Applications of Evolutionary Computing, Günther R. Raidl *et. al.* editors, Essex, UK
14–16 April 2003, LNCS 2611, pp87–98, Springer-Verlag

Comparison of AdaBoost and Genetic Programming for combining Neural Networks for Drug Discovery

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

[1] Data Exploration Sciences, GlaxoSmithKline, Research and Development, Greenford, Middlesex, UK
[2] Computer Science, University College, Gower Street, London, WC1E 6BT, UK
http://www.cs.ucl.ac.uk/staff/W.Langdon, /staff/B.Buxton

Abstract. Genetic programming (GP) based data fusion and AdaBoost can both improve *in vitro* prediction of Cytochrome P450 activity by combining artificial neural networks (ANN). Pharmaceutical drug design data provided by high throughput screening (HTS) is used to train many base ANN classifiers. In data mining (KDD) we must avoid over fitting. The ensembles do extrapolate from the training data to other unseen molecules. I.e. they predict inhibition of a P450 enzyme by compounds unlike the chemicals used to train them. Thus the models might provide *in silico* screens of virtual chemicals as well as physical ones from Glaxo SmithKline (GSK)'s cheminformatics database. The receiver operating characteristics (ROC) of boosted and evolved ensemble are given.

1 Introduction

Pharmaceuticals discovery research has evolved to the point of critical dependence upon computerised systems, databases and newer disciplines related to biological and chemical information processing and analysis. For instance, bioinformatics has enabled the discovery and characterisation of many more potential disease-related biological targets for screening, whilst cheminformatics concerns the capture and processing of chemical and biological information required to manage and optimise the overall screening process and to support decision-making for chemical lead selection. Machine learning can contribute to discovery processes in a variety of ways:

1. High-Throughput Screening (HTS), and increasingly Ultra-HTS, are resource-intensive. Models developed from diverse sets of previously-screened molecules may lead, via *in silico* screening, to smaller (or prioritised) screening-sets, targeted to specific biological activities.
2. Better ways to choose which of the vast numbers of "virtual molecules" should be synthesis and included in new chemical libraries.
3. After activity has been confirmed in primary screening, there are many additional tests of key properties which are required before forwarding compounds towards development. Traditionally, due to expense or the inability to screen the initialy vast numbers of molecules available, these have

been conducted on an *as needed* basis. This is expensive. Time and effort could be saved by computer based screening out of unsuitable chemicals earlier. Machine learning could produce predictions for other key properties such as target-selectivity, toxicity, tissue-permeability, solubility and drug metabolism (i.e. by P450 enzymes).

Here and in some other domains machine learning techniques based on a single paradigm have not been sufficient and so researchers have investigated mechanisms for combining them [Kittler and Roli, 2001; Gunatilaka and Baertlein, 2001]. Existing classifier fusion techniques, such as committees of experts [Jacobs *et al.*, 1991], bagging [Breiman, 1996] and boosting [Freund and Schapire, 1996], typically combine experts of the same type using a fixed way of combining their predictions. E.g. all the experts might be feed forward neural networks whose outputs are: simply summed, a weighted sum might be calculated, or a majority vote taken, to give the collective view of the classifier ensemble. That is, the fusion technique optimises the individual experts (e.g. using back propagation) while keeping the combination rule fixed. Genetic programming offers an alternative, which is to pretrain the experts and optimise the non-linear combination rule. Binary GAs have been used to find good committees of experts [Opitz and Shavlik, 1996; Kupinski and Anastasio, 1999; Kupinski *et al.*, 2000]. However genetic programming gives us the ability not only of deciding which experts to use in the ensemble but also how their predictions are to be combined. I.e. to simultaneously solve both the feature selection problem (at the individual expert level of granularity) and the combination rule. Because the individual experts are pretrained the GP does not need to know how they were trained and so has the ability to form superior ensembles of heterogeneous classifiers [Langdon and Buxton, 2001b; Langdon *et al.*, 2002].

Intelligent classification techniques, such as artificial neural networks (ANN), have had some success at predicting potential drug activity and we have shown genetic programming is able to fuse different neural networks to obtain better predictions [Langdon *et al.*, 2001]. We shall further demonstrate our system and also compare results with a popular boosting technique, AdaBoost [Schwenk and Bengio, 2000].

