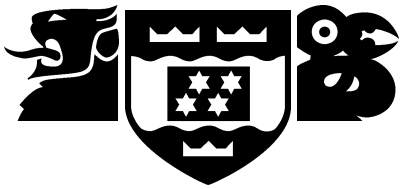
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



School of Mathematical and Computing Sciences
Computer Science

Probability Based Genetic Programming for Multiclass Object Classification

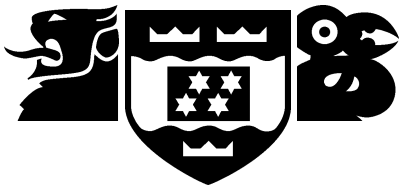
Will Smart, Mengjie Zhang

Technical Report CS-TR-04/7
April 2004

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

Probability Based Genetic Programming for Multiclass Object Classification

Will Smart, Mengjie Zhang

Technical Report CS-TR-04/7
April 2004

Abstract

This paper describes a probability based genetic programming (GP) approach to 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. 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. While the area measure was a bit more effective than the distance measure in most cases, the distance measure was more efficient to learn good program classifiers.

Keywords Probability based genetic programming, Gaussian distribution, overlap area, weighted distribution distance, multiclass object classification.

Author Information

Will Smart is a postgraduate student in computer science and Mengjie Zhang is an academic staff member in computer science. Both authors are in the School of Mathematical and Computing Sciences, Victoria University of Wellington, New Zealand.

Probability Based Genetic Programming for Multiclass Object Classification

Will Smart and Mengjie Zhang

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

Abstract. This paper describes a probability based genetic programming (GP) approach to 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. 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. While the area measure was a bit more effective than the distance measure in most cases, the distance measure was more efficient to learn good program classifiers.

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 fraud transactions [1, 2]. 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 [3, 4]. In GP, solutions to a problem are represented 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, makes 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 and classification rule sets [5]. Recently, a new form of classifier representation – numeric

expression classifiers – has been developed using GP [6–9]. This form has been successfully applied to real world classification problems such as detecting and recognising particular classes of objects in images [7, 8, 10, 11], demonstrating the potential of GP as a general method for 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 [7, 6, 8, 12–15]; 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 [9, 6, 11], dynamic range selection [6], centred and slotted dynamic class boundary determination methods [14, 15]. 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[11, 15].

1.1 Goals

To avoid the above disadvantages, the goal of this paper is to investigate a new approach to the use of Gaussian distribution and probability 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. This approach is examined on three multiclass object classification problems of increasing difficulty and compared with the basic GP approach [11] on the same problems. Specifically, we are interested in:

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

1.2 Structure

This paper is organised as follows. Section 2 describes genetic programming applied to classification. Section 3 describes the new fitness function for classification. Section 4 describes the object classification problems to be applied. Section 5 presents the results and section 6 gives the conclusions.

2 GP Applied to Multiclass Classification

2.1 Terminals and Functions

Terminals. In this approach, we used two kinds of terminals: *feature terminals* and *numeric parameter terminals*.

Feature terminals form the inputs from the environment. The feature 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. 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.

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.

Numeric parameter terminals are floating point numbers randomly generated using a uniform distribution at the beginning of evolution. Unlike feature terminal, the values of this kind of terminals are the same for all object images.

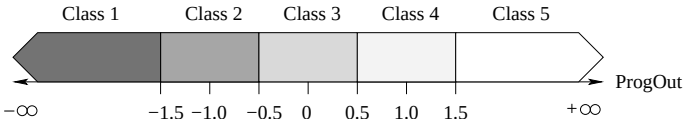
Functions. In the function set, the four standard arithmetic and a conditional operation were used to form the function set:

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

The $+$, $-$, $/$ and $*$ operators are addition, subtraction, multiplication and "protected" division with two arguments. The *if* function takes three arguments. If the first argument is negative, the *if* function returns its second argument; otherwise, it returns its third argument.

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* [11] to perform object classification. This variation 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 following classification map.



In the new approach, we applied the Gaussian distribution and used probability to construct the fitness function and to calculate the final classification accuracy. The details are presented in section 3.

2.3 Parameters and Termination Criteria

The parameter values used in this 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 (100% accuracy), 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%

3 Constructing Fitness Function Using Gaussian 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 and the calculation of classification accuracy.

3.1 Fitness Measures

Figure 1 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 1 c); if the two curves are clearly overlapped, some examples will be incorrectly classified by the program (figure 1 a, b). Clearly, the smaller the overlap, the better the program classifier. Two measures of overlap have been used: *area* and *distance*.

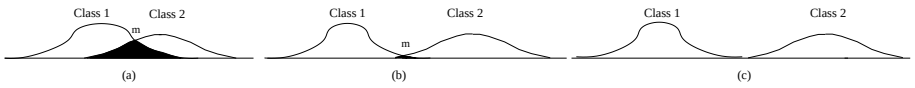


Fig. 1. Example normal distributions for a two-class problem.

Area Measure. In figure 1, 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 rightmost distribution in the region $(-\infty, m)$ forms the overlap area. Note that to calculate the areas directly is an NP-complete problem and an approximation must be made, which should be close enough to use.

At the beginning of evolution, the standard normal distribution $P(x)$ (Eq. 2) is sampled from its centre (0) to some large number β (i.e. 20) at regular intervals α (i.e. 0.04). The area under the distribution at each point x is calculated using Eq. 3 and the value is stored for later use to simplify the computation.

