GECCO'2001, San Francisco, 7–11 July 2001, Morgan Kaufmann

Genetic Programming for Combining Classifiers

W. B. Langdon and **B. F. Buxton**
Computer Science, University College, London, Gower Street, London, WC1E 6BT, UK
{W.Langdon,B.Buxton}@cs.ucl.ac.uk
http://www.cs.ucl.ac.uk/staff/W.Langdon, /staff/B.Buxton
Tel: +44 (0) 20 7679 4436, Fax: +44 (0) 20 7387 1397

Abstract

Genetic programming (GP) can automatically fuse given classifiers to produce a combined classifier whose Receiver Operating Characteristics (ROC) are better than [Scott *et al.*, 1998b]'s "Maximum Realisable Receiver Operating Characteristics" (MRROC). I.e. better than their convex hull. This is demonstrated on artificial, medical and satellite image processing bench marks.

1 INTRODUCTION

[Scott *et al.*, 1998b] has previously suggested the "Maximum Realisable Receiver Operating Characteristics" for a combination of classifiers is the convex hull of their individual ROCs. However the convex hull is not always optimal [Yusoff *et al.*, 1998]. We show, on the problems used by [Scott *et al.*, 1998b], that genetic programming can evolve a combination of classifiers whose ROC are better than the convex hull of the supplied classifier's ROCs.

The next section gives the back ground to data fusion, Section 3 summarises Scott's work, his three bench marks are described in Section 4. The genetic programming system and its results are given in Sections 5 and 6. Finally we finish in Sections 7 and 8 with a discussion and conclusions.

2 BACKGROUND

There is considerable interest in automatic means of making large volumes of data intelligible to people. Arguably traditional sciences such as Astronomy, Biology and Chemistry and branches of Industry and Commerce can now generate data so cheaply that it far outstrips human resources to make sense of it. Increasingly scientists and Industry are turning to their computers not only to generate data but to try and make sense of it. Indeed the new science of Bioinformatics has arisen from the need for computer scientists and biologists to work together on tough data rich problems such as rendering protein sequence data useful. Of particular interest are the Pharmaceutical (drug discovery) and food preparation industries.

The terms Data Mining and Knowledge Discovery are commonly used for the problem of getting information out of data. There are two common aims: 1) to produce a summary of all or an interesting part of the available data 2) to find interesting subsets of the data buried within it. Of course these may overlap. In addition to traditional techniques, a large range of "intelligent" or "soft computing" techniques, such as artificial neural networks, decision tables, fuzzy logic, radial basis functions, inductive logic programming, support vector machines, are being increasingly used. Many of these techniques have been used in connection with evolutionary computation techniques such as genetic algorithms and genetic programming.

We investigate ways of combining these and other classifiers with a view to producing one classifier which is better than each. Firstly we need to decide how we will measure the performance of a classifier. In practise when using any classifier a balance has to be chosen between missing positive examples and generating too many spurious alarms. Such a balancing act is not easy. Especially in the medical field where failing to detect a disease, such as cancer, has obvious consequences but raising false alarms (false positives) also has implications for patient well being. Receiver Operating Characteristics (ROC) curves allow us to show graphically the trade off each classifier makes between its "false positive rate" (false alarms) and its "true positive rate" [Swets *et al.*, 2000]. (The true positive rate is the fraction of all positive cases correctly classified. While the false positive rate is the fraction of negative cases incorrectly classified as positive). Example ROC curves are shown in Figures 1 and 3. We treat each classifier as though it has a sensitivity parameter (e.g a threshold) which allows the classifier to be tuned. At the lowest sensitivity level the classifier produces no false alarms but detects no positive cases, i.e. the origin of the ROC. As the sensitivity is increased, the classifier detects more positive examples but may also start generating false alarms (false positives). Eventually the sensitivity may become so high that the classifier always claims each case is positive. This corresponds to both true positive and false positive rates being unity, i.e. the top right hand corner of the ROC. On average a classifier which simply makes random guesses will have an operating point somewhere on the line between the origin and 1,1 (see dotted line in Figure 3).

