Soft Computing and Industry Recent Applications, R. Roy, M. Koppen, S. Ovaska, T. Furuhashi and F. Hoffmann, editors, pages 597–608, Springer-Verlag, 2002.

Genetic Programming for Combining Neural Networks for Drug Discovery

W. B. Langdon[1], S. J. Barrett[2], and B. F. Buxton[1]

[1] 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
[2] GlaxoSmithKline Research and Development, Harlow, Essex, UK

Abstract. We have previously shown on a range of benchmarks [Langdon and Buxton, 2001b] 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.*, 1998]'s "Maximum Realisable Receiver Operating Characteristics" (MRROC). I.e. better than their convex hull. Here our technique is used in a blind trial where artificial neural networks are trained by Clementine on P450 pharmaceutical data. Using just the networks, GP automatically evolves a composite classifier.

1 Introduction

There are an increasing range of cases where computer based systems are able to provide huge volumes of data but rendering it intelligible is seldom attempted or left to labour intensive intervention. This has provoked interest in artificial intelligence techniques to try and extract information from the data. A common task is classification. Here we are particularly interested in data rich Cheminformatics applications where we wish to be able to predict how chemicals, particularly potential drugs, will behave. Intelligent classification techniques such as artificial neural networks (ANN) have had limited success at predicting potential drug activity. Using genetic programming many diverse classifiers can be fused to yield a superior classifier.

Any classifier makes a trade off between catching positive examples and raising false alarms. Where the costs of these are not known in advance it may be useful to be able to tune the classifier to favour one over the other. The Receiver Operating Characteristics (ROC) of a classifier provides a helpful way of illustrating this trade off.

[Scott *et al.*, 1998] 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, 2001b] in at least some cases better classifiers, in terms of their ROCs, can be automatically produced. Here we apply our technique to a real world application namely classifying potential drug compounds as to whether they are inhibitors of a P450 enzyme.

Section 2 gives the back ground to data fusion. The collection of the P450 data at GlaxoSmithKline Pharmaceuticals is described in Sect. 3. Section 4 describes using the SPSS Clementine data mining tool to train 60 artificial neural networks on 699 chemical features. These ANN are used as the primary classifiers by the genetic programming data fusion system, which is described in Sect. 5. The results are given in Sect. 6. Finally we conclude in Sect. 7.

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 drug discovery.

The terms Data Mining and Knowledge Discovery are commonly used for the problem of getting information out of data. 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. Genetic programming, using Receiver Operating Characteristics (ROC), offers an automatic way of combining diverse classifiers to yield a superior composite.

More information on ROC curves can be found in our previous work [Langdon and Buxton, 2001a; Langdon and Buxton, 2001b; Langdon and Buxton, 2001c] and the companion web pages to this paper (`http://www.cs.ucl.ac.uk/staff/W.Langdon/wsc6/`). Briefly any binary classifier can be characterised by two scalars. Its "true positive" rate (TP) and its "false positive" rate (FP). I.e. the fraction of positive examples it correctly classifies and the fraction of negative examples it gets wrong (false alarms). When plotted against each other TP v. FP lie inside a unit square. An ideal classifier has TP = 1 and FP = 0. I.e. the upper left corner of the square (see Fig. 4). Many classifiers have a sensitivity parameter. This allows the user to trade off TP against FP. By varying the sensitivity the FP,TP point traces a curve. A good classifier will have a curve which lies as close to (0,1) as possible. A very poor classifier's ROC will lie near the diagonal (0,0) – (1,1). It is common to use the area under the ROC as a measure of the classifier's performance. (Although a single scalar measurement cannot capture all the possible variations between two curves it is widely used and is in most cases satisfactory).

A new classifier can always be constructed by randomly choosing between two available classifiers. [Scott *et al.*, 1998] showed that the ROC of the new classifier lies on a line between the ROC's of the two real classifiers. If we choose evenly (50%), then the new point lies halfway between them. (The element of chance may mean it cannot be used in some applications). Changing the ratio from

50% moves the point from the midpoint towards one of the existing classifiers. Since this can be done for any number of classifiers operating at any of their sensitivity parameter settings, a new classifier can be constructed at any point within the convex hull of their ROC curves. In fact only those on the convex hull are of interest, since they are bound to be better than points within the hull. Scott showed a new classifier whose ROC is the convex hull of the input classifiers ROC curves can indeed be constructed in practise. He called this the "Maximum Realisable Receiver Operating Characteristics" (MRROC). In fact it is possible in principle to do better and as we shall show, GP can do better in this application.