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

$$A(x) = \sum_{i=0}^{\frac{x}{\alpha}} \alpha P(\alpha i) \tag{3}$$

When the area under a normal distribution with mean μ and standard deviation σ is to be calculated (during evolution), Eq. 4 is used.

$$A(\mu, \sigma, x) = A(\frac{x-\mu}{\sigma}) \tag{4}$$

Accordingly, the approximation of the overlap area in figure 1 (a) will be A_o:

$$A_o = 1 - A(\mu_1, \sigma_1, m) - A(\mu_2, \sigma_2, m) \tag{5}$$

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).

Distance Measure. We also used a second measure, "weighted distribution distance" of the two distributions to evaluate the overlap, as shown in Eq. 6.

$$d = 2 \times \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2} \tag{6}$$

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.

$$d_s = \frac{1}{1+d} \tag{7}$$

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 $C_n^2 = \frac{n \times (n-1) \times \ldots \times 2 \times 1}{2}$ overlaps of distributions. The fitness of a program is calculated based on Eq. 7.

$$fitness = \sum_{i=1}^{C_n^2} M_i \tag{8}$$

where M_i is a fitness measure of the ith overlap of distributions of two classes, which can be either the area measure (Eq. 5) or the distance measure (Eq. 7).

3.3 Classification Accuracy

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. 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. 9.

$$Prob_c = \prod_{i=1}^{l} P(\mu_{i,c}, \sigma_{i,c}, r_i) \tag{9}$$

where P is the normal probability function (Eq. 10), 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.

$$P(\mu, \sigma, x) = \frac{\exp(\frac{-(x-\mu)^2}{2\sigma^2})}{\sigma\sqrt{2\pi}} \tag{10}$$

Based on Eq. 9, 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.

4 Data Sets

We used three data sets providing object classification problems of increasing difficulty in the experiments. Example images are shown in figure 2.

The first set of images (figure 2a) 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 2b) 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 and 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.

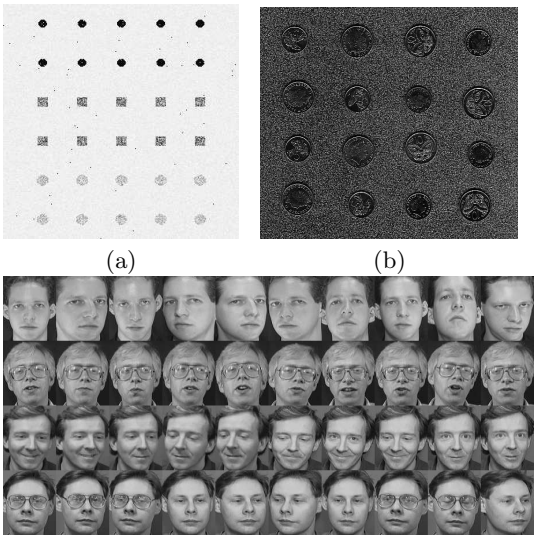


Fig. 2. Dataset examples: (a) Shapes, (b) Coins, and (c)Faces.

The third data set consists of 40 human faces (figure 2c) 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 [16]. 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.

5 Results and Discussion

This section presents a series of results of the new method on the three object classification data sets. These results are compared with those for the basic GP approach. For all experiments, we run 50 times with random seeds and the average results on the test set were presented.

5.1 Overall Results

Table 2 shows a comparison of the best classification results obtained by both the new method and the basic GP approach using the same sets of features, functions and parameters. 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.

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

Dataset	Strategy	Generations	Time (s)	Test Accuracy (%)
Shapes	Basic approach	16.06	4.36	99.65
	New approach	0.86	0.29	99.96
Coins	Basic approach	41.54	6.02	89.23
	New approach	19.42	2.93	98.41
Faces	Basic approach	8.32	0.37	85.00
	New approach	4.66	0.24	97.75

5.2 Different Programs and Fitness Measures in the New Approach

We used two fitness measures 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 numbers of programs used 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. This is particularly true for difficult object

classification problems. It seems that 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 suggest that the area measure resulted in better performance in accuracy in most cases, but it also took a bit 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. Nevertheless, both measures achieved quite good results, which were better than the basic approach for the same problems.

6 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 is better than the basic approach in terms of both classification accuracy and training time.

Two fitness measures, *overlap area* and *distribution distance*, were applied to constructing the fitness function. Both measures achieved much better results than the basic approach on the three classification problems of increasing difficulty. Although the overlap area measure resulted in better performance in most cases, it also took longer time to evolve good program classifiers due to its greater complexity of computation than the distance measure.

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.

While this approach does not need to manually define the fixed class boundaries, it does need to find a good number of programs used for object classification. The results suggest that there does not appear to be a reliable way to find a good number of programs, which is most likely problem dependent and needs an empirical search. However, if this can greatly improve the performance, such a search is a small price to pay.

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

For the future work, we will further investigate the number of programs for object classification, investigate the effectiveness of the new approach on other data sets, and compare this approach with other learning methods such as decision trees and neural networks.

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. Helen Gray. Genetic programming for classification of medical data. In John R. Koza, editor, *Late Breaking Papers at the 1997 Genetic Programming Conference*, pages 291–297. Standford University, 1997.
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. John R. Koza. *Genetic programming : on the programming of computers by means of natural selection*. Cambridge, Mass. : MIT Press, London, England, 1992.
5. John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, Mass. : MIT Press, London, England, 1994.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
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. Daniel Howard, S. C. Roberts, and R. Brankin. Target detection in SAR imagery by genetic programming. *Advances in Engineering Software*, 30:303–311, 1999.
13. 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, et al. editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 304–312, 1997.
14. 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.
15. 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.
16. 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.