Naturally we want our classifiers to have ROC curves that come as close to a true positive rate of one and simultaneously a false positive rate of zero. In Section 5 we score each classifier by the area under its ROC curve. An ideal classifier has an area of one. We also require the given classifiers, not only to indicate which class they think a data point belongs to, but also how confident they are of this. Values near zero indicate the classifier is not sure, possible because the data point lies near the classifier's decision boundary.

Arguably "Boosting" techniques combine classifiers [Freund and Schapire, 1996]. However Boosting is normally applied to only one classifier and produces improvements by iteratively retraining it. Here we will assume the classifiers we have are fixed, i.e. we do not wish to retrain them. Similarly Boosting is normally applied by assuming the classifier is operated at a single sensitivity (e.g a single threshold value). This means on each retraining it produces a single pair of false positive and true positive rates. Which is a single point on the ROC rather than the curve we require.

3 "MAXIMUM REALISABLE" ROC

[Scott *et al.*, 1998b] describes a procedure which will create from two existing classifiers a new one whose performance (in terms of its ROC) lies on a line connecting the performance of its two components. This is done by choosing one or other of the classifiers at random and using its result. E.g. if we need a classifier whose false positive rate vs. its true positive rate lies on a line half way between the ROC points of classifiers A and B, then the Scott's composite classifier will randomly give the answer given by A half the time and that given by B the other half, see Figure 1. (Of course persuading patients to accept such a random diagnose may not be straightforward).

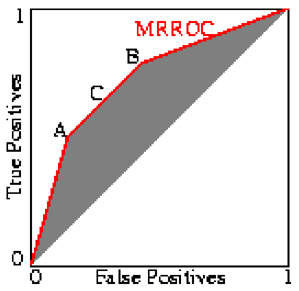

Figure 1: Classifier C is created by choosing equally between the output of classifier A and classifier B. Any point in the shaded area can be created. The "Maximum Realisable ROC" is its convex hull (solid line).

The performance of the composite can be readily set to any point along the line simply by varying the ratio between the number of times one classifier is used relative to the other. Indeed this can be readily extended to any number of classifiers to fill the space between them. The better classifiers are those closer to the zero false positive axis or with a higher true positive rate. In other words the classifiers lying on the convex hull.

Often classifiers have some variable threshold or tuning parameter whereby their trade off between false positives and true positives can be adjusted. This means their Receiver Operating Characteristics (ROC) are now a curve rather than a single point. Scott applied his random combination method to each set of points along the curve. So the "maximum realisable" ROC is the convex hull of the classifier's ROC. Indeed, if the ROC curve is not convex, an improved classifier can easily be created from it [Scott *et al.*, 1998b] (see Figure 4). The nice thing about the MRROC, is that it is always possible. But as we show it may be possible to do better automatically.

4 DEMONSTRATION PROBLEMS

[Scott *et al.*, 1998b] contains three benchmarks. Three of the following sections (4.2, 4.3 and 4.5) describe the preparation of the datasets. Sections 4.1 and 4.4 describe the two classifiers Scott used.

4.1 LINEAR CLASSIFIERS

In the first two examples (Sections 4.2 and 4.3) we use a tunable linear classifier for each data attribute (dimension). This classifier has a single decision value (a threshold). If examples of the class lie mostly at high values then, if a data point is above the threshold, the classifier says the data point is in the class. Otherwise it says it isn't. To produce a ROC curve the threshold is varied from the lowest possible value of the associated attribute to the highest.

To use a classifier in GP we adopt the convention that non-negative values indicate the data is in the class. We also require the classifier to indicate its "confidence" in its answer. In our GP, it does this by the magnitude of the value it returns.

(The use of the complex plane would allow extension of this signalling to more than two classes. Absolute magnitude would continue to indicate the classifiers confidence. While the complex plane could be divided into (possibly unequal) angular segments, one for each class. An alternative would be to allocate each class a point in the complex plane. The designated class would be the one closest in the complex plane. But if two class origins were a similar distance from the value returned by GP this would indicate the classifier was not sure which of the two classes to choose).

The linear classifier splits the training set at the threshold. When predicting, it uses only those examples which are the same side of the threshold as the point to be classified and chooses the class to which most of them belong. Its "confidence" is the difference between the number of training examples below the threshold in each class divided by their sum. Note the value returned to GP lies in the range $-1 \ldots +1$.