3 The Pharmaceutical Data

Volume inhibition data was extracted from the GlaxoSmithKline biological results database for all compounds screened against a P450 enzyme assay using high through put screening (HTS). If all data for an individual compound were initially within 15% inhibition points of each other they were retained and averaged. Otherwise noisy data was discarded. The clean data was then thresholded at an appropriate level of inhibition dividing the compounds into "actives" (enzyme inhibitors) and "inactives" (non-inhibitors).

The actives were separated from the inactives and each set hierarchically clustered separately using Ward's linkage in combination with Tanimoto similarity, computed from Daylight 2Kbit string chemical fingerprint data. Clusters were defined at 0.8tan (Tanimoto) for the (smaller) actives set and at 0.75 for the (larger) inactives set, as a first step to reduce the gross imbalance in actives vs. inactives. Then, from each cluster in each partitioning, centroid compounds were selected to reduce the internal chemical structure bias and the volume of the original data. This noise removal and clustering resulted in a "controlled diversity" dataset of 2256 compounds with a balance of approximately 4 inactives for every active. A total of 699, 2-d numerical, chemical features from a diverse array of families (electronic, structural, topological/shape, physico-chemical, etc.) were then computed for each centroid molecule, starting from a SMILES representation of it's primary chemical structure. 1500 compounds (300 actives, 1200 inactives) were selected for use as the training set, whilst the remaining 756 were retained as a separate "holdout" set.

4 Artificial Neural Networks

We used the Clementine data mining tool to train artificial neural networks (ANN) to model the training data. These models were then frozen and made available to genetic programming as a function. ANN models were generated using "train net" node using the "Quick" (BackProp) method, with options to prevent over training set (50% training data used) and to stop on 300 cycles. A fixed random seed was employed.

The ANN models were trained using Clementine on subsets of 1500 training records, containing the 699 features. These were divided by GlaxoSmithKline into 15 groups of about 50 attributes (features). Four single layer perceptrons were trained on each group.

It is well know that this kind of neural network performs best when trained on "balanced data sets", i.e. data sets containing an equal mix of active and inactive examples. However drug discovery tasks are seldom like this. It is common, as here, for many compounds to be inactive and only a few to be active. Four data sets were prepared. Each contained the same 300 active examples and 300 different inactive examples. That is each data set was balanced. Each neural network was trained on one of the 15 groups of attributes selected from one of the four balanced data sets. Making a total of 60 networks.

A script was written which automatically manipulated the Clementine stream (see Fig. 2). The script feeds one of the four sets of balanced examples via a "type node" (which selects the FEATURE SET) into Clementine's artificial neural network trainer (TRAIN NET NODE in Fig. 1). The script loops through each of the four balanced training sets and loops through the 15 groups of chemical attributes, exporting neural network models for each combination of FEATURE SET and DATA SET.