2 Receiver Operating Characteristics

The Receiver Operating Characteristics (ROC) of a classifier provide a helpful way of illustrating the trade off it makes between catching positive examples and raising false alarms [Swets *et al.*, 2000]. Figures 4 and 5 show ROC curves.

[Scott *et al.*, 1998] suggest the "Maximum Realisable Receiver Operating Characteristics" for a combination of classifiers is the convex hull of their individual ROCs, cf. also [Provost and Fawcett, 2001]. ("Lotteries" in game theory [Binmore, 1990] are somewhat similar.) However we have already shown GP can in some cases do better, including on Scott's own benchmarks [Langdon and Buxton, 2001a] and real world pharmaceutical classification tasks [Langdon *et al.*, 2001].

3 The Pharmaceutical Data

The pharmaceutical data are similar to [Langdon *et al.*, 2001] and [Langdon *et al.*, 2002]. Thousands of chemicals from a chemical library have been tested (using 2 triplicated HTS runs) to see if they inhibit one of the P450 enzymes involved in metabolism. This is an important screen in early drug discovery since P450 inhibition could be expected to lead to problems (when a compound is first evaluated in humans).

The chemicals are a very diverse set, covering the most common types of drug or drug-like compounds, such as would be found in the big pharmaceutical company compound banks. Hence they probably have a range of inhibition mechanisms. Some "primary" enzyme inhibition mechanisms are likely to be much more frequent within the tested set of compounds than others. This is precisely the kind of situation which can defeat individual classifiers.

Chemicals which gave inconsistent screening results (i.e. more than 15% variation between readings) were discarded. The mean of the 6 measurements taken was compared against an activity threshold. Those below the threshold are said to be inactive, while chemicals whose mean exceeded the threshold were classified as inhibitory, i.e. active against the P450 target.

These noise free chemicals were then hierarchically clustered using Ward's linkage in combination with Tanimoto similarity, computed from Daylight 2 Kbit string chemical fingerprint data[1]. Clusters were defined at 0.8 tan. Note unlike our previous work, active and inactive were not separated prior to clustering. This leads to three types of cluster. 1) Pure clusters, i.e. clusters containing either all active or all inactive compounds. 2) Impure, or mixed, clusters. 3) Singleton clusters, i.e. clusters consisting of a single chemical compound.

A total of 699, numerical and categorical, chemical features from a diverse array of families (electronic, structural, topological/shape, physico-chemical, etc.) were computed for each chemical, starting from a SMILES[2] representation of it's primary chemical structure (2-d chemical formula).

The chemicals selected for screening are naturally a highly biased sample. It consists of those chemicals which were considered interesting and necessarily were available. There are two things we would like to know about any predictive classifier; how well it will work on chemicals like those on which it was trained and secondly (and much more difficult), how well will it extrapolate outside the training domain. Naturally we know the distribution of the first but, we do not know the distribution of the second.

The classifiers are trained on chemicals selected from the clean data (i.e. compounds with little HTS noise) at random, ensuring the training data has the same proportion of inhibitory and inactive chemicals and the same proportions of pure, mixed and singleton clusters, as the the whole dataset. The generalisation performance of classifiers is estimated by measuring how well it predicts unseen chemicals, drawn at random from the same distribution.

[1] `http://www.daylight.com/dayhtml/doc/theory/theory.finger.html`

[2] `http://www.daylight.com/dayhtml/smiles/`

We keep back a number of the chemicals from the singleton clusters. Since these are likely to be the most different from the training set. We use the classifiers' performance on this "extrapolation set" to estimate how well they will perform on novel chemicals.

4 Training the Neural Networks

As before, the 699 features were divided into 15 functionally related groups of about 50 each. Clementine was used to train 5 feed forward neural networks on each of the 15 groups of features. Clementine default parameters (including a 50/50 training set/stop set split, to enable early stopping in order to limit over fitting) were used. (Following disappointing performance (due to over fitting) with C4.5 decision trees we decided to only to use neural networks.)

An imbalance between positives and negatives is common in many data mining tasks. However many machine learning techniques work best when trained on "balanced data sets", i.e. data sets containing an equal mix of inhibitory and inactive examples. [Chawla *et al.*, 2002]. The 1299 compounds were used to create five data sets. Each contained the same 279 inhibitory chemicals and approximately 200 different inactive chemicals. That is, each data set was approximately balanced. Each neural network was trained on one of the 15 groups of attributes selected from one of the five balanced data sets. Making a total of 75 weak classifiers.