4.2 OVERLAPPING GAUSSIAN

Following [Scott *et al.*, 1998b, Section 3.1 and Figure 3] we created a training and a verification dataset, each containing 5000 randomly chosen data points. The points are either in class 1 or class 2. 1250 values were created using Gaussian distributions each with a standard deviation of 0.5. Those of class 1 had means of 3 and 7. While those used to generate class 2 data had means of 5 and 9. Note this gives rise to interlocking regions with some degree of overlap at their boundaries, see Figure 2.

Clearly a linear classifier (LC) with only a single decision point can not do well on this problem. Figure 3 shows its performance in terms of the trade off between false positives and true positives.

4.3 THYROID

The data preparation for the Thyroid problem follows Scott's. The data was down loaded from the UCI machine learning repository[1]. `ann.train` was used for the training set and `ann.train2` for the verification set.

[1]`ftp://ftp.ics.uci.edu/pub/machine-learning-databases/thyroid-disease`

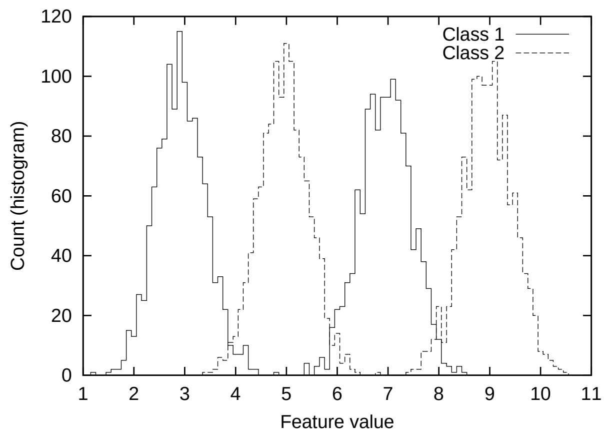

Figure 2: Example of a two class multi-modal data designed to be difficult for a linear classifier (Section 4.1).

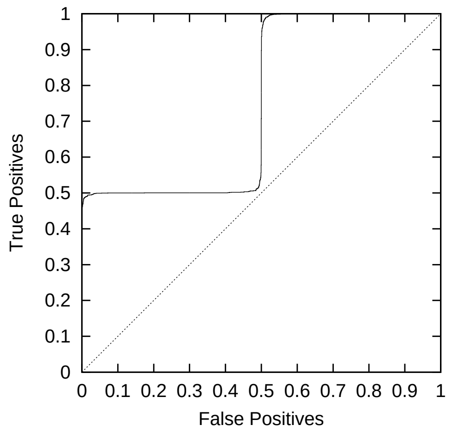

Figure 3: The Receiver Operating Characteristics curve produced by moving the decision boundary along the x-axis of Figure 2. The ROC are stepped as the classifier (Sect. 4.1) cannot capture the nature of the data.

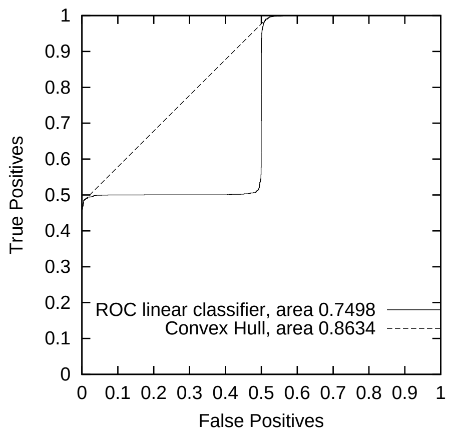

Figure 4: The convex hull of the ROC curve of Figure 3. Note a tunable classifier is improved by combining with itself, if its ROC are not convex.

(Both contain 3800 records). Originally it is a three class problem, the two classes for abnormal thyroids (79 and 199 records each in `ann.train`) were combined into one class. The GP is limited to using the two attributes (out of a total of 21) that Scott used. (Using all the attributes makes the problem much easier). Following strange floating point behaviour, both attributes were rescaled by multiplying by 1000. Rescaling means most numbers are integers between 1 and 200 (cf. Figure 10). Scott does not report rescaling. Two linear classifiers (LC18 and LC19) were trained, one on each attribute (D18 and D19) using the training set.

