the three data sets using different numbers of programs and the two fitness measures in the new approach.

Table 3: Results for different programs and the two fitness measures.

Dataset	Programs Used	Generations		Time (s)		Test Accuracy (%)	
		Fitness Measure		Fitness Measure		Fitness Measure	
		Distance	Area	Distance	Area	Distance	Area
Shapes	1	0.88	0.86	0.26	0.29	97.48	99.96
	3	0.38	0.34	0.19	0.20	99.84	99.83
	5	0.14	0.12	0.15	0.16	99.81	99.80
	10	0.04	0.04	0.14	0.15	99.79	99.78
	20	0.06	0.06	0.16	0.18	99.68	99.68
	50	0.30	0.16	0.30	0.27	99.46	99.46
Coins	1	32.98	33.78	4.55	5.37	94.23	97.49
	3	18.70	19.42	2.47	2.93	97.81	98.41
	5	14.96	15.76	1.96	2.39	98.12	98.38
	10	14.62	14.26	2.06	2.27	98.46	98.60
	20	10.82	9.06	1.60	1.57	98.40	98.25
	50	14.54	13.32	2.84	2.90	98.06	98.29
Faces	1	5.13	4.66	0.20	0.24	96.35	97.75
	3	2.09	1.73	0.08	0.10	96.95	95.90
	5	1.56	1.49	0.07	0.09	96.10	96.10
	10	1.03	0.88	0.05	0.06	95.45	94.70
	20	0.64	0.68	0.04	0.06	93.25	93.25
	50	0.25	0.18	0.03	0.04	91.20	90.30

As can be seen from table 3, different numbers of programs for classification resulted in different performances. In most cases, it seems that using multiple programs led to a better accuracy and a shorter training time than using the single best evolved genetic program. At certain numbers, the method achieved a high accuracy and spent a short training time, but the numbers leading to good results for various data sets appeared to be different. It generally needs an empirical search to obtain such good numbers for different data. However, if this can improve the performance, such a search is a small price to pay.

In terms of two fitness measures, the results show that both measures achieved very good results, which were always better than the basic approach for the same problems investigated here. The area measure achieved slightly better accuracy than the distance measure in some cases, but slightly worse results in other cases. While the area measure generally required slightly fewer evaluations (generations) than the distance measure, it often took a slightly longer time to learn the good program classifiers. This is mainly because the computational complexity of the area measure is greater than the distance measure in the new algorithm.

Another interesting observation from the results is that when we only used the (single) best evolved program classifier, the area measure always achieved better classification accuracy on all the three problems investigated here. This suggests that if we only pick up the single best learned program as most GP systems do, it seems that the area measure should be employed. If we use more than five "best" programs, either measure can be chosen and applied. Further investigation on other problems needs to be carried out in the future.

7 Conclusions

The main goal of this paper was to construct the fitness function using Gaussian distributions in genetic programming for multiclass object classification problems. This goal was achieved by introducing two measures for the overlaps of distributions between every two classes on the training examples. A second goal was to investigate whether this new approach was better than a basic GP approach using the same set of features (terminals) and functions. Both

approaches were examined on three object classification problems of increasing difficulty. The results suggest that the new approach outperformed the basic approach in terms of both classification accuracy and training time.

Two fitness measures, *overlap area* and *distribution distance*, were developed for constructing the fitness function. Both measures achieved much better results than the basic approach on the three classification problems of increasing difficulty. The results also suggest that if only one (the best) learned program as in most GP systems, the area measure should be employed.

Unlike most existing GP approaches for classification problems, where only the best learned/evolved program classifier was applied to the object examples to measure the accuracy, this approach used multiple top programs for classification and the class with the largest probability is used as the class of the object pattern. The results suggest that the multiple-program approach outperformed the single best program approach in most cases.

Compared with the basic GP approach, a major advantage of the new approach is that manually defining different boundaries for different classes over the program output space was successfully avoided. A minor disadvantage is that we need to do a bit empirical search for the best number of programs employed. Nevertheless, the results suggest that the two overlap fitness measures with different number of programs always achieved better performance than the basic GP approach.

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

The fitness function and the classification rule designed in this approach are based on the assumption that the output of the different classifiers/programs are independent random variables. In many cases, this is not accurate. We will consider improvement of this method using a chain rule for conditional probabilities in estimating the probability densities to investigate whether the performance can be further improved. We will also investigate the effectiveness of the new approach on other data sets, and compare this approach with other learning methods such as naive Bayes, decision trees, and neural networks in the future.

