EuroGP'2001 Julian Miller, Marco Tomassini, Pier Luca Lanzi, Conor Ryan, Andrea G. B. Tettamanzi and W.B. Langdon editors, pages 87–96 Lake Como, Italy, 18–20 April 2001, LNCS 2038, Springer-Verlag

Evolving Receiver Operating Characteristics for Data Fusion

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. It has been suggested that the "Maximum Realisable Receiver Operating Characteristics" for a combination of classifiers is the convex hull of their individual ROCs [Scott *et al.*, 1998]. As expected in at least some cases better ROCs can be produced. We show genetic programming (GP) can automatically produce a combination of classifiers whose ROC is better than the convex hull of the supplied classifier's ROCs.

1 Introduction

There is considerable interest in automatic means of making large volumes of data intelligible to people. The terms Data Mining and Knowledge Discovery are commonly used for this. 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. 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 [Freitas, 1999]. Of particular interest is their use to perform drug discovery for the pharmaceutical industry. To this end we have investigated four techniques and compared the accuracy of the classifiers produced and the computer time needed to train them [Burbidge *et al.*, 2001].

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 also has implications for patient well being. Receiver Operating Characteristics (ROC) curves allow us to graph the trade off each classifier makes between "false positive rate" (false alarms) and "true positive rate" [Swets *et al.*, 2000]. An example ROC curve is shown in Fig. 2. We can 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. 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. We score our each classifier by the area under its ROC curve. An ideal classifier has an area of one. We will also require our initial 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.

In the next section we discuss Scott's "Maximum Realisable" ROC then (Sect. 3) we present a simple example and show we can easily do better than the "Maximum Realisable ROC". Section 4 shows genetic programming can automatically do better. The discussion of our results (Sect. 5) is followed by our conclusions (Sect. 6) and plans (Sect. 7).

2 "Maximum Realisable" ROC

[Scott *et al.*, 1998] say the "best" way of combining two (or more) classifiers is by taking the convex hull on the ROC curve. I.e. a random linear combination of them both. Scott proves it is always possible to form a classifier with true positive v. false positive rates lying between those of two existing classifiers (A and B) by combing the classifiers. The combination is done by randomly choosing between the answers between given by A and B. E.g. if a ROC point on a line half way between the ROC points of A and of B is needed, then the composite classifier will randomly give the answer given by A half the time and that given by B the other half. (Of course it may not be straightforward to persuade patients to accept such a random diagnose).

3 An example

Figure 1 shows an example of a deterministic classification problem. Given the features (inputs) the data lie in one of two classes, called positive and negative. The response of two classifiers X and Y are plotted in Fig. 1.

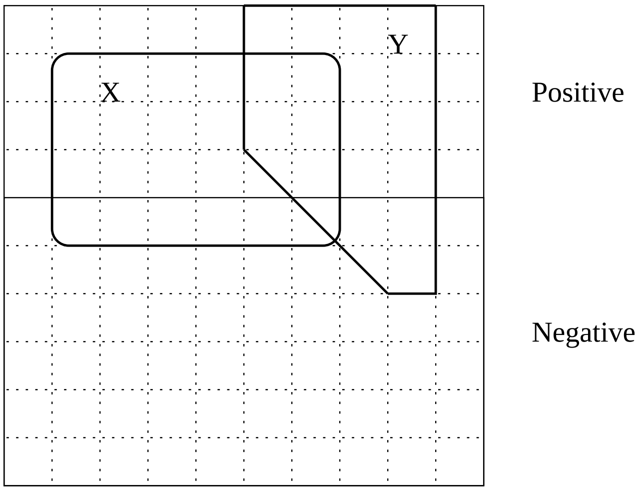

Fig. 1. Data points above horizontal line are in the class, while those below it are not. The deterministic classifiers X and Y return true if the input lies within them and otherwise false.

The performance of the classifiers X and Y in terms of the fraction of positive examples correctly classified as positive (true positives) etc. are calculated in Table 1 and shown graphically in Fig. 2. Also calculated are the performance of two classifier created by taking the union of X and Y and from the intersection of X and Y. Note the performance of both of the combined classifiers is superior to that indicated by the convex hull of X and Y.