4.4 NAIVE BAYES CLASSIFIERS

The Bayes [Ripley, 1996; Mitchell, 1997] approach attempts to estimate, from the training data, the probability of data being in each class. Its prediction is the class with the highest estimated probability. We extend it 1) to include a tuning parameter to bias its choice of class and 2) to make it return a confidence based upon the difference between the two probabilities.

Naive Bayes classifiers are based on the assumption that the data attributes are independent. I.e. the probabilities associated with a data point are calculated by multiplying the estimates of the probabilities associated with each of its attributes.

The probabilities estimates of each class are based upon counting the number of instances in the training set for each attribute (dimension) that match both the point to be classified and the class, and dividing by the total number of instances which match regardless of the class. The estimates for each attribute are then multiplied together to give the probability of the data point being in a particular class.

The functions $P_{0,a}$ and $P_{1,a}$ use to estimate the probabilities for classes from training set attributes a

$$P_{c,a}(E) \quad = \quad Pr(\text{class} = c) \prod_{j \in a} Pr(X_j = v_j | \text{class} = c)$$

As an example, consider the data point $E = (6, 7, 8, 9, 10, 11, 12, 13)$ and a classifier using the set of attributes $a = \{2, 3, 5\}$. Then the probability E is in class 0, $P_{0,a}(E)$, is estimated to be, the probability of class 0 times, the probability that attribute 2 is 7 given the data is in class zero times, the probability attribute 3 is 8 (given the class is zero) times, the probability attribute 5 is 10 (given the class is zero). The calculation is repeated for the other classes (i.e. for class 1). The classifier predicts that E belongs to the class with the highest probability estimate. I.e. if $P_{0,a}(E) < P_{1,a}(E)$ then the Naive Bayes classifier (working on the set a of attributes) will predict the example data point E is in class 1, otherwise 0.

If there is no training data for a given class/attribute value combination, we follow [Kohavi and Sommerfield, 1996, page 11] and estimate the probability based on assuming there was actually a count of 0.5. ([Mitchell, 1997] suggests a slightly different way of calculating the estimates).

Since the denominators in $P_{c,a}$ are the same for all classes we can remove them and instead work with B

$$B_{c,a}(E) = \text{Number}(\text{class} = c) \prod_{j \in a} \text{Number}(X_j = v_j \cap \text{class} = c)$$

A threshold T ($0 \leq T \leq 1$), allows us to introduce a bias. I.e. if $(1 - T) \times B_{0,a}(E) < T \times B_{1,a}(E)$ then our Bayes classifier will predicts E is in class 1, otherwise 0. Finally we define the classifiers "confidence" to be $\frac{|B_{0,a}(E) - B_{1,a}(E)|}{(B_{0,a}(E) + B_{1,a}(E))}$.

4.5 GREY LANDSAT

Despite some care we have not been able to reproduce exactly the graphical results pictured in [Scott *et al.*, 1998a] and [Scott *et al.*, 1998b]. The Naive Bayes classifiers on the data we have appear to perform some what better. This makes the problem more challenging since there is less scope for improvement. [Scott *et al.*, 1998a] and [Scott *et al.*, 1998b] show considerable crossings in the ROC curves of the five classifiers they use. The absence of this in our data may also make it harder (see Figure 11).

The Landsat data comes from the Stalog project via the UCI machine learning repository[2]. The data is spilt into training (`sat.trn` 4425 records) and test (`sat.tst` 2000). Each record has 36 continuous attributes (8 bit integer values nominally in the range 0–255) and 6 way classification. (Classes 1, 2, 3, 4, 5 and 7). Following Scott; classes 3, 4 and 7 were combined into one (positive, grey) while 1, 2 and 5 became the negative examples (not-grey). `sat.tst` was kept for the holdout set.

