EuroGP'2002 James A. Foster, Evelyne Lutton, Julian Miller, Conor Ryan and Andrea G. B. Tettamanzi *editors*, Kinsale, Ireland, 3–5 April, LNCS 2278, pp. 60-70, Springer

Combining Decision Trees and Neural Networks for Drug Discovery

William 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. Genetic programming (GP) offers a generic method of automatically fusing together classifiers using their receiver operating characteristics (ROC) to yield superior ensembles. We combine decision trees (C4.5) and artificial neural networks (ANN) on a difficult pharmaceutical data mining (KDD) drug discovery application. Specifically predicting inhibition of a P450 enzyme. Training data came from high throughput screening (HTS) runs. The evolved model may be used to predict behaviour of virtual (i.e. yet to be manufactured) chemicals. Measures to reduce over fitting are also described.

1 Introduction

Computers are very good at collecting and storing huge volumes of data (such as in data warehouses) but they have been less successful at extracting useful information from it. Machine Learning techniques have been used to extract or discover knowledge (KDD). However the exponential explosion of possible patterns defeats simple searches and so there is increasing interest in using heuristic methods, such as evolutionary computation, in data mining [Freitas, 1999]. Additionally in many cases Machine Learning techniques based on a single paradigm have not been sufficient and so people have investigated mechanisms for combining them [Kittler and Roli, 2001; Gunatilaka and Baertlein, 2001]. There are many possible interpretations of data fusion [Kelly, 1999], however 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. An interesting alternative is to pretrain the experts and optimise the combination rule. With a small number of experts [Sirlantzis *et al.*, 2001] and a simple voting rule it might be feasible to try all possible combinations of experts. However there are 2^n (where n = number of experts) possible combinations in such a voting scheme. 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. 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].

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. Note that although we use training data given by high throughput screening (HTS) tests of real chemicals, the classifiers we evolve are able to classify not only existing chemicals but also to generalise to related areas of chemical space in particular (we hope) to virtual chemicals. I.e. chemicals that do not yet exist but which could be manufactured if predictions indicate they might be useful drugs.

Intelligent classification techniques, such as artificial neural networks (ANN), have had limited success at predicting potential drug activity. However we have shown genetic programming is able to fuse different neural networks to obtain better predictions [Langdon *et al.*, 2001]. We shall show our system can also be used with C4.5 decision trees and indeed can combine C4.5 and ANN on this pharmaceutical classification task, predicting inhibition of a P450 enzyme. (GP achieves ensembles with the same performance as the with the ANN but using poorer initial classifiers.)

The system and problem have already been described in [Langdon and Buxton, 2001c] and [Langdon *et al.*, 2001] so only summaries of Receiver Operating Characteristics (ROC) and the application are given in Sects. 2 and 3. Section 4 describes how the base classifiers were trained, while Sect. 5 summarises the evolutionary system. The results (Sect. 6) are followed by a discussion, including over fitting, (Sect. 7) and conclusions (Sect. 8).

2 Receiver Operating Characteristics

Any classifier makes a trade off between catching positive examples and raising false alarms. Where the costs of these are not known, difficult to determine or subject to change, 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 [Swets *et al.*, 2000].

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. 3). Many classifiers have an adjustable threshold parameter. This allows the user to trade off TP (sensitivity) against FP (1 - specificity). By varying the threshold 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.

[Scott *et al.*, 1998] suggests 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 this pharmaceutical classification task (fusing ANN) [Langdon *et al.*, 2001].

3 The Pharmaceutical Data

Details of the preparation of pharmaceutical data are given in [Langdon *et al.*, 2001]. Briefly many thousands of chemicals have been tested 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 an adverse drug reaction were any of those molecules to make it to the clinical drug development stage (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 were discarded. The remainder were separated into "active" (inhibitory) and "inactive". These were separately clustered (based on primary chemical structure) into 445 active clusters and 1811 inactive clusters. The chemical at the centre of each of these 2256 clusters was chosen to represent its cluster. Using a collection of in-house and publicly available software, a total of 699 numeric chemical features were computed for each centroid molecule. 1500 compounds (300 inhibitory, 1200 inactives) were selected for use as the training set, whilst the remaining 756 were retained as a separate "holdout" set. The 699 features were divided into 15 groups of about 50 each.

4 Training the Neural Networks and Decision Trees

Details how Clementine was used to train 4 feed forward neural networks on each of the 15 groups of features were given in [Langdon *et al.*, 2001]. The training of the C4.5 decision trees was deliberately kept similar.