Table 1. Classifier Accuracies

Classifier	True Positive	True Negative	False Positive	False Negative
X	18.0/40 = 0.45	54.0/60	6.0/60 = 0.1	22.0/40
Y	15.5/40 = 0.3875	56.0/60	4.0/60 = 0.066666	24.5/40
X ∪ Y	28.0/40 = 0.7	50.5/60	9.5/60 = 0.158333	12.0/40
X ∩ Y	5.5/40 = 0.1375	59.5/60	0.5/60 = 0.008333	34.5/40
1	40.0/40 = 1	.0/60	60.0/60 = 1	.0/40
0	.0/40 = 0	60.0/60	.0/60 = 0	40.0/40

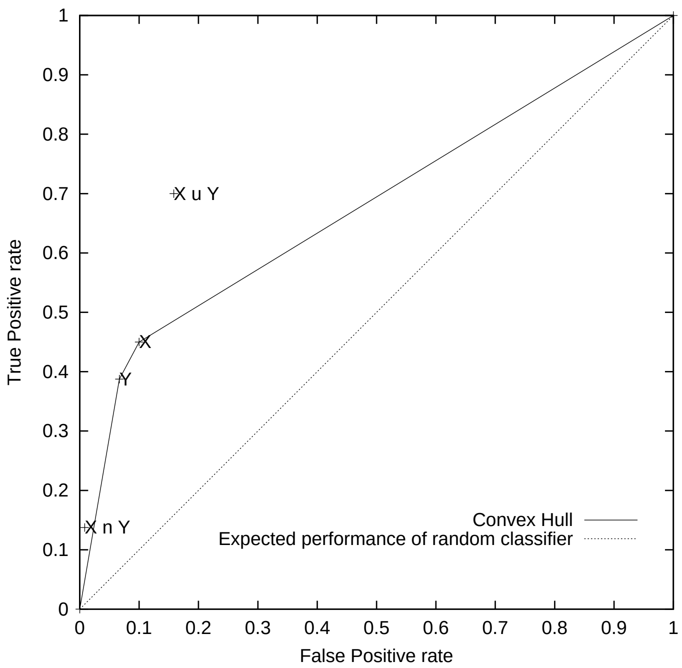

Fig. 2. Receiver Operating Characteristics (ROC) of: the two classifiers shown in Fig. 1, their union (X ∪ Y) and their intersection (X ∩ Y). The convex hull of X and Y is also plotted. Note, in this example, both the classifier formed by taking the union of X and Y and that formed by taking their intersection lie outside the convex hull.

4 Evolving a combined Classifier

4.1 Function and Terminal Sets

The function set [Langdon, 1998] included the four floating arithmetic operators ($+$, $\times$, $-$ and protected division) and Maximum and Minimum (both takes two arguments) and IFLTE. See Table 2. 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.

The fixed threshold classifiers shown in Fig. 1 are both extended to include a threshold. When this is zero, the area both classifiers say is positive shrinks to a single point. See Figs. 3 and 4. As the threshold is increased, their positive areas increase. X's positive area is always rectangular (until it expands into the boundary of the unit square). Y's lozenge shape grows from (0.9,1.0). Only at the largest threshold values does Y suggest the area right of $i = 0.9$ is positive. Eventually it, like X, expands its positive area to cover the whole space.

The two classifier X and Y are represented as floating point functions. Their threshold is supplied as their single argument. If the current fitness test point lies within the decision boundary the function returns a positive value and if it

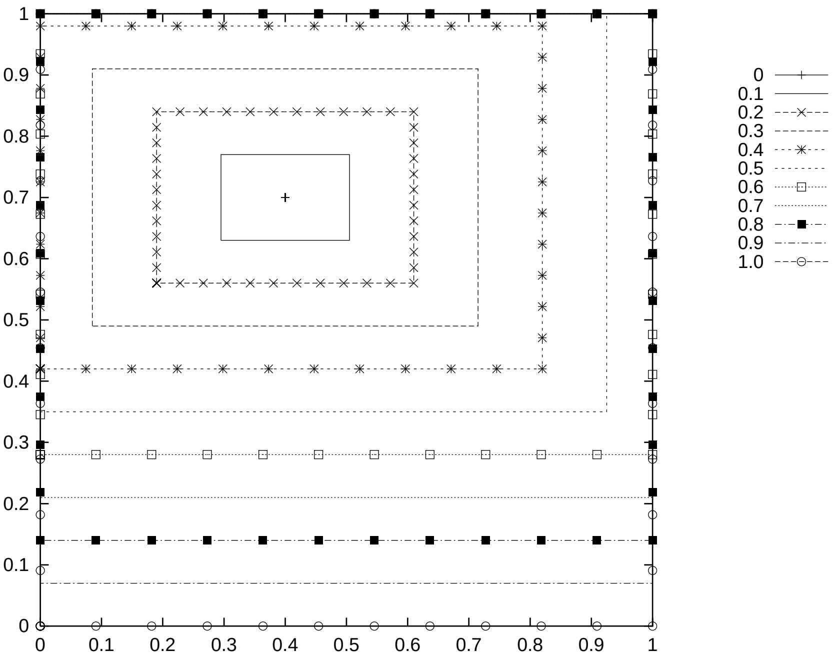

Fig. 3. Decision boundary for classifier X at 11 thresholds.

