Using Genetic Programming to Evolve a Team of Data Classifiers

Gregor A. Morrison, Dominic P. Searson and Mark J. Willis

Abstract**—**The purpose of this paper is to demonstrate the ability of a genetic programming (GP) algorithm to evolve a team of data classification models. The GP algorithm used in this work is "multigene" in nature, i.e. there are multiple tree structures (genes) that are used to represent team members. Each team member assigns a data sample to one of a fixed set of output classes. A majority vote, determined using the mode (highest occurrence) of classes predicted by the individual genes, is used to determine the final class prediction. The algorithm is tested on a binary classification problem. For the case study investigated, compact classification models are obtained with comparable accuracy to alternative approaches.

Keywords**—**classification, genetic programming.

I. Introduction

GENETIC programming [1] is a biologically inspired machine learning method that evolves computer programs to perform a task. It does this by randomly generating a population of computer programs (represented by tree structures) and then mutating and crossing over the best performing trees to create a new population. This process is iterated until the population contains programs that (hopefully) solve the task well.

GP is often used to evolve predicted numerical models but it can also be used to evolve classifiers, i.e. rules that can classify a number of data "objects" (using a set of known attributes) correctly into 2 or more classes. In [2] a review of classification algorithms developed using GP is provided. The authors of [2] conclude that when compared to decision trees or neural networks GP can evolve models and rules that have comparable (or better) performance, see e.g. [3], [4], [5]. Recent literature [6] reports improved accuracy through the fusion of multiple independent classifiers into ensemble classifiers, using (for instance) majority voting strategies. However, as autonomous algorithms are not specifically developed to trade-off contributions made by individual members of the ensemble, fusion can result in complex algorithm structures whose improved performance is not guaranteed [6], [7]. The purpose of this paper is to demonstrate that the multigene GP algorithm [8], [9], [10] has the potential to produce an accurate, relatively compact co-operating ensemble (team) of classifiers. Here, the members of the ensemble are evolved together as a team in order to solve the classification task.

This paper is structured as follows. Section II provides a brief overview of the multigene GP algorithm. Next, in section III, the methodology adopted for the development of classification rules using the multigene algorithm is described. In section IV, the results obtained for the classification of a public domain data set, the Wisconsin breast cancer data, are presented. Finally, in section V, a concluding discussion is given. The following material assumes a basic familiarity with GP. If this is not the case then an excellent, free to download introduction and review of the literature is provided by [11].

II. Multigene GP

Typically, GP evolves a population of individual trees, each of which encodes a single mathematical equation or rule that predicts a $(N \times 1)$ vector $\mathbf{y}$ of real numbers using a corresponding $(N \times M)$ matrix of inputs $\mathbf{X}$ where N is the number of observations of the response variable and M is the number of input (predictor) variables. That is, the ith column of $\mathbf{X}$ comprises the N input values for the ith input variable.

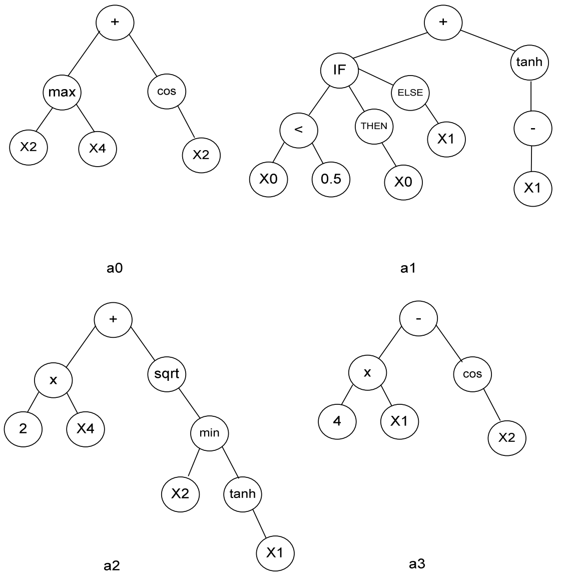

Fig. 1 An example of a multigene "team" of models.

In contrast, in multigene GP a single population member consists of a number of trees. For instance, Fig. 1 shows a multigene individual with four trees. The mathematical models that each tree represents are:

G. A. Morrison is with the School of Chemical Engineering and Advanced Materials at the University of Newcastle, Newcastle-upon-Tyne, UK.