5 Genetic Programming Configuration

5.1 Function and Terminal Sets

The genetic programming data fusion system is deliberately identical (except for the choice of random constants) to that described in [Langdon and Buxton, 2001c], cf. Table 1.

In order to use the neural networks within GP they are presented to GP as 75 problem specific functions. Each returns the classification and associated confidence of the corresponding neural network for the current chemical. The Clementine neural networks yield a single floating point number (between 0 and 1). Values below 0.5 are treated as class 0, while those above 0.5 are regarded as predicting class 1. We treat the magnitude of the difference from 0.5 as the indicating the network's confidence.

Normally the output of a neural network is converted into a binary classification (i.e. the chemical is inhibitory 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. However by replacing the fixed value of 0.5 by a tunable threshold ($0 \ldots 1$) we can vary this trade off, so that we get a complete curve in the ROC square. By making the threshold the argument of the function we leave the choice of suitable operating point to the GP. These arguments are treated like any other by the GP and so can be any valid arithmetic operation, including the base classifiers

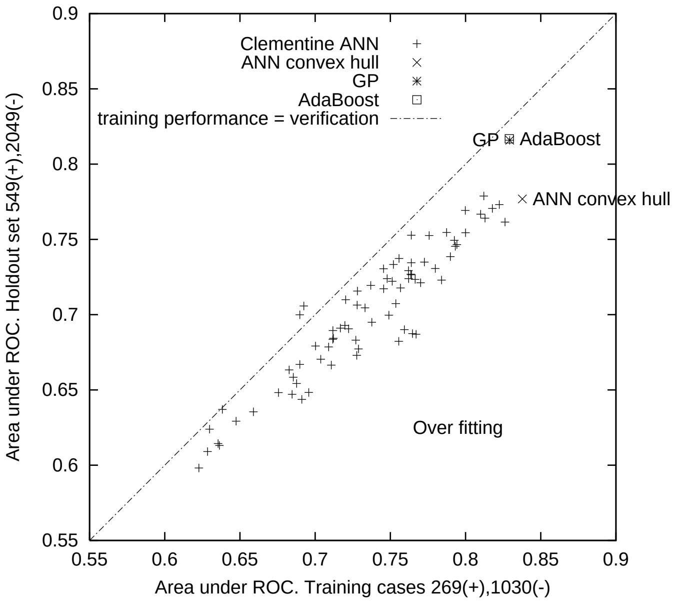

Fig. 1. Area under ROC curve of P450 Clementine neural networks. Points below the diagonal indicate possible over training. Their convex hull is indicated, as are the corresponding points for the boosted and evolved classifiers (which lie almost on top of each other).

themselves. Note GP simultaneously adapts how the non-linear combination of the base classifiers and their operating points.

The terminals or leaves of the trees being evolved by the GP are either constants or the adjustable threshold "T" (see Table 1).

5.2 GP Representation

We continue to create each individual in the initial population with five random trees [Jacobs *et al.*, 1991; Soule, 1999; Langdon, 1998]. Each tree within an individual returns a signed real number. The classification of the individual is the sum of the answers given by the five trees. Note the GP can combine the supplied classifiers in an almost totally arbitrary, non-linear way.

Following [Angeline, 1998] and others, we use a high mutation rate and a mixture of different mutation operators. To avoid bloat, we also use size fair crossover [Langdon, 2000], see Table 1.

5.3 GP Fitness Function

Unlike in previous work and in an effort to reduce over fitting, the chemicals used to train the base level classifiers (ANN) were not used to train the GP. Instead approximately the same number (1300) of chemicals, drawn from the

Table 1. GP P450 Data Fusion Parameters

Objective:	Evolve a combination of neural networks with maximum ROC convex hull area on P450 inhibition prediction
Function set:	INT FRAC Max Min MaxA MinA MUL ADD DIV SUB IFLTE 75 ANN trained on P450 data
Terminal set:	T 0 0.5 1 plus 100 unique random constants -1..1
Fitness:	Area under convex hull of 11 ROC points (plus 0,0 and 1,1)
Selection:	generational (non elitist), tournament size 7
Wrapper:	$\geq 0 \Rightarrow$ inhibitory, inactive otherwise
Pop Size:	500
No size or depth limits	
Initial pop:	Each individual comprises five trees each created by ramped half-and-half (5:8) (half terminals are constants, each initial tree limited to 300)
Parameters:	50% size fair crossover, crossover fragments $\leq$ 30 [Langdon, 2000] 50% mutation (point 22.5%, constants 22.5%, shrink 2.5% subtree 2.5%)
Termination:	generation 10

same distribution were used. I.e. a total of 2599 chemicals were used in training. Almost twice as many (1500) as in [Langdon *et al.*, 2002].