References

[1] J. Eggermont, A. E. Eiben, and J. I. van Hemert. A comparison of genetic programming variants for data classification. In *Proceedings of the Third Symposium on Intelligent Data Analysis (IDA-99), LNCS 1642*. Dpringer-Verlag, 1999.

[2] H. F. Gray, R. J. Maxwell, I. Martinez-Perez, C. Arus, and S. Cerdan. Genetic programming for classification of brain tumours from nuclear magnetic resonance biopsy spectra. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, page 424, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[3] D. Valentin, H. Abdi, and O'Toole. Categorization and identification of human face images by neural networks: A review of linear auto-associator and principal component approaches. *Journal of Biological Systems*, 2(3):413–429, 1994.

[4] 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*. San Francisco, Calif. : Morgan Kaufmann Publishers; Heidelburg : Dpunkt-verlag, 1998. Subject: Genetic programming (Computer science); ISBN: 1-55860-510-X.

[5] John R. Koza. *Genetic programming : on the programming of computers by means of natural selection*. Cambridge, Mass. : MIT Press, London, England, 1992.

[6] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, Mass. : MIT Press, London, England, 1994.

[7] Thomas Loveard and Victor Ciesielski. Representing classification problems in genetic programming. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 1070–1077, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.

[8] Andy Song, Vic Ciesielski, and Hugh Williams. Texture classifiers generated by genetic programming. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 243–248. IEEE Press, 2002.

[9] Walter Alden Tackett. Genetic programming for feature discovery and image discrimination. In Stephanie Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms, ICGA-93*, pages 303–309, University of Illinois at Urbana-Champaign, 17-21 July 1993. Morgan Kaufmann.

[10] 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 (AI'99)*, pages 180–192, Sydney, Australia, December 1999. Springer-Verlag Berlin Heidelberg. Lecture Notes in Artificial Intelligence (LNAI Volume 1747).

[11] Mengjie Zhang, Peter Andreae, and Mark Pritchard. Pixel statistics and false alarm area in genetic programming for object detection. In Stefano Cagnoni, editor, *Applications of Evolutionary Computing, Lecture Notes in Computer Science, LNCS Vol. 2611*, pages 455–466. Springer-Verlag, 2003.

[12] 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.

[13] Daniel Howard, Simon C. Roberts, and Richard Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.

[14] Jamie R. Sherrah, Robert E. Bogner, and Abdesselam Bouzerdoum. The evolutionary pre-processor: Automatic feature extraction for supervised classification using genetic

programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 304–312, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[15] 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, Palmerston North, New Zealand, November 2003.

[16] Mengjie Zhang and Will Smart. Multiclass object classification using genetic programming. In Guenther R. Raidl, Stefano Cagnoni, Jurgen Branke, David W. Corne, Rolf Drechsler, Yaochu Jin, Colin Johnson, Penousal Machado, Elena Marchiori, Franz Rothlauf, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, volume 3005 of *LNCS*, pages 367–376, Coimbra, Portugal, 5-7 April 2004. Springer Verlag.

[17] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003.

[18] Terry Caelli and Walter F. Bischof. *Machine Learning and Image Interpretation*. Plenum Press, New York and London, 1997. ISBN 0-306-45761-X.

[19] Earl Gose, Richard Johnsonbaugh, and Steve Jost. *Pattern Recognition and Image Analysis*. Prentice Hall PTR, Upper Saddle River, NJ 07458, 1996. ISBN 0-13-236415-8.

[20] Ayanna Howard, Curtis Padgett, and Carl Christian Liebe. A multi-stage neural network for automatic target detection. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 231–236, Anchorage, Alaska, 1998. 0-7803-4859-1/98.

[21] Yee Chin Wong and Malur K. Sundareshan. Data fusion and tracking of complex target maneuvers with a simplex-trained neural network-based architecture. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1024–1029, Anchorage, Alaska, May 1998. 0-7803-4859-1/98.

[22] Astro Teller and Manuela Veloso. A controlled experiment : Evolution for learning difficult image classification. In Carlos Pinto-Ferreira and Nuno J. Mamede, editors, *Proceedings of the 7th Portuguese Conference on Artificial Intelligence*, volume 990 of *LNAI*, pages 165–176, Berlin, 3–6 October 1995. Springer Verlag.