An imbalance between positive 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

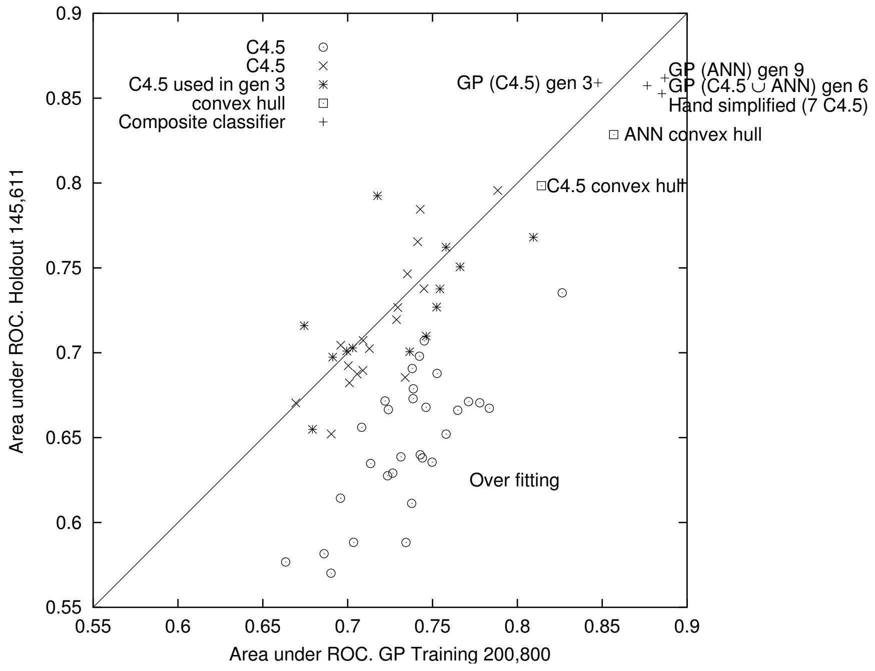

Fig. 1. Area under ROC (AUROC) curve of P450 decision trees (C4.5). Points below the diagonal indicate possible over training. 30 decision trees ⊙ which do not generalise were not used by GP. The convex hulls of the remaining 30 and the 60 neural networks are plotted as □. The AUROC of evolved classifiers is plotted with +.

inactive examples. The 1500 compounds were used to create four data sets. Each contained the same 300 inhibitory chemicals and 300 different inactive chemicals. That is, each data set was balanced. As with the ANN, each decision tree was trained on one of the 15 groups of attributes selected from one of the four balanced data sets. Making a total of 60 classifiers.

The C4.5 decision trees were generated by Clementine (5.01) using "Build rule". Following problems with over fitting we used the following expert options: windowing (5% increment 1%), persistence 10, stop when reach accuracy of 70% and pruning (min unpruned 40%, tightness of fit 10%). Unfortunately many of the models still had poor generalisation (Clementine 6.0 is improved in this respect). The performance of each decision tree was measured on both the training and holdout sets, see Fig. 1. Those which performed significantly worse on the hold out data were not used by the GP (one sided test [Hanley and McNeil, 1983] p=0.1 and r=0). This left 30 decision trees. These were made available to genetic programming as functions.

5 Genetic Programming Configuration

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.

5.1 Function and Terminal Sets

In order to use the decision trees (and neural networks) within GP they were exported from Clementine as C code and packaged up and presented to GP as 30 (60) problem specific functions. The GP is run separately from Clementine using 30, 60 or 90 files (depending if fusing C4.5 trees, ANN or both). There is one file for each decision tree or ANN. Each contains the classification given by the corresponding decision tree (or ANN) for the current chemical. Clementine decision trees not only return a predicted class but also their confidence. Prior to running the GP, the class and confidence were combined into a single floating point value between zero and one. For both classes, if the decision tree has zero confidence a value of 0.5 is used. If the chemical is predicted to have an inhibitory effect; as the confidence increases to 1.0 the value increases linearly to 1.0 (and decreases to 0 if inactive). This becomes the value returned by the function inside the GP system (with a neutral adjustable threshold bias).

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. I.e. one trade off between catching all positives but raising too many false alarms. However, for neural networks, decision trees or other classifiers, 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 achieved by replacing the fixed value of 0.5 by a tunable threshold. By continuously varying the threshold from 0 to 1, the output of any of the classifiers is 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 inhibitory). In fact we leave the choice of suitable operating point to the GP, by making it the argument of the function. These arguments are treated like any other by the GP and so can be any valid arithmetic operation, including the base classifiers themselves.