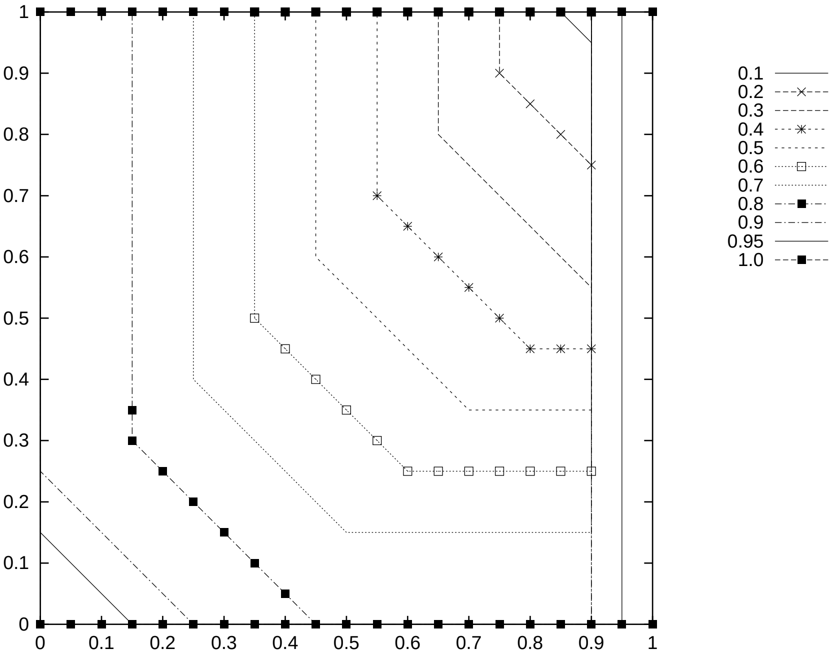

Fig. 4. Decision boundary for classifier Y at 11 thresholds.

is outside a negative value. The magnitude of the result is related to the distance between the test point and the boundary (with the given threshold setting).

There is one terminal. This is the current value of the threshold being applied to the classifier being evolved by GP. Finally the GP population was initially constructed from 200 randomly chosen floating point values between -1 and +1. These constants do not change as the population evolves.

4.2 Fitness Function

For each new invidual, 1024 test points were uniformly randomly chosen in the unit square (cf. Fig. 1). Each new individual is tested on each with the threshold parameter (T) taking values from 0 to 1 every 0.1 (i.e. 11 values). So it is run 11,264 times. For each threshold value the number of correct positive cases it gets right divided by the total number of positive cases, i.e. the true positive rate, is calculated. Similarly its false positive rate is given by the number of negative cases it gets wrong divided by the total number of negative cases.

The eleven true positive and false positive rates are plotted and the area under their convex hull is calculated. 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 when constructing the convex hull. The area under the convex hull is the fitness of the individual GP program. Note the GP is not only rewarded for getting answers right but also (using the threshold parameter) to get a range of high scores.

Table 2. GP Parameters

Objective:	Evolve a function of X and Y classifiers with Maximum Convex Hull Area
Functions set:	XC YC Max Min MUL ADD DIV SUB IFLTE
Terminal set:	T, 200 constants randomly chosen in $-1 \ldots +1$
Fitness:	Area under convex hull of 11 ROC points 1024 randomly chosen points
Selection:	generational (non elitist), tournament size 7
Wrapper:	$\geq 0 \Rightarrow$ positive, negative otherwise
Pop Size:	500
Max Program Size:	200
Initial pop:	ramped half-and-half (randomly half terminals are T, the other half are constants)
Parameters:	90% crossover 10% mutation (point mutation 2.5%, Constant change 2.5%, Shrink mutation 2.5% subtree mutation 2.5%)
Termination:	by generation 50

4.3 Results

After 50 generations on the first run GP the best program in the population contained 154 terminals and functions and had a fitness of 0.94268 on the training set. On another randomly chosen set of verification points the area under the convex hull of its ROC was 0.936052. The ROC is plotted in Fig. 5. It can be seen that the evolved classifier has a ROC outside the convex hull of the two supplied classifiers. It is also better than both the simple union and the intersection of them.