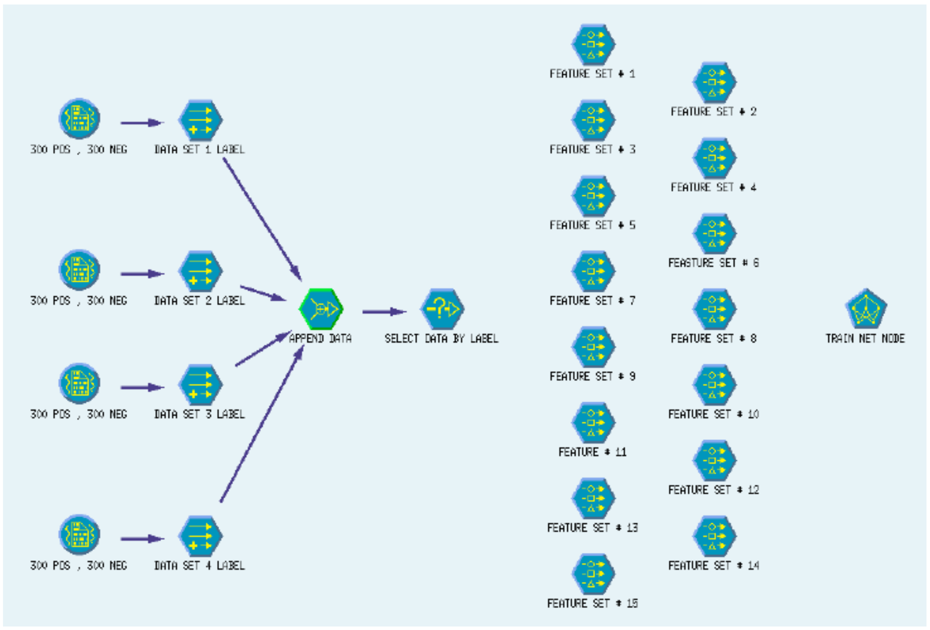

Fig. 1. Clementine streams for training multiple Artificial Neural Networks. The command script for switching the training stream between the data files is given in Fig. 2

```
For F in Feature_Set#'s 1 to 15   # selects FEATURE SET typenode name
  For [dataset] N from 1 to 4   # select DATA SET
    set m=F><N                 # concatenates FEATURE SET & DATA SET names
    set :trainnet.netname = M         # rename TRAIN NET node
    connect ^F between select and ^M  # connects dataset to 'feaset'
    execute : trainnet
    disconnect ^F
    export generated ^M in [file directory] codexport
  endfor
endfor
```

Fig. 2. Clementine script used to train 15 groups of Artificial Neural Networks. The ANN in each group are trained on a group of about 50 chemical features. Each group contains four networks, each trained on the same active chemicals but different inactive examples. Cf. Fig. 1

5 Genetic Programming Configuration

The genetic programming data fusion system is deliberately almost identical to that described in [Langdon and Buxton, 2001b].

5.1 Function Set

The Function set is the collection of basic floating point operations that are available to form the individual programs in the GP population. They (and the other GP parameters) are summarised in Table 1. The functions include the four basic arithmetic operations "+", "−", "×" and "÷". Note however ÷ by zero always yields "1". This "protects" it and prevents the GP system failing with a divide-by-zero fault. Max (Min) takes two arguments and returns the value of the largest (smallest). MaxA (MinA) also takes two arguments but returns the signed value of the largest (smallest) in absolute terms. E.g. MaxA(-2,1) returns -2. INT returns the integer part of its input. E.g. INT(3.23) returns 3. FRAC returns the fractional part of its input. E.g. FRAC(3.23) returns 0.23. Finally 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. E.g. FRAC(0, 0.345, ANN1..., ANN2...) returns the value given to it by the subtree starting with ANN1.

In order to use the neural networks within the GP they are packaged up and presented to GP as 60 problem specific functions. (The GP is run separately from Clementine using 60 files containing the ANNs. Note the ANN are frozen and not retrained). Each returns the classification given by the corresponding neural network for the current chemical. Clementine codes its multi-layer perceptron neural networks to have one output neuron which yields a floating point value between zero and one. For a mid-point threshold, this is exactly the value returned by the function inside the GP system. Values near either zero or one mean the neural network is highly confident of its answer. While values near 0.5 suggest the classifier is less confident. Normally the output of the neural network is converted into a binary classification (i.e. the chemical is active or is inactive) by testing to see if the value is greater or less than 0.5. This gives a single point in the ROC square. I.e. one trade off between catching all positives but raising too many false alarms. However we can change this trade off. So that instead of getting a single point, we get a complete curve in the ROC square. This is easily done by replacing the fixed value of 0.5 by a tunable threshold. By continuously varying the threshold from below zero to above one the (binarized) output of any of the neural networks will be biased from saying every chemical is inactive, through the usable range, to catching all positive examples but being 100% wrong on the negative examples (by saying all chemicals are active). In fact we leave the choice of suitable operating point to the GP to automatically evolve. This is done by making the threshold point an argument to the function. These arguments are treated like any other by the GP and so can be any valid arithmetic operation, including the neural networks themselves.

5.2 Terminal Set

The terminals or leafs of the trees being evolved by the GP are either constants or the threshold (see Table 1). The threshold allows us to change the bias of the tree produced by GP. As with the neural networks, changing the bias allows each tree to sweep out an ROC curve rather than operate at a single point. This is done by running each tree with the threshold taking values 0, 0.1, 0.2, ... 1.0 (i.e. 11 values). Note, as with all the functions and terminals, it is entirely up to the evolutionary process how it uses the threshold parameter.

