A Classifier Combination Tree Algorithm

Ross A. McDonald[1], Idris A. Eckley[2], and David J. Hand[1]

[1] Imperial College London
[2] Shell Research Ltd.

Abstract. In recent years a number of authors have suggested that combining classifiers within local regions of the measurement space might yield superior classification performance to rigid global weighting schemes. In this paper we describe a modified version of the CART algorithm, called ARPACC, that performs local classifier combination. One obstacle to such combination is the fact that the 'optimal' covariance combination results originally assumed only two classes and classifier unbiasedness. In this paper we adopt an approach based on minimizing the Brier score and introduce a generalized matrix inverse solution for use in cases where the error matrix is singular. We also report some preliminary experimental results on simulated data.

Keywords: Local Combination, Brier Score, CART

1 Introduction

The notion that the combination of classifiers based on local accuracy estimates may prove superior to an ensemble based on globally defined weights is one that has emerged several times in recent years ([10],[9],[13],[2]), though as yet there would seem to exist little understanding of why such methods should work in practice beyond the purely intuitive, and no algorithmic method for defining a useful partition of the feature space for local combination. In this paper we propose using a modified version of the CART tree algorithm [3] for this purpose. We combine the classifiers in our ensemble locally, by assigning separate weighting schemes to each leaf node of our 'combination tree'. Note the distinction between this idea and the model trees proposed by other authors [6], where local models are fitted within nodes of a tree; we assume that our ensemble classifiers are induced using the full design set, and given this our aim is to combine their output in the most efficient way possible.

Our algorithm, called ARPACC (Automatic Recursive Partitioning Algorithm for Classifier Combination), relies on the use of the 'optimal' results for the weighted sum combination of classifiers through the minimisation of combined error covariance. We have extended these results in a number of ways. We drop the assumption of classifier unbiasedness with respect to the training data, and propose a natural extension to the case of more than two classes. We motivate this through the use of the *Brier score* measure of classifier fit. We also introduce a solution for the optimal weights in cases where the error matrix is singular, together with a necessary and sufficient condition for this to be

A. Fred et al. (Eds.): SSPR&SPR 2004, LNCS 3138, pp. 609–617, 2004.

true (singular error matrices become more common as we consider smaller and smaller subsamples of the design data).

Our intention is to increase the flexibility of our combination scheme and thereby exploit a corresponding decrease in the bias of the combined classifier. The computation of the optimal weighting scheme across a node depends, however, upon our ability to estimate the mean error of a classifier across the corresponding subsample of the design data. As we recursively partition our training space the variance of these estimates is bound to increase, due to the diminishing size of the sample over which it is calculated. This is the familiar bias-variance tradeoff that affects most recursive classification methods. We have observed, however, that ARPACC trees tend to be robust to overfitting especially when combining classifiers that tend naturally to underfit. This is most likely to occur in feature spaces of high dimensionality, or when the number of class labels is large.

In Section 2 of this paper we describe the optimal combination scheme argument that we use to calculate weighting schemes for terminal nodes. Section 3 details the ARPACC algorithm and the results of some simulated trials, while Section 5 provides a visual illustration of local classifier combination.

2 Optimal Weighted-Sum Combination

Suppose that we are presented with a supervised classification problem in some measurement space $\mathbf{x}$, where for a fixed location the posterior class probability of observing class c is generated by some underlying target function $f(c|\mathbf{x})$ taking real values in $[0, 1]$. Suppose also that we have an ensemble of classifiers $\hat{f}_k(c|\mathbf{x})$ that are approximations to this underlying function. Our goal is to combine these classifiers in such a way as to ensure optimal classifier performance, measured in terms of test error rate on future observations. Kittler [11] has demonstrated that one of the most robust combination strategies in this circumstance is the weighted sum rule, or

$$\hat{f}(c|\mathbf{x}) = \sum_{k=1}^{K} w_k \hat{f}_k(c|\mathbf{x}).$$

where the w_k are positive classifier weights to be determined. It is natural to treat this as a regression problem and adopt the strategy of minimizing the total mean-squared distance across the training data between each observed value and our approximation,

$$\frac{1}{N} \sum_{c} \sum_{n=1}^{N} \left(\delta(c|\mathbf{x}_n) - \hat{f}(c|\mathbf{x}_n) \right)^2,$$

where $\delta(c|\mathbf{x}_n) = 1$ if training example n has class c, and 0 otherwise. The above is the *Brier* or *quadratic* score, a common measure of classifier performance. The Brier score is an estimate of the *Brier Inaccuracy* over the full space χ:

$$E_{\chi} \sum_{c} E_{\delta|c,\mathbf{x}} \left(\delta(c|\mathbf{x}) - \hat{f}(c|\mathbf{x}) \right)^2.$$