The 36 data values represent intensity values for nine neighbouring pixels and four spectral bands (see Figure 5). While the classification refers to just the central pixel. Since each pixel has eight neighbours and

[2]`ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/satimage`

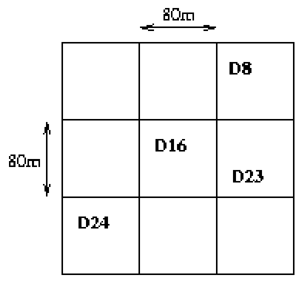

Figure 5: Each record contains data from nine adjacent Landsat pixels. Scott's five classifiers (nb16, nb16,23 nb16,23,24 nb23,24 and nb8,23,24) together use four attributes, Three (8, 16, 24) use spectral band 0 and the other (23) uses band 3. Notice how they straddle the central pixel in a diagonal configuration. However nb23,24 (which straddles both the area and the spectrum) has the best performance of Scott's Naive Bayes classifiers.

each may be in the dataset, data values appear multiple times in the data set. But when they do, they are presented as being different attributes each time. The data come from a rectangular area approximately five miles wide.

After reducing to two classes, the continuous values in `sat.trn` were partitioned into bins before it was used by the Naive Bayes classifier. Following [Scott *et al.*, 1998a, page 8], we used entropy based discretisation [Kohavi and Sommerfield, 1996], implemented in MLC++ `discretize.exe`[3], with default parameters. (Giving between 4 and 7 bins per attribute). To avoid introducing bias, the holdout data (`sat.tst`) was partitioned using the same bin boundaries.

`sat.trn` was randomly split into training (2956 records) and verification (1479) sets. The Bayes classifiers use the discrete data. In some experiments, the GP system was able to read data attributes values directly. In which case it used the continuous (floating point) value, rather than the attribute bin number.

5 GP CONFIGURATION

The GP is set up to signal its prediction of the class of each data value in the same was as the classifiers it can use. I.e. by return a floating point value, whose sign indicates the class and whose magnitude indicates the "confidence". (Note confidence is not constrained to lie in a particular range).

Following earlier work [Jacobs *et al.*, 1991; Soule, 1999; Langdon, 1998] each GP individual is composed of five trees. Each of which is capable of acting as a classifier. The use of signed numbers makes it natural to combine classifiers by adding them. I.e. the classification of the "ensemble" is the sum of the answers given by the five trees. Should a single classifier be very confident about its answer this allows it to "out vote" the all others.

We have not systematically experimented with the number of trees or alternative methods of combining them. The simplest problem can be solved with only one. Also in many individuals one or more of the trees appear to have little or a very basic function, such as always returning the same value or biasing the result by the threshold parameter.

5.1 FUNCTION AND TERMINAL SETS

The function set includes the four binary floating arithmetic operators ($+$, $\times$, $-$ and protected division), maximum and minimum and absolute maximum and minimum. The latter two return the (signed) value of the largest, (or smallest) in absolute terms, of their inputs. IFLTE takes four arguments. If the first is less than or equal to the second, IFLTE returns the value of its third argument. Otherwise it returns the value of its fourth argument. INT returns the integer part of its argument, while FRAC(e) returns e - INT(e).

The classifiers are represented as floating point functions. Their threshold is supplied as their single argument. As described in Sections 4.1 and 4.4.

The terminal T yields the current value of the threshold being applied to the classifier being evolved by GP. In some experiments the terminals Dn were used. These contain the value of attribute n. Finally the GP population was initially constructed from a number of floating point values. These constants do not change as the population evolves. However crossover and mutation do change which constants are used and in which parts of the program. GPQUICK limits the number of constants to about 200.

5.2 FITNESS FUNCTION

Each new individual is tested on each training example with the threshold parameter (T) taking values from 0 to 1 every 0.1 (i.e. 11 values). So, depending upon the problem, it is run 55000, 41800 or 32516 times. For each threshold value the true positive rate is calculated. (The number of correct positive cases divided by the total number of positive cases). If a floating point exception occurs its answer is assumed to be wrong. Similarly its false positive rate is given by the no. of negative cases it gets wrong divided by the total no. of negative cases. It is possible to do worse

[3]`http://www.sgi.com/Technology/mlc`

than random guessing. When this happens, i.e. the true positive rate is less than the false positive rate, the sign of the output is reversed. This is common practise in classifiers.

Since a classifier can always achieve both a zero success rate and 100% false positive rate, the points (0,0) and (1,1) are always included. These plus the eleven true positive and false positive rates are plotted and the area under the convex hull is calculated. The area is the fitness of the individual GP program. Note the GP individual is not only rewarded for getting answers right but also for using the threshold parameter to get a range of high scores. Cf. Table 1.