5.3 Representation

The GP is set up to signal its prediction of the class of each data value by returning a floating point value, whose sign indicates the class and whose magnitude indicates the "confidence".

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. Note that although this has some similarity with some neural network "ensembles", the GP can combine the supplied classifiers in an almost totally arbitrary non-linear way. It is not constrained to a weighted linear sum of all or even a subset of them.

Table 1. GP Data Fusion Parameters

Objective:	Evolve a Non-Linear Combination of Neural Networks with Maximum ROC Convex Hull Area on P450
Function set:	INT FRAC Max Min MaxA MinA MUL ADD DIV SUB IFLTE 60 ANN trained on P450 data
Terminal set:	T 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 plus 100 unique random constants 0..1
Fitness:	Area under convex hull of 11 ROC points.
Selection:	generational (non elitist), tournament size 7
Wrapper:	$\geq 0 \Rightarrow$ active, inactive otherwise
Pop Size:	500
No size or depth limits	
Initial pop:	ramped half-and-half (5:8) (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

5.4 Fitness Function

The fitness function is crucial to any optimisation process, including evolutionary computation techniques. By giving intermediate results a score (in GP known as the individual program's fitness) it guides the optimiser. Often fitness functions are not given sufficient thought. For example when classifying, one scheme is simply to assign an individual's fitness equal to the number of correct predictions it makes. In some cases this is sufficient. But where one class is much more common than the other, this can lead to a trap where the optimiser creates a classifier which always says which ever class occurs most often. This classifier has a high score but no predictive ability. By using ROC curves we avoid this trap.

Since we may not know in advance the trade off between the costs of misclassification of the two classes, we use GP to produce a *tunable* classifier. We asses the usefulness of each candidate classifier produced by GP from its Receiver Operating Characteristics (ROC) curve.

To assign a fitness to each evolved classifier. The tuning parameter is set to values 0.1 a part, starting at 0 and increasing to 1. For each setting it is used to predict the activity of each chemical in the training set. These predictions are compared with measured activity. The proportions of active chemicals correctly predicted (TP) and the proportion of inactive one incorrectly predicted (FP) are calculated. Each TP,FP pair gives a point on a curve. The fitness of the classifier is the area under the convex hull of these (plus the fixed points 0,0 and 1,1).

5.5 Genetic Operations and other Parameters

As mentioned above, we used the same mix of genetic operators as had proved themselves in earlier experiments. Other work (for example [Angeline, 1998]) has suggested that a high mutation rate and a mixture of different mutation operators for evolving pattern matching functions. As described in Table 1, 50% of new classifiers were created by random changes of a single parent from the previous generation. On average 112.5 programs per generation were created by point mutation on the functions and 112.5 by random changes to the constants. Point mutation was applied on average to 10% of functions within the program. Point mutation on a function randomly replaces it with another (which takes the same number of arguments). Mutating a constant, replaces it with another randomly constant. However the new constant is not chosen uniformly, instead a new value is chosen from an approximately Normal distribution centred on the current value and standard deviation of 5% of the mean. There are 110 available constants. The one closest to the randomly generated value replaces the original constant. (The new constant will be different from the original). Note bigger programs will on average have more changes made to them.

Two other mutation operators were also used. On average 12.5 programs in each generation were created using "subtree" mutation and 12.5 by "subtree shrink" mutation. Both select a subtree within (a copy of) the parent program and replace it with another. Note unlike point mutation and constant mutation, these mutation operators change the size and shape of the program. In subtree mutation the new subtree is created using the same algorithm used to create the initial random population, i.e. "ramped-half-and-half" [Koza, 1992, pages 92–93]. This tends to increase the size of programs. (The behaviour of ramped-half-and-half is discussed in [Luke and Panait, 2001]). In contrast "subtree shrink" replaces the subtree with another chosen at random from within it. This must be smaller leading to the new program being smaller than the program in the previous generation from which it was created.

On average 250 of the new programs are created using size fair crossover [Langdon, 2000]. In genetic programming crossover selects two fit programs to be parents for a new one. The new program is composed of parts drawn from (copies of) the two parents. It thus provides an analogue of sexual reproduction in the artificial evolution used by genetic programming. There are a number of crossover operators available in GP [Langdon and Poli, 2001], size fair crossover was introduced to reduced the tendency seen in the usual GP crossover operator [Koza, 1992] for programs in successive generations to increase in size (known as bloat [Langdon *et al.*, 1999]) but still allow size and shape to be determined by the evolutionary process rather than by hard limits or initial conditions.