It can be shown (see [7]) that minimizing the Brier inaccuracy is equivalent to minimising

$$E_\chi \sum_c (f(c|\mathbf{x}) - \hat{f}(c|\mathbf{x}))^2,$$

which is the mean squared error of our $\hat{f}$ relative to f. This is important because it means that in the absence of any knowledge about $f(c|\mathbf{x})$, we can use our observed design set class labels as a proxy in solving for the optimal weights. Of course, any solution derived in this way is only 'optimal' with respect to the training data: as always we rely on a large sample that is in some sense a good reflection of the target function.

If we denote the 'error' of a particular classifier within our ensemble relative to a specific training object as $e_k(\mathbf{x}_n) = \delta(c|\mathbf{x}_n) - \hat{f}_k(\mathbf{x}_n)$ for class c, then minimising the Brier score of our ensemble is the same as minimising

$$\frac{1}{N} \sum_{n=1}^{N} \sum_c \left[\sum_{k=1}^{K} w_k e_k(\mathbf{x}_n) \right]^2,$$

which may be written in matrix form as $w'\mathrm{E}w$, where w is a vector of weights and the ijth entry of the square 'error' matrix E, $e_{ij} = \frac{1}{N} \sum_n [e_i e_j]$. Prior applications of this method set out to minimize either the total error variance of the combined classifier, or the total mean square error of a two class combined classifier. By adopting the multi-class Brier score, we are able to extend the solution to more than two classes in a natural way. We assume that each class label has its own separate associated weighting scheme, so that by minimizing the error of the combined classifier for each $\delta(c|\mathbf{x})$ we are minimizing the total Brier score.

The solution for the weights vector was first derived by Bates and Granger in 1969 [1] for ensembles of size two, and extended to ensembles of more than two classifiers by Dickinson [4]. Dickinson began by assuming that the errors were normally distributed with mean zero, ie. that $E_\chi e_k = 0 \; \forall \; k$, and this assumption leads to a solution that minimizes the error *variance* of the combined classifier. In fact the assumption is unnecessary, and for our purposes it would clearly be restrictive. Across the global space it may be fair to assume that the expected error of a classifier is zero with respect to the target function, but within a small local region of that space this is very unlikely to be true.

Dropping the assumption leads us to the following revised solution for classifier combination based on mean square error, derived by Hashem and Schmeiser [8] in the context of regression neural networks. They impose the restriction that the w_k should sum to one by introducing the Lagrange multiplier λ, and minimising the function

$$\sigma = \sum_{i,j}^{K} w_i w_j e_{ij} + \lambda \left(\sum_{i=1}^{k} w_i - 1 \right).$$

We differentiate with respect to each of the parameters $(w_1, ..., w_k, \lambda)$ in turn, so that

$$\frac{\delta\sigma}{\delta w_i} = 2\sum_{j=1}^{K} w_j e_{ij} + \lambda \qquad (i = 1, ..., K)$$

and

$$\frac{\delta\sigma}{\delta\lambda} = \sum_{i=1}^{K} w_i - 1.$$

Setting these equal to 0, we get

$$\begin{pmatrix} 1'_K & (0) \\ E & 1_K \end{pmatrix} \begin{pmatrix} w_{\min} \\ \lambda/2 \end{pmatrix} = \begin{pmatrix} (1) \\ 0_K \end{pmatrix}$$

where 1_K and 0_K are the column vectors of K ones and K zeros respectively, and the notation () denotes a single integer. Thus

$$\begin{pmatrix} w_{\min} \\ \lambda/2 \end{pmatrix} = \begin{pmatrix} 1'_K & (0) \\ E & 1_K \end{pmatrix}^{-1} \begin{pmatrix} (1) \\ 0_K \end{pmatrix}.$$

The solution to the above is given by the first entry in the inverted matrix, so:

$$w_{\min} = \frac{E^{-1}1_K}{1'_K E^{-1} 1_K}.$$

Negative weights may arise for ensembles of more than two classifiers. In extreme cases, this can lead to combined predictions outside the range $[0, 1]$. As a stop-gap measure, we can deal with this by setting negative output predictions to 0, and output predictions greater than 1 to 1.

