Multiple Classifier Systems, Josef Kittler and Fabio Roli, editors, Cambridge, 2–4 July 2001, LNCS 2096, Springer-Verlag

Genetic Programming for Improved Receiver Operating Characteristics

W. B. Langdon and B. F. Buxton

Computer Science, University College, 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 of diverse types to produce a combined classifier whose Receiver Operating Characteristics (ROC) are better than [Scott *et al.*, 1998b]'s "Maximum Realisable Receiver Operating Characteristics" (MR-ROC). I.e. better than their convex hull. This is demonstrated on a satellite image processing bench mark using Naive Bayes, Decision Trees (C4.5) and Clementine artificial neural networks.

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 the best that can be achieved [Yusoff *et al.*, 1998]. Previously we showed [Langdon and Buxton, 2001a; Langdon and Buxton, 2001b] in at least some cases better ROCs can be automatically produced. We extend [Langdon and Buxton, 2001b] to show, on the problems derived from those proposed by [Scott *et al.*, 1998b], that genetic programming can automatically fuse different classifiers trained on different data to yield a classifier whose ROC are better than the convex hull of the supplied classifier's ROCs.

Section 2 gives the back ground to data fusion and Sect. 3 summarises Scott's work. The three classifiers are described in Sect. 4, while Sect. 5 describes the satellite data. The genetic programming system and its results are given in Sects. 6 and 7. Finally we finish in Sects. 8 and 9 with a discussion and conclusions. Sections 2–6 (excluding Sects. 4.1 and 4.3) are similar to [Langdon and Buxton, 2001b] however the experimental work (Sect. 7 onwards) extends [Langdon and Buxton, 2001b] to consider fusing classifiers of very different types.

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 [Langdon, 1998].

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). ROC curves are shown in Figs. 3 and 4. 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 (cf. Fig. 1).

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 Sect. 6 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, possibly because the data point lies near the classifier's decision boundary.

Arguably the well known "boosting" techniques combine classifiers to get a better one. 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's Parcel system [Scott *et al.*, 1998b] followed on from work on using wrappers for feature subset selection [Kohavi and John, 1997] and the use of ROC hulls [Provost and Fawcett, 2001]. However [Scott *et al.*, 1998b] describe a method to create, from two existing classifiers, a new one whose ROC lie on a line connecting the ROC of its two components. This is done by choosing one or other of the component classifiers at random and using its result. E.g. if we need a classifier whose false positive rate vs. its true positive rate lies 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 Fig. 1. (Of course persuading patients to accept such a random diagnose may not be straightforward).

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.

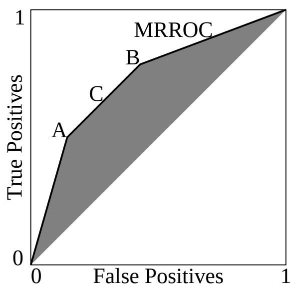

Fig. 1. Classifier C is created by choosing equally between the output of classifier A and classifier B. Any point in the shaded area can be created. The "Maximum Realisable ROC" is its convex hull (solid line).

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. We can apply Scott's random combination method to any set of points along the curve. So a "maximum realisable" ROC is the convex hull of the (single) classifier's ROC. Indeed, if the ROC curve is not convex, an improved classifier can easily be created from it [Scott *et al.*, 1998b]. The nice thing about the MRROC, is that it is always possible. But as we show it may be possible to do better.

4 Classifiers

4.1 C4.5

C4.5 [Quinlan, 1993], like the other classifiers, was extended to allow its use within our GP system. Each classifier takes a threshold parameter. To produce an ROC curve the threshold is varied from zero to one.

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.

C4.5 was run with defaults setting to produce pruned trees containing "confidence" values Z0 and Z1. Normally the decision tree's final classification would depend on which of Z0 and Z1 was the bigger. When the threshold is 0.5, this is what GP returns. However if it is near 0, GP is more likely to return class 0. While if the threshold is near 1, GP is biased towards class 1. (In detail GP returns class 0 iff $(1 - \text{threshold})\text{Z0} \geq \text{threshold Z1}$). This determines the sign of the value returned to the GP system. The magnitude is the C4.5 "confidence". This is $|\text{Z0} - \text{Z1}|$.

4.2 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.

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).

A threshold T $(0 \leq T \leq 1)$, allows us to introduce a bias. That is if $(1 - T) \times P_{0,a}(E) < T \times P_{1,a}(E)$ then our Bayes classifier will predicts E is in class 1, otherwise 0. ($P_{c,a}(E)$ is the probability estimated from the training data, using attributes from the set a, that E is in class c). Finally we define the classifiers "confidence" to be $|P_{0,a}(E) - P_{1,a}(E)|$.

4.3 Artificial Neural Networks

We used the Clementine data mining tool to train an artificial neural network to model the training data. This model was then frozen and made available to genetic programming as a function with one argument.