D. P. Searson is with School of Chemical Engineering and Advanced Materials at the University of Newcastle, Newcastle-upon-Tyne, UK (e-mail: d.p.searson@ncl.ac.uk).

M. J. Willis is with the School of Chemical Engineering and Advanced Materials at the University of Newcastle, Newcastle-upon-Tyne, UK. (phone +44 191 222 7242; e-mail mark.willis@ncl.ac.uk).

$$a_0 \colon \max(x_2 x_4) + \cos(x_2)$$

$$a_1 \colon ((if\ x_0 < 0.5) then\ x_0 \\ else x_1) + \tanh(-x_1))$$

$$a_2 \colon (2x_4 + sqrt(\min(x_2, \tanh(x_1))))$$

$$a_3 \colon 4x_1 - \cos(x_2)$$

In practice, the user specifies the maximum number of genes $G_{\max}$ a model is allowed to have and the maximum tree depth $D_{\max}$ any gene may have and therefore can exert control over the maximum complexity of the evolved models.

In particular, we have found that enforcing stringent gene number and tree depth restrictions (e.g. maximum number of 6 genes and tree depth of 5) allows the evolution of relatively compact individuals that may be used for the development of classification rules, even when there are a large number of input variables in the dataset.

The initial population is constructed by creating individuals that contain randomly generated GP trees with between 1 and $G_{\max}$ genes. During a GP run, genes are acquired and deleted using a tree crossover operator called two point high level crossover. This allows the exchange of genes between individuals and it is used in addition to the "standard" GP recombination operators.

If the *i*th gene in an individual is labelled G_i then a two point high level crossover is performed as in the following example. Here, the first parent individual contains the genes (G_1 G_2 G_3) and the second contains the genes (G_4 G_5 G_6 G_7) where $G_{\max} = 5$. Two randomly selected crossover points are created for each individual. The genes enclosed by the crossover points are denoted by < … >.

$(G_1 < \boldsymbol{G_2} > G_3)$ $(G_4 < \boldsymbol{G_5}\ \boldsymbol{G_6}\ \boldsymbol{G_7} >)$

The genes enclosed by the crossover points are then exchanged resulting in the two new individuals below.

$(G_1\ G_5\ G_6\ G_7\ G_3)$ $(G_4\ G_2)$

Two point high level crossover allows the acquisition of new genes for both individuals but also allows genes to be removed. If an exchange of genes results in an individual containing more genes than $G_{\max}$ then genes are randomly selected and deleted until the individual contains $G_{\max}$ genes.

The standard GP subtree crossover is referred to as low level crossover. In this case, a gene is selected randomly from each parent individual, standard subtree crossover is performed and the resulting trees replace the parent trees in the otherwise unaltered individual in the next generation.

III. DATA CLASSIFICATION USING THE MULTIGENE ALGORITHM

When presented with real valued inputs, each gene will output a real number that theoretically lies within the range -∞ to ∞. In order to predict which class an object lies in, this output needs to be processed so that the output of a tree indicates a discrete class number (e.g. class 0, class 1etc.). To achieve this, the outputs are first 'squashed' to lie within a pre-determined range which is defined according to the total number of classes. The function used to do this is shown below.

$$(N_C - 1)\left(\frac{1}{(1 + \exp(-g_o)}\right)$$

Where N_C is the number of output classes and g_o is the output of a particular gene. Therefore, for a four class problem, each gene output will lie in the range 0 – 3. Each numeric value is then rounded to the nearest integer, in this example, 0, 1, 2 or 3 enabling each gene to predict either class '0', '1', '2' or '3'. Similarly for a two class problem, the output of a gene will now be squashed within the range 0 – 1. Subsequent rounding of these numbers to the nearest integer will produce a prediction that an object is in either class '0' or class '1'.

The predictions obtained from each gene are then interpreted to produce an overall prediction of the output class when presented with the input data for an object. To do this, a voting strategy is adopted; accepting the majority vote determined using the mode (highest occurrence) of the classes predicted by the individual genes. For example, given a six gene model predicting binary classes, if the individual gene outputs are: {0,1,0,0,1,0} the mode, and hence the final prediction is class '0'. If there is a draw – which can happen if a particular population member has an even number of genes - the prediction is assigned to the lowest class number.