6 RESULTS

6.1 OVERLAPPING GAUSSIAN

In the first run the best fitness score (on the training data) was 0.981556. The first individual with this score was found in generation 21 and was treated as the output of the GP. Its total size (remember it has five trees) is 92. On another 5000 random data points its fitness was 0.981607. Its ROC are shown in Figure 6. (The linear classifier's convex hull area is 0.85).

Since we know the under lying distribution in this (artificial) example, we can calculate the optimal ROC curve, see Figure 6. The optimal classifier requires three decision boundaries, which correspond to the overlap between the four interlocking Gaussians. Figure 6 shows this GP individual has near optimal behaviour. Its output for one threshold setting (0.3) is given in Figure 7. Figure 7 shows GP has been able to use the output of the linear classifier to create three decision points (remember the linear classifier has just one) and these lie at the correct points.

Figure 8 shows, in each of the problems, little change in program size occurs after the first five generations or so. This is despite little or no improvement in the best fitness. This may be due to "size fair crossover" [Langdon, 2000].

6.2 THYROID

In one run the best fitness rose steadily to a peak of 0.838019 at generation 50. The program with this fitness has a total size of 60. On the verification set it has a fitness of 0.860040. Its ROC are shown in Figure 9.

Its bulk behaviour is to combine the two given (single attribute, single threshold) classifiers to yield a rectangular area near the origin. As the threshold is increased, the rectangle grows to include more data

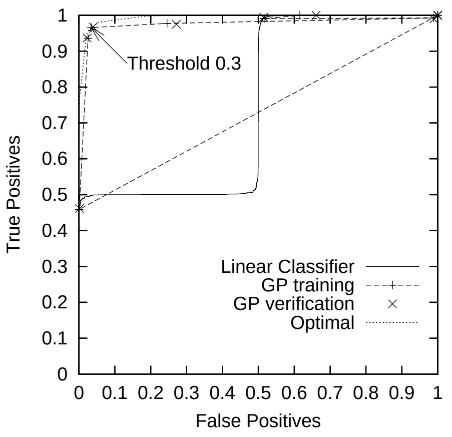

Figure 6: The ROC of GP (generation 21) classifier on interlocking Gaussians. Note it has near optimal performance.

points. Thus increasing the number of true positives, albeit at the expense of also increasing the number of false positive. Eventually with a threshold of 1, the rectangle covers all thyroid disease cases. Figure 10 shows the decision boundary for a threshold of 0.5. The superior performance of the GP classifier arises, at least in part, because it has learnt to recognise regularities in the training data. In particular it has spotted columns of data which are predominantly either all negative or positive and adjusted its decision boundary to cover these.

6.3 GREY LANDSAT

In the first GP run fitness rose quickly in the first six generations but much slower after that. The best training fitness was 0.981855 which was first discovered in generation 49. The ROC of this individual are shown in Figure 11. The area of its convex hull is bigger than all of those of its constituent classifiers. On the holdout set, its ROC are better than all of them, except for one threshold value where it has 3 false negatives v. 1 for the best of the Naive Bayes classifiers.

7 DISCUSSION

So far we have used simple classifiers with few parameters that are learnt. This appears to make them robust to over fitting. In contrast one often needs to be careful when using GP to avoid over fitting. In these experiments we have seen little evidence of over fitting. This may be related to the problems themselves or the choice of multiple tree programs or the absence of "bloat". The absence of bloat may be due

Table 1: GP Parameters (Variations between problems given in brackets or on separate lines)