Fitness of each individual is calculated on the training set. The adjustable threshold "T" is set to values 0.1 apart, starting at 0 and increasing to 1. For each setting, the evolved program is used to predict the activity of each chemical in the training set and true positive (TP) and false positive (FP) rates are calculated. Each TP,FP pair gives a point on an ROC curve. The fitness of the classifier is the area under the convex hull of these (plus the fixed points 0,0 and 1,1).

6 Forming a Composite using AdaBoost

Boosting [Freund and Schapire, 1996] is a deterministic algorithm whereby a series of weak (i.e. poorly performing) but different classifiers are trained. A single composite classifier is formed by the weighted sum of the individual classifiers. The composite should be better than the initial weak classifier.

At each boosting round, a new classifier is trained on weighted training data. At each round the classifier produced is different because at each round the weight associated with each training example is adjusted. The weights of examples on which the last trained classifier did badly are increased. The idea is to encourage the next classifier to be trained to pay more attention to hard examples. Cf. "Dynamic Subset Selection" [Gathercole and Ross, 1994]. As well as being used to adjust the weights of training examples, the performance of each weak classifier is used to determine the strength of its contribution to the final composite classifier.

Note AdaBoost uses the accuracy of the classifier, rather than its ROC, and so implicitly assumes false positives and false negatives have equal costs.

6.1 Matlab Neural Networks Ensembles

The Matlab neural network tool box was used to train single hidden layer, fully connected feed forward networks. Each input unit is connected to one of the networks trained by Clementine (75). The output layer consists of two output units (one for each class). In preliminary experiments, the number of units in the hidden layer was set to 20 and then progressively reduced. Little performance variation was seen until it was reduced to a single node. In the hope of avoiding over fitting and reducing run time the hidden layer was minimised to just two units.

Approximately half the training data was used by Matlab as a stopping set. The training and stop sets do not contain the same chemicals. Matlab used between 7 and 55 training epochs (see Fig. 2).

The default parameters provided by Matlab were used (e.g. back propagation, minimise the sum of square of the difference between output units and the actual class and momentum constant of 0.95). Due to randomised initialisation training a network on the same data will not necessarily produce the same final network each time.

6.2 Boosting Matlab Neural Networks

The first neural network is trained using all the training data (half is used for the stopping set) and initialy each training chemical has the same weight. After the first network has been successfully trained the weights are adjusted, using the AdaBoost algorithm [Schwenk and Bengio, 2000]. At each subsequent iteration, a new training set is created by randomly sampling from training examples in proportion to their weights. The training set used by Matlab is the same size on each boosting cycle. Each time the training set is split approximately equally into actual training and stopping sets.

[Schwenk and Bengio, 2000] suggests boosting should stop once the error exceeds 50%. However, in this application, low performing networks were produced very early in the boosting process, even in the first round. We discard such networks, create a new random sample of training chemicals using the current distribution of weights and then train again. Boosting is stopped after training 20 networks (successful or otherwise). The composite is based only on the successful networks. Approximately 10% of trained networks have to be ignored.

AdaBoost specifies that the class with the highest (weighted) vote of all the weak classifiers will be taken as the predicted class. To produce a complete ROC curve, we use the normalised difference between the weighted votes for each output neuron (i.e. class) as the ensemble's confidence.