In order to evolve accurate classification rules, a suitable fitness function is required. There are many situations where data has unequal class representation and commonly it is the smaller class that is of interest. For example, in the chemical and allied industries the interest is normally in those variables that fail a quality test. In this situation fitness functions based upon measures of sensitivity and specificity are required. Consider binary classification where an outcome is either class '0' or class '1'. The sensitivity rule is the proportion of true cases which it correctly predicts as being class '0'. Specificity refers to the portion of class '1' cases which it correctly predicts as being class '1'. The fitness function used is a weighted combination of these two measures. For binary classification problems, this Sensitivity Fitness Function (*SFF*) is given by

$$SFF = w\frac{n_c(i,0)}{n_a(i,0)} + (1-w)\frac{n_c(i,1)}{n_a(i,1)}$$

where $n_c(i,k)$ is the number of cases in class k correctly predicted by the *i*th solution to be in class k; $n_a(i,k)$ is the number of records which are actually in class k and w is a weight in the range [0, 1].

Normally equal weights are used. That is, the average of the sensitivity and specificity values is taken. Thus the fitness function takes into account the proportion of correct classifications in each class, so that the fitness for a rule that misclassifies all the cases in one of the classes is at most 0.5. In some situations, the weighting may need to be adjusted depending upon the cost associated for misclassifying cases from each class or to deal with an unbalanced dataset. The fitness function may be easily generalised to cover multi-way classifications.

In order to promote the evolution of compact models a solution length penalty was also included within the fitness function

$$SFF = w\frac{n_c(i,0)}{n_a(i,0)} + (1-w)\frac{n_c(i,1)}{n_a(i,1)} - pN$$

where p is an additional weighting and N is the number of nodes in a given population member.

IV. THE WISCONSIN BREAST CANCER DATASET

The data set used is the Wisconsin diagnostic breast cancer[1] dataset which consists of 569 occurrences of 30 tumour related measurements from women with either a benign or malignant tumour. The dataset is partitioned, with approximately 60% having a benign tumour and the remainder malignant. The classification objective is to develop a model to predict whether a tumour is malignant or benign using as few of the input dimensions as possible. For the purposes of classification, those with malignant tumour have been given class ‘1’ and benign, class ‘0’. The input data consists of ten real-valued features obtained from a suspicious tumour; radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry and fractal dimension. For each record the mean, standard error and worst error is provided (giving 30 dimensions in total).

A. GP run settings

All of the runs were performed using our free GPTIPS ([9],[10]) software for MATLAB. A custom fitness function was written for the purposes of binary classification, using the SFF described above.

In all the experimental runs the following GPTIPS settings were used: Population size = 200, Number of generations = 150, Tournament size = 6 (with lexicographic selection pressure), $D_{\max}$ = 5, $G_{\max}$ = 6, function node set = {plus, minus, times, tanh, sin, square, cos, max, min, if-then-else, exp}. Terminal set = {the input values x_1 – x_{30}, random ephemeral constants in the range [-10 10]}.

The following (default) recombination operator event probabilities were used: Crossover events = 0.80, mutation events = 0.1, direct reproduction = 0.1. The following sub-event probabilities were used: high level crossover = 0.2, low level crossover = 0.8, subtree mutation = 0.9, replace input terminal with another random terminal = 0.05, Gaussian perturbation of randomly selected constant = 0.05 (with standard deviation of Gaussian = 0.1). These settings were based on experience with the predictive modelling of other data sets of similar size, and so they may not be optimal.

To attempt to the prevent overfitting, 25% of the data were randomly selected for use as a holdout validation dataset. Holdout validation was performed as follows: at the end of each generation, the “best” individual (as evaluated on the training data) is then evaluated on the holdout validation set. The individual that performs best on the holdout set (over the course of the run) is stored and may be accessed after the run. To test the accuracy of the final models a further 25% of the data was used as an additional (unseen) testing data set.

A weighting of 0.9 was given to the malignant class in the breast cancer dataset to promote classification models / rules that misclassify cases that are malignant cancer and not benign; the safer misclassification. The length penalty p was set to 0.001.

B. Results

Due to the stochastic nature of GP, multiple runs (20) of the algorithm were performed. Table 1 shows the classification accuracy obtained.