Objective:	Evolve a function with Maximum Convex Hull Area	
Function set:	INT FRAC Max Min MaxA MinA MUL ADD DIV SUB IFLTE	
common plus	*Gaussians*	LC
	Thyroid	LC17 LC18
	Grey Landsat	nb16 nb16,23 nb16,23,24 nb23,24 nb8,23,24
Terminal set:	*Gaussians*	T, 0, 1, 200 unique constants randomly chosen in $-1 \ldots +1$
	Thyroid	T, D17, D18, 0, 0.1, 1, 212 unique constants randomly chosen from the test set.
	Grey Landsat	T 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Fitness:	Area under convex hull of 11 ROC points. (5000, 3800, 2956) randomly chosen test points	
Selection:	generational (non elitist), tournament size 7	
Wrapper:	$\geq 0 \Rightarrow$ positive, negative otherwise	
Pop Size:	500	
No size or depth limits		
Initial pop:	ramped half-and-half (2:6) (half terminals are constants)	
Parameters:	50% size fair crossover [Langdon, 2000], 50% mutation (point 22.5%, constants 22.5%, shrink 2.5% subtree 2.5%)	
Termination:	generation 50	

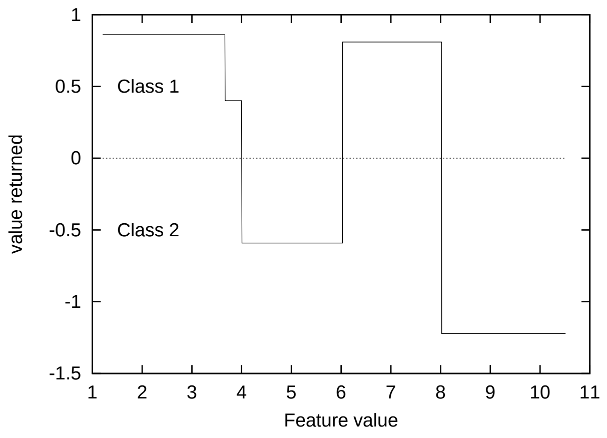

Figure 7: Value returned by evolved classifier (threshold=0.3) evolved on the interlocking Gaussians problem. High fitness comes from GP being able to use given classifier to distinguish each of the Gaussians. Note zero crossings align with Gaussians, Figure 2.

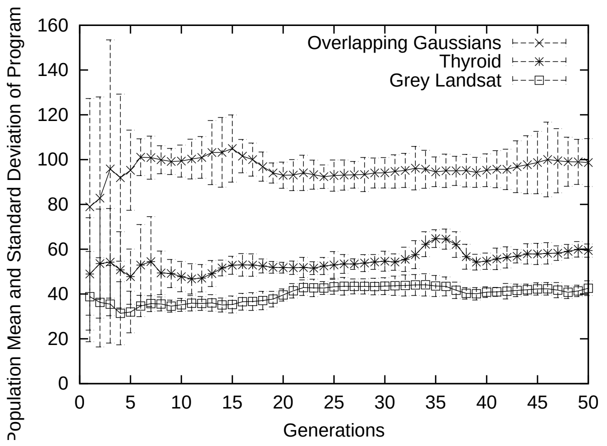

Figure 8: Evolution of total program size in one GP run of each the three problems.

to our choice of size fair crossover and a high mutation rate. Our intention is to evaluate this GP approach on more sophisticated classifiers and on harder problems. Here we expect it will be essential to ensure the classifiers GP uses do not over fit, however this may not be enough to ensure the GP does not.

8 CONCLUSIONS

[Scott *et al.*, 1998b] has proved one can always combine classifiers with variable thresholds to yield a composite with the "Maximum Realisable Receiver Operating Characteristics" (MRROC). Scott's MRROC is the convex hull of the Receiver Operating Characteristics of the individual classifiers. Previously we showed [Langdon and Buxton, 2001] genetic programming can in principle do better automatically. Here we have shown, using Scott's own bench marks, that GP offers a systematic approach to combining classifiers which may exceed Scott's MRROC. (Using [Scott *et al.*, 1998b]'s proof, we can ensure GP does no worse than the MRROC).

Mutation and size fair crossover [Langdon, 2000] mean there is little bloat.

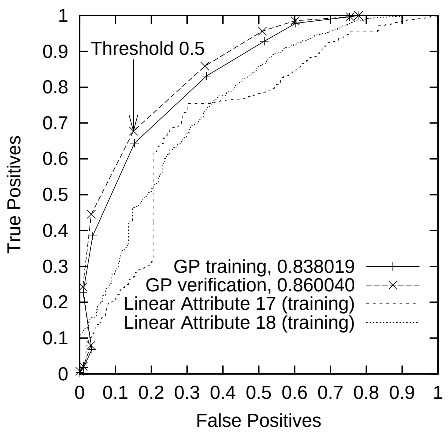

Figure 9: The ROC produced by GP (gen 50) using threshold values $0, 0.1, \ldots, 1.0$ on the Thyroid data.