[23] P. Winter, W. Yang, S. Sokhansanj, and H. Wood. Discrimination of hard-to-pop popcorn kernels by machine vision and neural network. In *ASAE/CSAE meeting*, Saskatoon, Canada, Sept. 1996. Paper No. MANSASK 96-107.

[24] David Andre. Automatically defined features: The simultaneous evolution of 2-dimensional feature detectors and an algorithm for using them. In Kenneth E. Kinnear, editor, *Advances in Genetic Programming*, pages 477–494. MIT Press, 1994.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[26] Brijesh Verma. A neural network based technique to locate and classify microcalcifications in digital mammograms. In *1998 IEEE World Congress on Computational Intelligence – IJCNN'98*, pages 1790–1793, Anchorage, Alaska, 1998. 0-7803-4859-1/98, IEEE.

[27] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hibbard. Handwritten digit recognition: application of neural network chips and automatic learning. *IEEE Communications Magazine*, pages 41–46, November 1989.

[28] Dick de Ridder, Aarnoud Hoekstra, and Robert P. W. Duin. Feature extraction in shared weights neural networks. In *Proceedings of the Second Annual Conference of the Advanced School for Computing and imaging, ASCI*, pages 289–294, Delft, June 1996.

[29] Andy Song, Thomas Loveard, and Victor Ciesielski:. Towards genetic programming for texture classification. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, pages 461–472. Springer Verlag, 2001.

[30] M. R. Azimi-Sadjadi, D. Yao, Q. Huang, and G. J. Dobeck. Underwater target classification using wavelet packets and neural networks. *IEEE Transactions on Neural Networks*, 11(3):784–794, May 2000.

[31] C. Stahl, DaimlerChrysler Aerospace, and P. Schoppmann. Advanced automatic target recognition for police helicopter missions. In F. A. Sadjadi, editor, *Proceedings of SPIE Volume 4050, Automatic Target Recognition X*, April 2000. pp. 61-68.

[32] T. Wessels and C. W. Omlin. A hybrid system for signature verification. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), Volume V*, Como, Italy, July 2000.

[33] J. Bala, K. De Jong, J. Huang, H. Vafaie, and H. Wechsler. Using learning to facilitate the evolution of features for recognising visual concepts. *Evolutionary Computation*, 4(3):297–312, 1997.

[34] Jeng-Sheng Huang and Hsiao-Chung liu. Object recognition using genetic algorithms with a Hopfield's neural model. *Expert Systems with Applications*, 13(3):191–199, 1997.

[35] Stuart Russell and Peter Norvig. *Artificial Intelligence, A modern Approach*. Prentice Hall, 2nd edition, 2003.

[36] Margaret H. Dunham. *Data Mining: Introductory and Advanced Topics*. Prentice Hall, 2003.

[37] Daniel Howard, Simon C. Roberts, and Conor Ryan. The boru data crawler for object detection tasks in machine vision. In Stefano Cagnoni, Jens Gottlieb, Emma Hart, Martin Middendorf, and G"unther Raidl, editors, *Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim*, volume 2279 of *LNCS*, pages 220–230, Kinsale, Ireland, 3-4 April 2002. Springer-Verlag.

[38] Jay F. Winkeler and B. S. Manjunath. Genetic programming for object detection. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 330–335, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[39] Peter G. Korning. Training neural networks by means of genetic algorithms working on very long chromosomes. *International Journal of Neural Systems*, 6(3):299–316, September 1995.

[40] Victor Ciesielski and Jeff Riley. An evolutionary approach to training feed forward and recurrent neural networks. In L. C. Jain and R. K. Jain, editors, *Proceedings of the Second International Conference on Knowledge Based Intelligent Electronic Systems*, pages 596–602, Adelaide, April 1998.

[41] X. Yao and Y. Liu. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, May 1997.

[42] Andy Song. *Texture Classification: A Genetic Programming Approach*. PhD thesis, Department of Computer Science, RMIT University, Melbourne, Australia, 2003.

[43] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference*. Prentice Hall, 6th edition, 2000.

[44] Terence Soule. Voting teams: A cooperative approach to non-typical problems using genetic programming. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H.

Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 916–922, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[45] 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.