TABLE I
CLASSIFICATION ACCURACY (WISCONSIN BREAST CANCER DATA)

Data Set	Best (%)	Mean (%)	Std. (%)
Training	96.8	90.3	3.8
Validation	94.9	88.4	3.7
Testing	94.8	90.3	2.8

For each GP run, good classification accuracies were obtained. The best classifier found, on the hold out validation data had a 94.9% classification accuracy with 88.4% of the misclassified cases were being classified as malignant - the more serious diagnosis. The mean prediction accuracy over the twenty runs was 88.4% on the validation data with a standard deviation of 3.7%. Similar performance was observed on the unseen testing data, with 94.8% classification accuracy. The mean number of nodes obtained within the classifiers was 37.3. The best classifier found had three genes and 29 nodes and is shown below

$$x_8(x_{21} - \sin(x_{30})) - \cos(x_{12})$$

$$x_{30}(x_{21} - 9.615) - (if(0.6370 > x_{29}) \\ then\ (0.6370) else (x_{29}))$$

$$x_{28}x_{21} - 9.615x_{28} + \sin(3.929 + x_{30})$$

This classifier has 3 genes and uses 6 of the input variables - x_8 concave points, x_{12} the texture (standard error), x_{21} the radius (worst error), x_{28} concave points (worst error), x_{29} the symmetry (worst error) and x_{30} the fractal dimension (worst error). The authors of [12] used this data set to compare 56 different classification algorithms available within Weka 3.5.5[2] (a suite of classification algorithms). Using the full

[1] http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[2] http://www.cs.waikato.ac.nz/ml/weka/

attribute set (all tumour dimensions) and 10 fold cross validation [12] examined the percentage of correctly classified instances. The best performing classifier correctly classified 97.9% of the data (reported over the entire data set). Two additional tests were performed using a reduced set of attributes. The first test used all records associated with mean area, mean perimeter and mean radius (giving nine inputs in total) as suggested by [13]. Here, the best performing classifier correctly classified 90.5% of the data. The second test used each of the 3 input records associated with mean texture, the worst area and the worst smoothness as suggested by [14]. In this case, the best classifier (a multilayer perceptron) correctly classified 97.2% of the data. Hence it can be seen that the evolved multigene model can achieve classifier performance of the order of the current state of the art classification algorithms. Moreover, the final model comprises a relatively compact team of mathematical models, which uses only six input dimensions.

V.CONCLUSIONS

In this article we have used GP to evolve multigene (team based) classifiers and demonstrated the approach with an application to a binary classification problem. It was shown that the evolved model was compact and offered similar high performance to published results using the same data. We emphasise that the multigene (team based) classifier methodology is not necessarily better or worse than other methods, but that it is a complementary alternative to existing approaches.

REFERENCES

[1] Koza JR. Genetic programming: on the programming of computers by means of natural selection. The MIT Press, USA, 1992.

[2] Jabeen, H. And Baig, A.R. Review of classification using Genetic Programming, Int. J. of Eng. Sci. And Tech., Vol 2 (2), 94 – 103., 2010.

[3] Bojarczuk, Lopes, H.S. and Freitas, A.A. An innovative application of a constrained-syntax genetic programming system to the problem of predicting survival of patients, Lecture notes in computer science, Vol. 2610, Springer-Verlag., 2003.

[4] Eggermont, J. Data Mining using Genetic Programming: Classification and Symbolic Regression. Leiden University, PhD Thesis., 2005.

[5] Taskonas, A. A comparison of classification accuracy of four genetic programming-evolved intelligent structures, Information Sciences, pp. 691-724., 2006.

[6] Ruta, D. and Gabrys, B. Classifier selection for majority voting, Information fusion 6, 63-81., 2005.

[7] Rogova, G. Combining results of several neural network classifiers, Neural Networks, 7 (5), pp 777-781., 1994.

[8] Hinchliffe MP, Willis MJ, Hiden H, Tham MT, McKay B & Barton, GW. Modelling chemical process systems using a multi-gene genetic programming algorithm. In Genetic Programming: Proceedings of the First Annual Conference (late breaking papers), 56-65. The MIT Press, USA, 1996.

[9] Searson, D. GPTIPS: Genetic Programming & Symbolic Regression for MATLAB, http://sites.google.com/site/gptips4matlab/, 2009.

[10] Searson, D.P., Leahy, D.E. and Willis, M.J. GPTIPS: An open source genetic programming toolbox for multigene symbolic regression. Lecture Notes in Engineering and Computer Science: Proceedings of the international multiconference of engineers and computer scientists IMECS 2010, 17-19 March, 2010, Hong Kong.