7 Results

Figure 3 plots the evolution of fitness (on the GP training set). For the best in the population, the area under ROC on the verification set was also measured. In

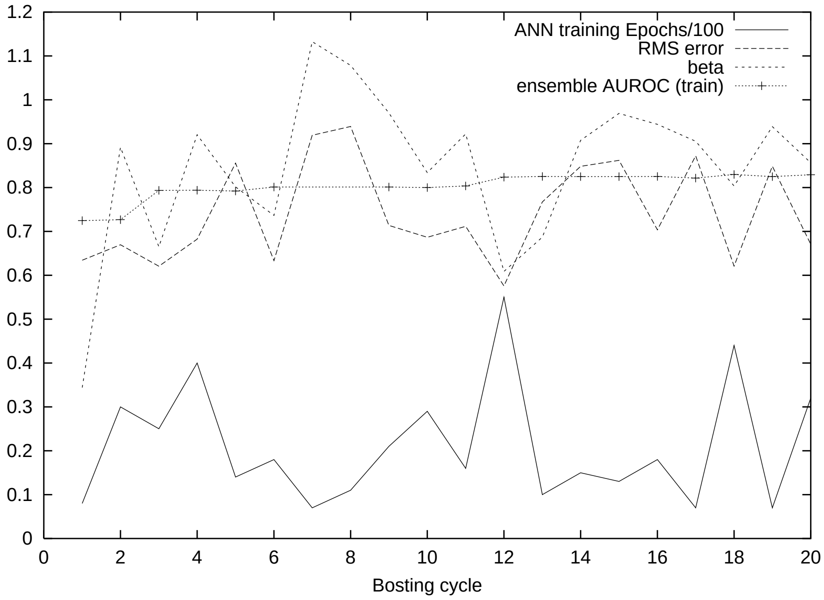

Fig. 2. Performance of individual Matlab neural networks (RMS training error and the AdaBoost weighted pseudoloss function, beta) and of complete AdaBoost ensemble AUROC during boosting. Note beta = error/(1-error) does not converge to zero.

contrast to previous work, the gap in performance on the training and verification sets is modest and grows only slowly. There are two causes for this.

Firstly GP is stopped after generation 10. Earlier work indicated that on problems of this type most of the useful learning was accomplished at the start of the run and most of the apparent performance gain achieved later was illusory, caused by over fitting. (Stopping early also has the advantage of shortening run time to 27 minutes.)

The second potential cause for reduced over fitting, is that the GP is now trained on data that the lower level classifiers have not seen. This appears to have had the beneficial affect of countering any lack of generality which has slipped into the neural networks as they were being trained (see Fig. 1). However the evidence for this is not overwhelming. Certainly in Fig. 3 we see strong correlation between the performance of different GP individuals on the different datasets and so we could reach the conclusion that some parts of the P450 HTS data are simply harder than others.

Figure 4 shows the classifiers produced by AdaBoost and genetic programming have essentially the same performance. (Earlier experiments indicated a small, but significant advantage for a classifier evolved by GP.) However they are significantly better than, not only each classifier in their function set, but also the convex hull of these base classifiers

The use of size fair crossover [Langdon, 2000] and mutation means we do not see explosive increase in program size (bloat [Langdon *et al.*, 1999]) and

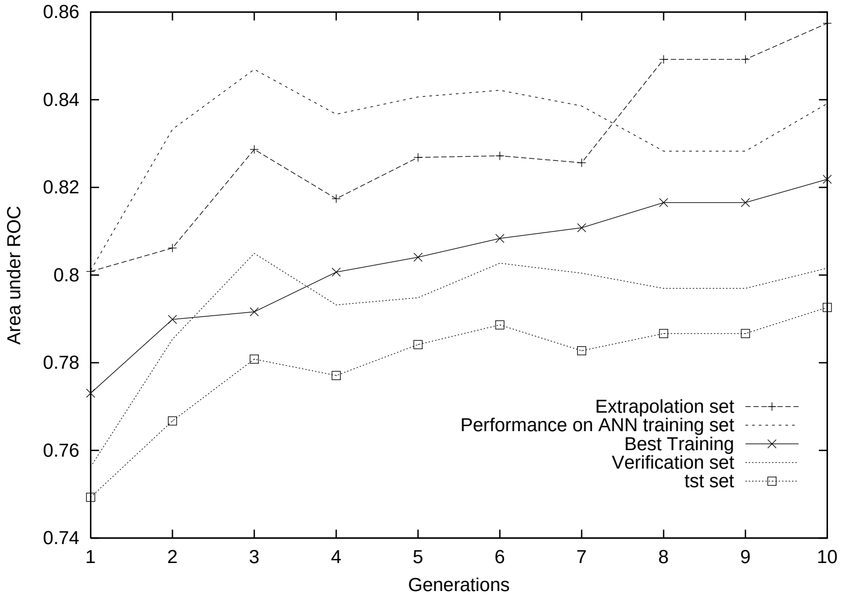

Fig. 3. Evolution of performance of best of generation evolved classifier on other datasets.

preliminary experiments suggest over fitting is more closely related to number of generations over which the population has been exposed to the same environment than to the size of the programs.