The ANN model was trained using Clementine version 5.0 on 2956 training records. Each record had nine integer inputs (from the last of the four spectral bands, see next section) and an integer range output. The output was 0 or 1 depending on whether the pixel was "grey" or not (see next section). The defaults were used, i.e. quick training, prevention of over training (50%), sensitivity analysis and default stopping criterion for training. The model has one hidden layer of four nodes, whose performance Clementine estimates to be 72.32%. (Performance of ANN models for the first band to third bands were estimated at 83.78%, 71.36% and 65.90%).

The neural net model gives a continuous valued output. Values below 0.5 indicate class 0. For use in GP, we subtract 1.0 and add the threshold parameter. This means values below zero indicate class 0, while non-negative values indicate class 1. As usual the continuous threshold parameter allows us to tune the neural network to trade off false positive against true positives and so obtain a complete ROC curve rather than a single error rate. (A threshold of 0.5 indicates no bias, i.e. use the raw neural network). Notice that the "confidence" the GP sees is directly related to how far from the neural networks idle value (0.5) its output is.

5 Grey Landsat

The Landsat data comes from the Stalog project via the UCI machine learning repository[1]. 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 [Scott *et al.*, 1998b], 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 Fig. 2). While the classification refers to just the central pixel. Since each pixel has eight neighbours and 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 all come from a rectangular area approximately five miles wide. Each of the three types of classifiers is trained on data from one spectral band. (Naive Bayes - first band, C4.5 - second band, artificial neural network - last band).

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

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

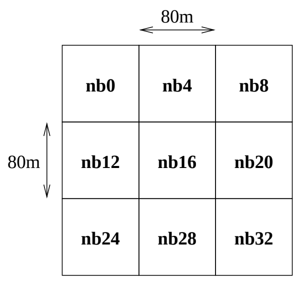

Fig. 2. Each record contains data from nine adjacent Landsat pixels. In these experiments, all of the Naive Bayes classifiers are trained on the first spectral band. There are two types of Naive Bayes classifiers, single attribute and those trained on pairs of attributes. All nine single attribute and all pairs of attributes 0, 4, 12, 16, 20 and 32 are available to GP. C4.5 was trained on nine attributes 1, 5, ... 33 (second band) while the ANN was trained on the fourth band (attributes 3, 7, ... 35).

Sommerfield, 1996], implemented in MLC++ `discretize.exe`[2], 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.

6 Genetic Programming Configuration

The genetic programming system is almost identical to that described in [Langdon and Buxton, 2001b]. The GP is set up to signal its prediction of the class of each data value in the same way as the classifiers. I.e. by returning a floating point value, whose sign indicates the class and whose magnitude indicates the "confidence". (Note confidence is not constrained to 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" all the others.

6.1 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.

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

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 Sect. 4.

The terminal T yields the current value of the threshold being applied to the classifier being evolved by GP. 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.

6.2 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 it is run 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 worst 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. Parameters are summarised in Table 1.

Table 1. Grey Landsat GP Parameters

Objective:	Evolve a function with Maximum Convex Hull Area
Function set:	INT FRAC Max Min MaxA MinA MUL ADD DIV SUB IFLTE C4.5 ANN (nb0 nb4 nb8 nb12 nb16 nb20 nb24 nb28 nb32 nb0,4 nb0,12 nb0,16 nb0,20 nb0,32 nb4,12 nb4,16 nb4,20 nb4,32 nb12,16 nb12,20 nb12,32 nb16,20 nb16,32 nb20,32)
Terminal set:	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 on 2956 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

7 Results

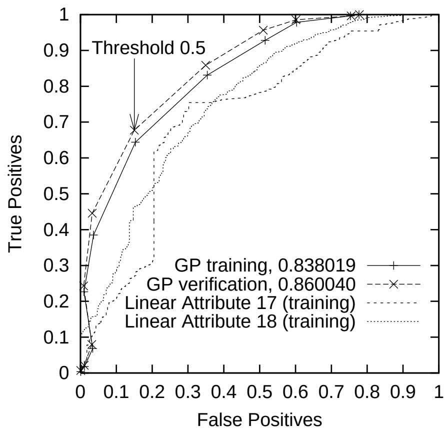

Fig. 3. The ROC produced by GP (generation 50) using threshold values $0, 0.1, \ldots, 1.0$ on the Thyroid data. Details of the experiment are reported in [Langdon and Buxton, 2001b].

The three types of classifier (C4.5, Naive Bayes and ANN) were made available to GP, singly, in pairs and finally all three together. I.e. seven experiments were run. (The 21 Naive Bayes classifiers are treated as a group, i.e. they are either all included or all excluded). In each run, GP's answer was chosen as the first occurrence of a program with the the largest ROC area (on the training data) found in the whole run. The ROC of these seven programs (on the hold-out data) are plotted in Fig. 4 and tabulated in Table 2. In all seven cases GP automatically produced a classifier with better performance than those it was given. That is genetic programming fused classifiers of different types, trained on different data, to yield superior classifiers.

8 Over Fitting

We have taken some care to ensure our input classifiers do not over fit the training data. Similarly one needs to be careful when using GP to avoid over fitting. So far 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 to our choice of size fair crossover [Langdon, 2000] 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.