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 Representation and Genetic Operators

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 real 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 is like

Table 1. GP P450 Data Fusion Parameters

Objective:	Evolve a combination of decision trees and/or 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 30 C4.5 (60 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 [Langdon, 2000], crossover fragments $\leq$ 30 50% mutation (point 22.5%, constants 22.5%, shrink 2.5% subtree 2.5%)
Termination:	generation 50

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.

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 Training Data and Fitness Function

The 1500 examples used to train the decision trees were randomly split into 1000 to be used to train the GP and 500 (containing 100 inhibitory chemicals) kept back as a verification set. NB performance was finally evaluated on the 756 compounds which had not been used either by the GP or by Clementine.

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. These predictions are compared with measured activity. The proportions of inhibitory chemicals correctly predicted (TP) and the proportion of inactive ones incorrectly predicted (FP) 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 Results

Figure 2 plots (for populations using 30 C4.5 decision trees, 60 ANN and both) the evolution of fitness of the best individual (on the training set) in the population. For the best in the population, the area under ROC on the verification set was also measured (lines with crosses in Fig. 2). The gap in performance on the training and verification sets is large and grows. This indicates GP is responsible for some over training. Analysis of these and other runs suggests the degree of over fitting is not well correlated to program size but instead length of training seems to be a better indicator. Accordingly, when we chose individual programs to represent the whole run, we took the best of each population from near the beginning of the run where the performance on the verification set was high. (These are shown with vertical lines in Fig. 2 at generations 3, 6 and 9.) Only then was their performance assessed on the holdout set. In the runs where the population contained ANN there was a marked drop in performance predicted by the verification set, indicating the ANN were themselves responsible for some over fitting. The drop in the C4.5 only population was smaller.

The performance of these three programs are not significantly different (plotted in Fig. 1 with +). However they are significantly better than, not only each classifier in their function set, but also the convex hull of these base classifiers (two □ in Fig. 1). An evolved program was simplified by hand to yield a simple addition rule of similar performance (also plotted in Fig. 1 with +). Even if we had been prepared to restrict the ensemble to this particular type of combination rule, the search problem is still far from trivial (2^{30}, 2^{60} and 2^{90}).

Figure 3 shows the ROC of the evolved classifiers, measured on the holdout set. Figure 3 also shows, for comparison, the ROC of the classifier produced by taking the convex hull of the 30 C4.5 decision trees and 60 ANNs (generated on the training data). Of course it is convex on the training data, but need not be on the hold out data. Note the convex hull of the ANN contains that of the C4.5, so it is also the convex hull of the ANN and C4.5 together.

7 Over Fitting

Over fitting is to some extent endemic in Machine Learning and it is no surprise to see it in GP. In fact it goes further than that. There is a case that natural evolution itself tends to over fit. When Darwin says finches have adapted to a particular type of food, an alternative view is they have become over fitted to their current environmental niche (i.e. their training data). Taken out of the niche and exposed to a new environment they may fare less well. Alternatively if the niche itself changes they may have to re-adapt or become extinct.

GP's environment is the fitness cases. Where these are small and static we must fear GP will over fit. If large volumes of data are available for training then it should all be used. Sampling [Gathercole and Ross, 1997; Gathercole, 1998; Teller and Andre, 1997] and/or caching [Handley, 1994; Langdon, 1998] techniques can be used to reduce run time.

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 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. This supports [Schmiedle *et al.*, 2001]'s