While earlier experiments using different training data for the base and evolved classifiers were not encouraging, the results for both GP and boosting indicate that using separate chemicals to train base level classifiers and composites of them can indeed be successful.

The performance of the composite classifiers on an "extrapolation" set (cf. Fig. 5) is good. This is encouraging, since we really wish to make predictions for untested chemicals.

8 Conclusions

AdaBoost [Schwenk and Bengio, 2000], like genetic programming (GP), can be used to form non-linear combinations of lower level classifiers. Again like, GP, these exceed Scott's "Maximum Realisable Receiver Operating Characteristics" [Scott *et al.*, 1998] on a non-trivial industrial problem (cf. Figs. 1 and 4).

It is especially encouraging that both methods of automatically forming classifier ensembles are able to extrapolate away from their training data and make predictions on new chemicals, cf. Fig. 5.

Acknowledgements

We would like to thank Sunny Hung, George Seibel, David Corney and Matthew Trotter.

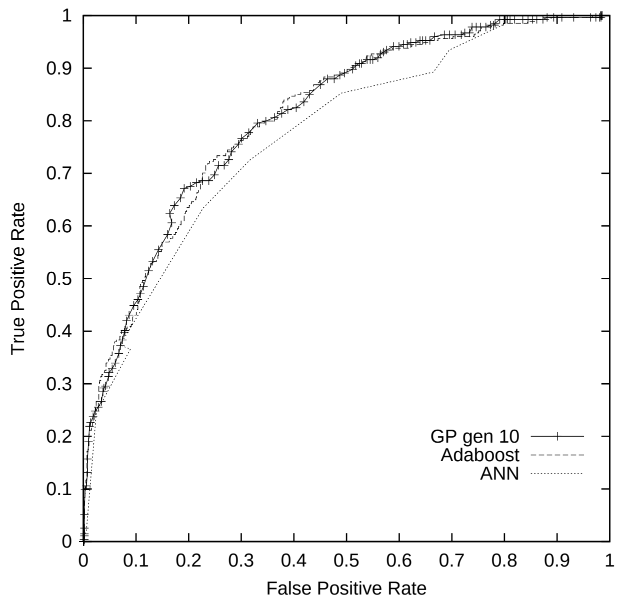

Fig. 4. Receiver Operating Characteristics of evolved and boosted composite classifiers (1298 verification chemicals). In both cases the classifier lies outside the convex hull of their base classifiers (lines without crosses). Note the convex hull classifiers are determined on the training set and so need no longer be convex when used to classify the holdout data.

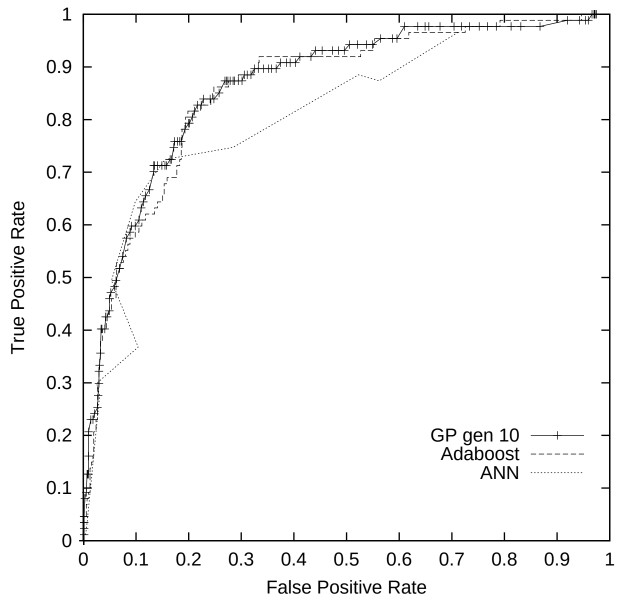

Fig. 5. Receiver Operating Characteristics of evolved and boosted composite classifiers on extrapolation set (779 singleton chemicals).