Size fair, like Koza's subtree crossover, creates a new program by selecting a subtree from each parent program. In one the subtree is discarded and replaced by the subtree selected in the other parent. The difference is, in the older crossover both subtrees are selected independently at random. In size fair the subtree to be removed is selected at random but the the subtree to be inserted is chosen from those in the other parent program of a chosen size. The size is given by the size of the subtree to be deleted. To allow size fair crossover to change the size of the new program, the inserted subtree's size need not be identical to that being deleted but is randomly chosen from 1 to up to twice the size of the subtree tree being deleted. However care is taken to ensure size fair does not, of itself, lead to either an average increase or decrease in size. (Details are given in [Langdon, 2000]).

5.6 GP Training Data

The 1500 examples used to train the neural networks were randomly split into 1000 to be used to training the GP and 500 (containing 100 active examples) kept back as a verification set.

6 Results

In one run the GP evolved a combined classifier with a fitness of 0.90 on the training data and 0.86 on both the validation and holdout data. Figure 3 shows its performance (on the training set vs. the holdout set) in comparison with that of the neural networks. If performance on the training set and holdout were identical, then all the points would lie along the diagonal and there would be no over fitting. In practise, since we are dealing with finite samples, there is bound to be some statistical scatter. As scatter of points is around the diagonal this indicates there is little over fitting.

To obtain improved performance from any classifier fusion the input classifiers must be different. Figure 3 shows their performance varies considerably. Doing pairwise comparisons, most ANN are significantly different (McNemar's test at 5%) from each other on the training data. The GP selected only 17 ANN for inclusion in the evolved classifier. These are shown with double crosses in Fig. 3. Figure 3 shows GP has selected a broad range of ANNs.

Figure 4 shows the Receiver Operating Characteristics of the evolved classifier, measured on its training data and on the holdout set. There is slight reduction in performance on the holdout set, potentially indicating some over fitting. Figure 4 also shows, for comparison, the ROC of the MRROC classifier produced by taking the convex hull of the 60 ANNs. The MRROC is shown measured on the training data and on the holdout data. Of course the MRROC is convex on the training data, but need not be on the hold out data. On both the training and holdout data, the evolved classifier is slightly better than the MRROC combination of the 60 ANN trained by Clementine and therefore better then each of the ANN individually.

7 Conclusions

Where accuracy is paramount (and so its stochastic basis can be ignored) [Scott *et al.*, 1998]'s convex hull classifier, MRROC, offers an automatic means of combining classifiers. However it is not guaranteed to be optimal. In [Langdon and Buxton, 2001b] we showed, using Scott's own bench marks, that genetic programming can do better than the MRROC both in theory and practise. Nevertheless we cannot guarantee GP will always do better and so it is important to demonstrate it on interesting applications. Here we have shown (cf. Figs. 3 and 4) that our GP technique can be used in a large classification application related to drug discovery with just a few minutes of computer processing time.

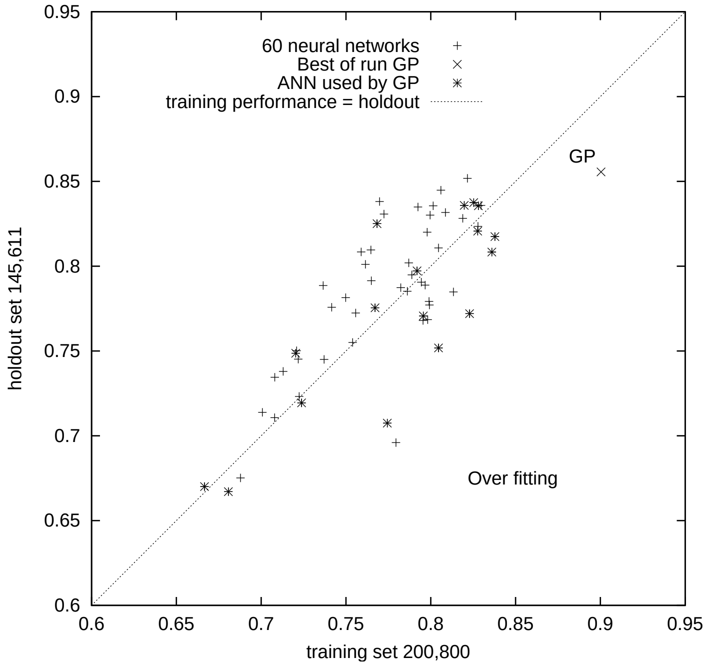

Fig. 3. Performance of 60 given neural networks and genetic programming. The area under the Receiver Operating Characteristics (ROC) is plotted on the training data (horizontal) v. holdout data (vertical). Points below the diagonal indicate a degree of over training.