The phenotypic behaviour of the program is a fairly complex combination of the X and Y classifiers which depends upon the threshold parameter (see Fig. 6). At low thresholds the GP appears to be trying to combine X and Y to yield a decision boundary lying primarily in the upper (positive) region of the unit square. While with higher threshold much of the input area is covered with less apparent discrimination. In this way the GP produces a genuine ROC curve with a large area. (On other problems the ROC curves were less smooth and well behaved and high scores were obtained even though only a few of the ROC points lay on the outside of the convex hull).

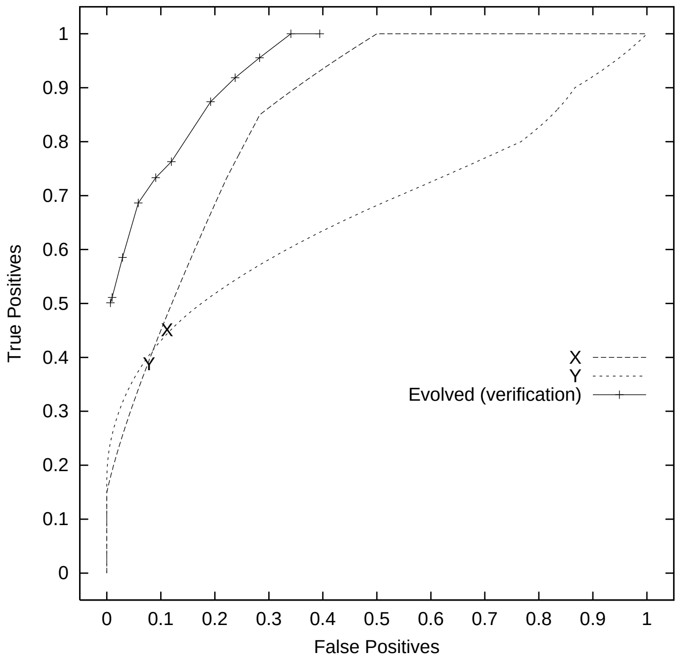

Fig. 5. Receiver Operating Characteristics (ROC) of evolved and two given classifiers, X and Y (cf. Fig. 1).

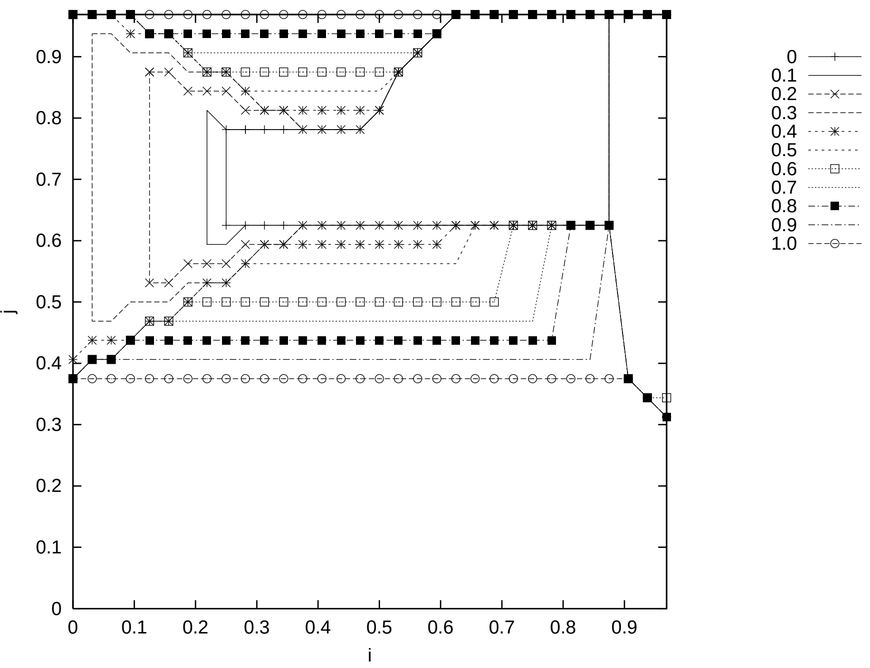

Fig. 6. Decision boundary for the evolved classifier at 11 thresholds. The GP classifier says points inside the boundary are positive. Remember only points above 0.6 are actually positive, cf. Fig. 1. Notice that even at low threshold values approximately half the positive region (and the whole of the negative region) are correctly classified. The positive area grows as the threshold is increased. Like X and Y (cf. Figs. 3 and 4) all the positive area is correctly classified when the threshold reaches its maximum value. However, unlike X and Y, the GP classifier does not cover the whole space but instead correctly classifies more than half of the negative region. (Data collected on $1/32 \times 1/32$ grid).