Source Code

C++ and AdaBoost.M2 Matlab source code can be obtained from `ftp://cs.ucl.ac.uk/genetic/gp-code/` and `ftp://cs.ucl.ac.uk/genetic/boosting/` respectively.

References

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

Binmore, 1990. Ken Binmore. *Fun and Games*. D. C. Heath, Lexington, MA, USA, 1990.

Breiman, 1996. Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

Chawla *et al.*, 2002. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

Freund and Schapire, 1996. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the thirteenth International Conference*, pages 148–156. Morgan Kaufmann, 1996.

Gathercole and Ross, 1994. Chris Gathercole and Peter Ross. Dynamic training subset selection for supervised learning in genetic programming. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature III*, volume 866 of *LNCS*, pages 312–321, Jerusalem, 9-14 October 1994. Springer-Verlag.

Gunatilaka and Baertlein, 2001. Ajith H. Gunatilaka and Brian A. Baertlein. Feature-level and decision level fusion of noncoincidently sampled sensors for land mine detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):577–589, June 2001.

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.

Jones, 1998. Gareth Jones. Genetic and evolutionary algorithms. In Paul von Rague, editor, *Encyclopedia of Computational Chemistry*. John Wiley and Sons, 1998.

Kittler and Roli, 2001. Josef Kittler and Fabio Roli, editors. *Second International Conference on Multiple Classifier Systems*, volume 2096 of *LNCS*, Cambridge, 2-4 July 2001. Springer Verlag.

Kordon and Smits, 2001. Arthur K. Kordon and Guido F. Smits. Soft sensor development using genetic programming. In Lee Spector *et al.*, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1346–1351, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

Kupinski and Anastasio, 1999. M. A. Kupinski and M. A. Anastasio. Multiobjective genetic optimization of diagnostic classifiers with implications for generating receiver operating characteristic curves. *IEEE Transactions on Medical Imaging*, 18(8):675–685, Aug 1999.

Kupinski *et al.*, 2000. Matthew A. Kupinski, Mark A. Anastasio, and Maryellem L. Giger. Multiobjective genetic optimization of diagnostic classifiers used in the computerized detection of mass lesions in mammography. In Kenneth M. Hanson, editor, *SPIE Medical Imaging Conference*, volume 3979, San Diego, California, 2000.

Langdon and Buxton, 2001a. W. B. Langdon and B. F. Buxton. Genetic programming for combining classifiers. In Lee Spector *et al.*, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 66–73, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.

Langdon and Buxton, 2001b. 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 Buxton, 2001c. William B. Langdon and Bernard F. Buxton. Evolving receiver operating characteristics for data fusion. In Julian F. Miller *et al.*, editors, *Genetic Programming, Proceedings of EuroGP'2001*, volume 2038 of *LNCS*, pages 87–96, Lake Como, Italy, 18-20 April 2001. Springer-Verlag.

Langdon *et al.*, 1999. William B. Langdon, Terry Soule, Riccardo Poli, and James A. Foster. The evolution of size and shape. In Lee Spector, William 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 *et al.*, 2001. W. B. Langdon, S. J. Barrett, and B. F. Buxton. Genetic programming for combining neural networks for drug discovery. In Rajkumar Roy *et al.*, editors, *Soft Computing and Industry Recent Applications*, pages 597–608. Springer-Verlag, 10–24 September 2001. Published 2002.

Langdon *et al.*, 2002. William B. Langdon, S. J. Barrett, and B. F. Buxton. Combining decision trees and neural networks for drug discovery. In James A. Foster *et al.*, editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 60–70, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.

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

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

Opitz and Shavlik, 1996. David W. Opitz and Jude W. Shavlik. Actively searching for an effective neural-network ensemble. *Connection Science*, 8(3-4):337–353, 1996.

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

Schwenk and Bengio, 2000. Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural Computation*, 12(8):1869–1887, 2000.

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 *et al.*, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 916–922, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

Swets *et al.*, 2000. John A. Swets, Robyn M. Dawes, and John Monahan. Better decisions through science. *Scientific American*, 283(4):70–75, October 2000.

Turney, 1995. Peter D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995.