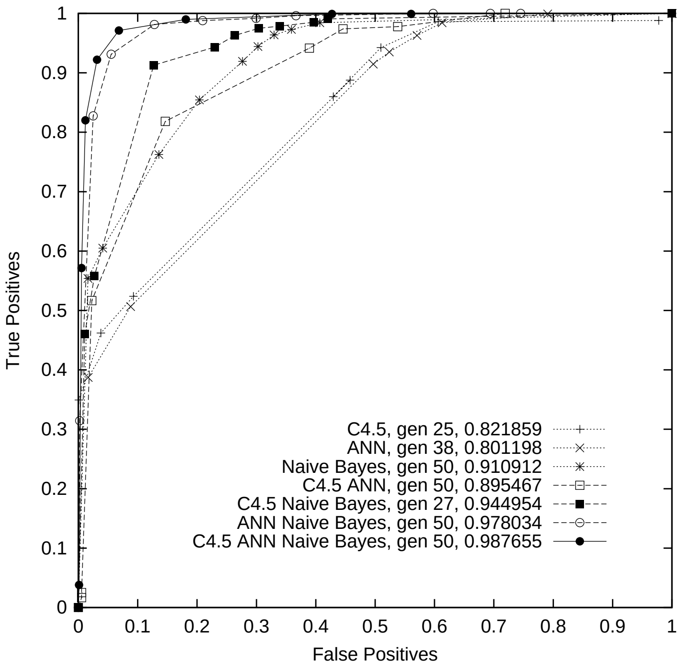

Fig. 4. The ROC produced by GP using seven combinations of classifiers on the Grey Landsat holdout data. The caption gives the area under the ROC (holdout) and the first generation to give the maximum area on the training data. (For simplicity only the convex hull of each classifier is plotted).

Table 2. Grey Landsat, Area under ROC on holdout set

		Given Classifiers				Genetic Programming	
				nb0,12	0.877244		
				nb0,16	0.880381		
				nb0,20	0.895346		
		nb0	0.873961	nb0,32	0.893585		
		nb4	0.883431	nb0,4	0.883431	ANN	0.801198
		nb8	0.886417	nb4,12	0.890616	C4.5	0.821859
ANN	0.764945	nb12	0.877244	nb4,16	0.888611	nb	0.910912
C4.5	0.74271	nb16	0.880381	nb4,20	0.89729	ANN + C4.5	0.895467
		nb20	0.895346	nb4,32	0.901857	ANN + nb	0.978034
		nb24	0.888831	nb12,16	0.888113	C4.5 + nb	0.944954
		nb28	0.8898	nb12,20	0.896233	all	0.987655
		nb32	0.893585	nb12,32	0.891777		
				nb16,20	0.901793		
				nb16,32	0.892662		
				nb20,32	0.900468		

9 Conclusions

Previously [Langdon and Buxton, 2001a] we showed, using Scott's own bench marks, that genetic programming can do better than [Scott *et al.*, 1998b]'s MRROC [Langdon and Buxton, 2001b]. Here we have shown, GP can deal not only with different classifiers but with classifiers of different types, trained on different data. Genetic programming offers an automatic means of data fusion by evolving combined classifiers.

References

[Jacobs *et al.*, 1991] Robert A. Jacobs, Michael I. Jordon, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

[Kohavi and John, 1997] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1–2):273–323, 1997.

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

[Langdon and Buxton, 2001a] W. B. Langdon and B. F. Buxton. Evolving receiver operating characteristics for data fusion. In Julian F. Miller, *et. al.* editors, *Genetic Programming, Proceedings of EuroGP'2001*, *LNCS* 2038, pages 87–96. Springer-Verlag.

[Langdon and Buxton, 2001b] W. B. Langdon and B. F. Buxton. Genetic programming for combining classifiers. In *GECCO'2001*. Morgan Kaufmann.

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

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

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

[Provost and Fawcett, 2001] Foster Provost and Tom Fawcett. Robust classification for imprecise environments. *Machine Learning*, 42(3):203–231, 2001.

[Quinlan, 1993] *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[Ripley, 1996] *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 Engineering Department, Trumpington Street, CB2 1PZ, UK.

[Scott *et al.*, 1998b] M. J. J. Scott, M. Niranjan, and R. W. Prager. Realisable classifiers: Improving operating performance on variable cost problems. In Paul H. Lewis and Mark S. Nixon, editors, *Ninth British Machine Vision Conference*, volume 1, pages 304–315, University of Southampton, UK, 14-17 September 1998.

[Soule, 1999] Terence Soule. Voting teams: A cooperative approach to non-typical problems using genetic programming. In Wolfgang Banzhaf, *et. al.* editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 916–922. Morgan Kaufmann.

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

[Yusoff *et al.*, 1998] Y. Yusoff, J. Kittler, and W. Christmas. Combining multiple experts for classifying shot changes in video sequences. In *IEEE International Conference on Multimedia Computing and Systems*, volume II, pages 700–704, Florence.