5 Discussion

In this experiment we demonstrate that better than convex hull performance can be automatically obtained in at least some cases. Indeed the composite classifier exceeds the performance of both given classifiers at all threshold values. In other experiments we have demonstrated, on a few small but real world benchmarks, that GP can evolve combined classifiers with performance, which exceed that of the convex hull of the individual classifiers, at all threshold values. That is there are at least some real cases where GP can automatically produce combined classifiers with ROC performance better than [Scott *et al.*, 1998]'s "Maximum Realisable Receiver Operating Characteristics".

So far GP has been given simple classifiers. These have few parameters. While this limits the complexity of their decision boundaries, it tends to make them robust and have little risk of over fitting. In contrast GP is very flexible and one normally has to be wary of over fitting. So far we have seen little sign of it, nevertheless when dealing with more real applications, it is something that will require care. We anticipate over fitting could arise both in the methods used to create the initial classifiers and in the GP. On the face of it, one would need to avoid over fitting from both sources. We would expect care to be taken to avoid over fitting in the production of the initial classifiers, be they artificial neural networks, SVMs or other techniques. There are a number of approaches to avoiding over fitting in GP. These often stress the use of "parsimony" fitness or other mechanisms to reduce the size of the evolved programs [Langdon *et al.*, 1999]. Early experiments with size fair crossover [Langdon, 2000b] and high mutation rates seem encouraging, in that we have seen neither rapid increase in programs size, "bloat", or over fitting. We suspect the two are related.

This approach to combining classifiers allows arbitrary combinations of different classifiers. For example we might fuse neural networks, decision trees, SVMs and Naive Bayes classifiers. In contrast "boosting" attempts to get better performance, with a single trade off between true and false positive rates, from a single "weak learner" by repeated relearning on different subsets of the training data. While we have obtained improved performance across the whole ROC range. We expect the best results when GP is given different and complementary classifiers.

On a benchmark problem we have estimated classification rates in the order of 300 10^6 per hour per cpu. This is a simple estimate of the raw speed of a combination of Bayes classifiers evolved by GP and does not include file I/O or network overheads. Nevertheless this is an encouraging figure.

6 Conclusions

As expected and has previously been shown [Yusoff *et al.*, 1998] Scott's "Maximum Realisable" ROC [Scott *et al.*, 1998] need not be the best that can be obtained by combining classifiers. While [Langdon, 2000a] gives a single example point which improves on Scott's convex hull, we have shown a complete Receiver Operating Characteristics (ROC) curve can be automatically generated from existing fixed classifiers using genetic programming (GP) to simultaneously combine their answers and set their operating thresholds.

7 Future Work

We are applying GP to Machine Learning benchmarks [Langdon and Buxton, 2001] and hope to apply it to QSAR (Quantitative Structure Analysis) problems. Our immediate goal is to use this for data fusion within the pharmaceutical industry.

References

[Burbidge *et al.*, 2001] R. Burbidge, M. Trotter, B. Buxton, and S. Holden. Drug design by machine learning: Support vector machines for pharmaceutical data analysis. *Computers in Chemistry*, 2001. Under review.

[Freitas, 1999] Alex A. Freitas. Data mining with evolutionary algorithms: Research directions. Technical Report WS-99-06, AAAI, Orlando, USA, July 1999. Papers from the AAAI Workshop `http://www.aaai.org/Press/Reports/Workshops/ws-99-06.html`.

[Langdon and Buxton, 2001] W. B. Langdon and B. F. Buxton. Genetic programming for combining classifiers. 2001. Submitted.

[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, Cambridge, MA, USA, June 1999.

[Langdon, 1998] William B. Langdon. *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming!*, volume 1 of *Genetic Programming*. Kluwer, Boston, 1998.

[Langdon, 2000a] W. B. Langdon. Problems with ROC convex hulls. `http://www.cs.ucl.ac.uk/staff/W.Langdon/roc.ps.gz`, 24 May 2000.

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

[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, Florence, Italy, 7-11 June 1998.