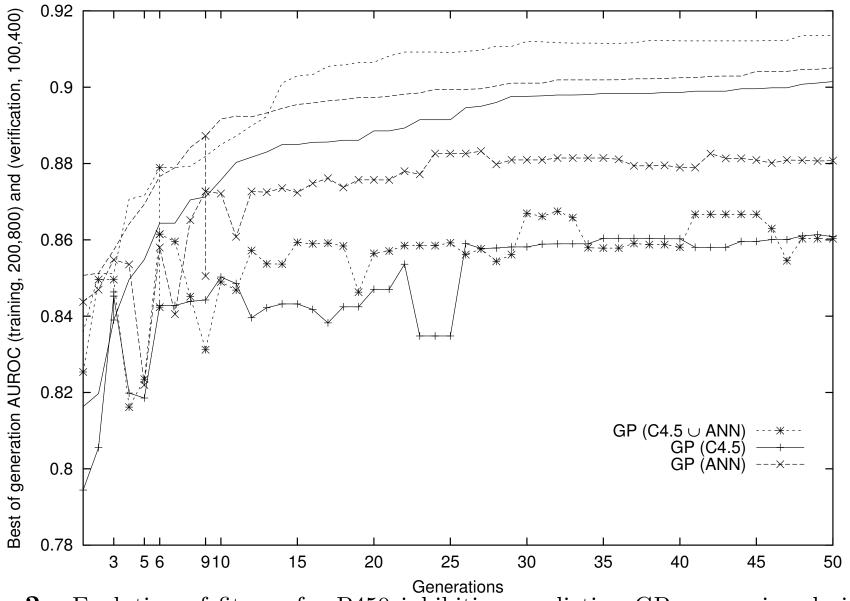

Fig. 2. Evolution of fitness for P450 inhibition prediction GP runs using decision trees, neural networks and both. Plot shows performance on training set (no crosses) and verification set (crosses) for best of each generation. Vertical lines show generation (3, 6 and 9) selected as output of each run (lower point is performance on holdout set).

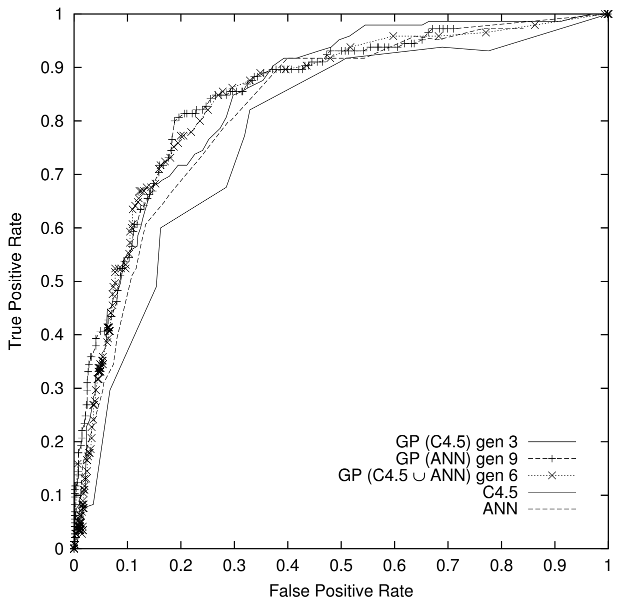

Fig. 3. Receiver Operating Characteristics of evolved composite classifiers (holdout data). In all 3 cases the evolved classifier lies outside the the convex hull of their base classifiers (lines without crosses). Note the convex hull classifiers are determined on the training set and so need no longer be convex when used to classify the holdout data. Also note the hull of the decision trees lies almost entirely within than of the ANN.

suggestion for pragmatic limits on run time, as an alternative to parsimony pressures (i.e. fitness rewards for smaller programs).

In traditional neural networks etc., over fitting may tackled by "regularization" parameters, which bias the learning system to produce simple or smooth functions. [Davidson *et al.*, 2000] shows regularization can be incorporated into GP, provided we are prepared to restrict the nature of models we will allow to evolve or if we have some reason for preferring smooth or simple functions.

While [Sollich and Krogh, 1996] suggests it might be good to allow individual base classifiers to over fit we have seen little to support this. In preliminary experiments, over fit base classifiers dragged the population in the same direction as themselves, leading to the evolution of similarly over fit ensembles. This may be due to using the same data to train both the base classifiers and the GP, but initial experiments using different training data for the base and evolved classifiers were not encouraging (possibly due to insufficient training data).

8 Conclusions

In [Langdon and Buxton, 2001a] we showed, using [Scott *et al.*, 1998]'s own bench marks, that genetic programming can do better than the receiver operating characteristics (ROC) convex hull both in theory and practice. 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. 1 and 3) that GP can be used in a large classification application (related to drug discovery) to automatically create ensembles of decision trees, neural networks and indeed both. Even though GP starts with poorer classifiers (all the C4.5 ROCs lie within the convex hull of the previously used neural network classifiers), ensembles of similar performance have been automatically evolved. While GP allows arbitrary combination rules, it can also be used to aid finding simple rules.

References

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

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

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

Davidson *et al.*, 2000. J.W. Davidson, D.A. Savic, and G.A. Walters. Rainfall runoff modeling using a new polynomial regression method. In *Proc. 4th Int. Conf. on Hydroinformatics*, Iowa City.

Freitas, 1999. A.A. Freitas. Data mining with evolutionary algorithms: Research directions. Technical Report WS-99-06, AAAI, Orlando.

Freund and Schapire, 1996. Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th ICML*, pp148–156. Morgan Kaufmann.

Gathercole and Ross, 1997. C. Gathercole and P. Ross. Tackling the boolean even N parity problem with genetic programming and limited-error fitness. In J.R. Koza *et al.*, eds., *Proc. GP'97*, pp119–127, Stanford University. Morgan Kaufmann.