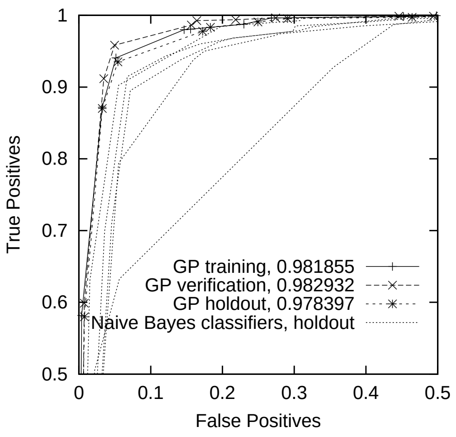

Figure 11: The ROC produced by GP (generation 49) using threshold values $0, 0.1, \ldots, 1.0$ on the Grey Landsat data. The ROC of the five given Naive Bayes classifiers are given on the holdout set.

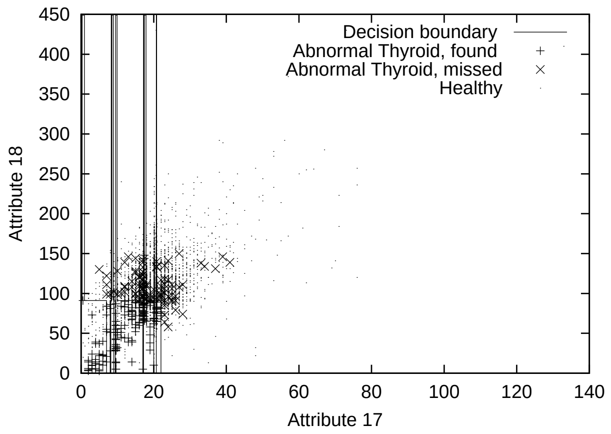

Figure 10: Decision boundary (threshold 0.5) for the Thyroid data produced by GP. The origin side of the boundary are abnormal (179 found, missed 99). 2982 correctly cleared, 540 false alarms.

References

[Freund and Schapire, 1996] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proc.* 13^{th} *International Conference*, pp 148–156. Morgan Kaufmann.

[Jacobs *et al.*, 1991] R. A. Jacobs, M. I. Jordon, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[Kohavi and Sommerfield, 1996] R. Kohavi and D. Sommerfield. MLC++: Machine learning library in C++. Technical report, `http://www.sgi.com/Technology/mlc/util/util.ps`.

[Langdon and Buxton, 2001] Evolving receiver operating characteristics for data fusion. In J. F. Miller *et al.*, eds., *EuroGP'2001*, *LNCS 2038*, pp 87–96, Springer-Verlag.

[Langdon, 1998] W. B. Langdon. *Data Structures and Genetic Programming.* Kluwer.

[Langdon, 2000] W. B. Langdon. Size fair and homologous tree genetic programming crossovers. *Genetic Programming & Evolvable Machines*, 1(1/2):95–119.

[Mitchell, 1997] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[Ripley, 1996] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press.

[Scott *et al.*, 1998a] M. J. J. Scott, M. Niranjan, and R. W. Prager. Parcel: feature subset selection in variable cost domains. Technical Report CUED/F-INFENG/TR.323, Cambridge University, UK.

[Scott *et al.*, 1998b] Realisable classifiers: Improving operating performance on variable cost problems. In P. H. Lewis and M. S. Nixon, eds., *Ninth British Machine Vision Conference*, pages 304–315,

[Soule, 1999] T. Soule. Voting teams: A cooperative approach to non-typical problems using genetic programming. In W. Banzhaf *et al.*, eds., *GECCO*, pages 916–922. Morgan Kaufmann.

[Swets *et al.*, 2000] J. A. Swets, R. M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, pages 70–75, October.

[Yusoff *et al.*, 1998] Combining multiple experts for classifying shot changes in video sequences. In *IEEE Int. Conf. on Multimedia Computing and Systems.*