References

Angeline, 1998. Peter J. Angeline. Multiple interacting programs: A representation for evolving complex behaviors. *Cybernetics and Systems*, 29(8):779–806, November 1998.

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, 1991.

Koza, 1992. John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992.

Langdon and Buxton, 2001a. W. B. Langdon and B. F. Buxton. Evolving receiver operating characteristics for data fusion. In Julian F. Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea G. B. Tettamanzi, and W. B. Langdon, editors, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 87–96, Lake Como, Italy, 18-20 April 2001. Springer-Verlag.

Langdon and Buxton, 2001b. W. B. Langdon and B. F. Buxton. Genetic programming for combining classifiers. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 66–73, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

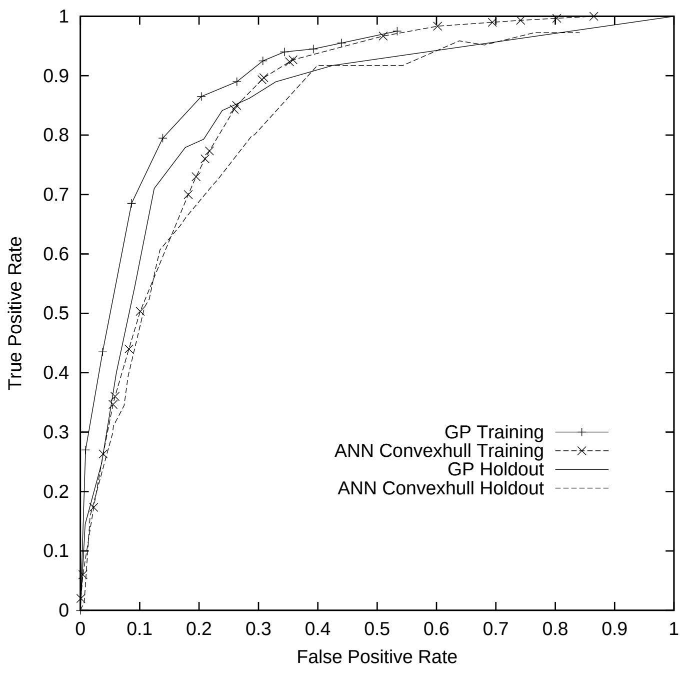

Fig. 4. Receiver Operating Characteristics of evolved composite classifier. For comparison the convex hull of the 60 given neural networks, on the training and holdout data, is given. Note the convex hull classifier is no longer convex when used to classify the holdout data.

Langdon and Buxton, 2001c. W. B. Langdon and B. F. Buxton. Genetic programming for improved receiver operating characteristics. In Josef Kittler and Fabio Roli, editors, *Second International Conference on Multiple Classifier System*, volume 2096 of *LNCS*, pages 68–77, Cambridge, 2-4 July 2001. Springer Verlag.

Langdon and Poli, 2001. W. B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer, 2001.

Langdon *et al.*, 1999. W. B. Langdon, Terry Soule, Riccardo Poli, and James A. Foster. The evolution of size and shape. In Lee Spector, W. B. Langdon, Una-May O'Reilly, and Peter J. Angeline, editors, *Advances in Genetic Programming 3*, chapter 8, pages 163–190. MIT Press, 1999.

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

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

Luke and Panait, 2001. Sean Luke and Liviu Panait. A survey and comparison of tree generation algorithms. In Lee Spector, Erik D. Goodman, Annie Wu, W. B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 81–88, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

Scott *et al.*, 1998. 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, *Proceedings of the 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, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 916–922, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

Yusoff *et al.*, 1998. Y. Yusoff, J. Kittler, and W. Christmas. 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, Italy, 7-11 June 1998.