Gathercole, 1998. C. Gathercole. *An Investigation of Supervised Learning in Genetic Programming*. PhD thesis, University of Edinburgh, 1998.

Gunatilaka and Baertlein, 2001. A.H. Gunatilaka and B.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.

Handley, 1994. S. Handley. On the use of a directed acyclic graph to represent a population of computer programs. In *Proc. WCCI'94*, pp154–159, Orlando. IEEE.

Hanley and McNeil, 1983. J.A. Hanley and B.J. McNeil. A method of comparing the areas under ROC curves derived from the same cases. *Radiology*, 148:839–843.

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.

Kelly, 1999. G. Kelly. Data fusion: from primary metrology to process measurement. In V. Piuri and M. Savino, eds., *Proc. 16th Instrumentation and Measurement Technology Conference. IMTC/99.*, vol 3, pp1325–1329, Venice, Italy. IEEE.

Kittler and Roli, 2001. J. Kittler and F. Roli, eds.. *Second International Conference on Multiple Classifier Systems*, vol 2096 of *LNCS*, Cambridge. Springer Verlag.

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

Kupinski *et al.*, 2000. M.A. Kupinski, M.A. Anastasio, and M.L. Giger. Multiobjective genetic optimization of diagnostic classifiers used in the computerized detection of mass lesions in mammography. In K.M. Hanson, ed., *SPIE Medical Imaging Conference*, vol 3979, San Diego.

Langdon and Buxton, 2001a. W.B. Langdon and B.F. Buxton. W.B. Langdon and B.F. Buxton. Genetic programming for combining classifiers. In L. Spector *et al.*, eds., *GECCO-2001*, pp66–73, San Francisco. Morgan Kaufmann.

Langdon and Buxton, 2001b. W.B. Langdon and B.F. Buxton. Genetic programming for improved receiver operating characteristics. In J. Kittler and F. Roli, eds., *Second International Conference on Multiple Classifier System*, pp68–77.

Langdon and Buxton, 2001c. W.B. Langdon and B.F. Buxton. Evolving receiver operating characteristics for data fusion. In J.F. Miller *at al.*, eds., *EuroGP'2001*, vol 2038 of *LNCS*, pp87–96, Lake Como, Italy. Springer.

Langdon *et al.*, 1999. W.B. Langdon, T. Soule, R. Poli, and J.A. Foster. The evolution of size and shape. In L. Spector *at al.*, eds., *Advances in Genetic Programming 3*, ch 8, pp163–190. MIT Press.

Langdon *et al.*, 2001. W.B. Langdon, S.J. Barrett, and B.F. Buxton. Genetic programming for combining neural networks for drug discovery. In *WSC6, 6th World Conference on Soft Computing in Industrial Applications*, Springer-Verlag. Forthcoming.

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

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

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

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

Schmiedle *et al.*, 2001. F. Schmiedle, D. Grosse, R. Drechsler, and B. Becker. Too much knowledge hurts: Acceleration of genetic programs for learning heuristics. In B. Reusch, ed., *Computational Intelligence : Theory and Applications*, vol 2206 of *LNCS*, pp479–491, Dortmund, Germany. 7th Fuzzy Days, Springer.

Scott *et al.*, 1998. M.J.J. Scott, M. Niranjan, and R.W. Prager. Realisable classifiers: Improving operating performance on variable cost problems. In P.H. Lewis and M.S. Nixon, eds., *Proc. 9th British Machine Vision Conference*, vol 1, pp304–315, University of Southampton, UK.

Sirlantzis *et al.*, 2001. K. Sirlantzis, M.C. Fairhurst, and M.S. Hoque. Genetic algorithms for multi-classifier system configuration: A case study in character recognition. In J. Kittler and F. Roli, eds., *Second International Conference on Multiple Classifier System*, pp99–108.

Sollich and Krogh, 1996. P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. In D.S. Touretzky *et al.*, eds., *Advances in Neural Information Processing Systems*, vol 8, pp190–196. MIT Press.

Soule, 1999. T. Soule. Voting teams: A cooperative approach to non-typical problems using genetic programming. In W. Banzhaf *et al.*, eds., *GECCO-1999*, vol 1, pp916–922, Orlando. Morgan Kaufmann.

Swets *et al.*, 2000. J.A. Swets, R.M. Dawes, and J. Monahan. Better decisions through science. *Scientific American*, pp70–75, October.

Teller and Andre, 1997. A. Teller and D. Andre. Automatically choosing the number of fitness cases: The rational allocation of trials. In J.R. Koza *et al.*, eds., *GP'97*.