In the course of our research we also extended the solution to cases where the matrix E proves to be singular. This can frequently happen by accident, especially where component classifiers make similar predictions, or where we perform a large number of calculations on training sets of diminishing size, as with the algorithm in the next section. As we base our combination weights on smaller and smaller subsets of the training data, the likelihood of our encountering a singular error matrix becomes very high. Our more general solution is as follows:
If E is non-singular

$$w_{min} = \frac{E^{-1}1_K}{1'_K E^{-1} 1_K}.$$

If E is singular form the matrix $Q = E + 1_K 1'_K$. If Q is non-singular

$$w_{min} = Q^{-1} 1_K.$$

If Q is singular

$$w_{min} = \frac{Q^g 1_K}{1'_K Q^g 1_K}$$

where g denotes the generalized matrix inverse. We derived the following as a necessary and sufficient condition for the matrix E to be singular. Let $\hat{Y}_i$ denote

the vector of predictions of ensemble classifier i on the training data and Y denote the vector of true class labels. If the matrix E is singular, then for some subset(s) of classifiers $i \in S$, say, there exists a linear combination $\sum_{i \in S} s_i \hat{Y}_i$ such that

$$(\sum_{i \in S} s_i) Y = \sum_{i \in S} s_i \hat{Y}_i.$$

The converse is also true.

The above condition will be met if one or more classifiers match the training data predictions exactly (but note that this does not necessarily imply that the classifiers are identical). It will also be met if two or more classifiers, or linear combinations of subsets of classifiers, make identical predictions (this corresponds to $\sum_{i \in S} s_i = 0$). This is most likely to happen when the number of training samples is small, as for a fine partition of the training data.

It is also possible to show that the total squared error of the combined ensemble is less than or equal to that of any component classifier. The proof is similar to the proof in [5] relating to combined variance.

3 The ARPACC Algorithm

In this section we outline the results of some preliminary experiments on simulated data produced using ARPACC, a modification of the CART algorithm [3] that uses the optimal weighting results of the previous section to build classifier combination trees.

To modify the CART algorithm to perform local classifier combination we need to change the the splitting criterion and the calculation of node predictions. Rather than associating each terminal node with a prediction based on the assigned training data, each node is treated as a partition and is assigned the optimal local weighting scheme for the given training sample at that node, calculated via the expressions at the end of the previous section. The splitting criterion is calculated by applying these weights to the predictions of the classifiers under consideration, and summing the training error of the combined classifier.

We deal with problems of more than two classes by associating a separate set of weights with each class (so that we might simultaneously decide that we trust ensemble classifier A's class 1 predictions, but assign a low weight to its class 3 predictions, for example). During splitting the total current error of the tree is replaced by the expected Brier score across all nodes and training examples.

The chief difference between this and the standard tree classifier is that our algorithm *does not make any predictions in its own right.* The model output by ARPACC is only a framework in which global classifiers are combined locally. This is underlined by the fact that any or all of these classifiers could themselves be trees.

Our first simulated experiment consists of a series of trials, each comprising five random repetitions of classifier fitting and combination. In each trial we first fix the value of three parameters: s, the sample size for each class, d, the dimensionality of the problem, and c, the total number of classes.

For each class and each dimension, we generate two integer values between 1 and 100 uniformly and at random, and then generate s random normal variates using these values as the mean and variance. These values are rounded down to the nearest integer (to speed up computation). The resulting clusters are rotated by forty-five degrees in consecutive pairs of dimensions. Two classifiers are fitted to the training sample: a CART tree computed using S-Plus with the default parameters, and Venables and Ripley's multinomial method, computed with S-Plus using the function *multinom* [12], which uses neural networks to fit log-linear models. The CART tree stopping criterion is a node size of 5, and the ARPACC trees are grown to a fixed size of 50 terminal nodes.

Five trials are run for each choice of parameters. The results in Table 1 below summarize the sample sizes, and the mean improvement of the global and local combination methods over the best single-classifier test error rate, as a percentage of points misclassified.

Table 1. Results of simulated experiments with a local combination algorithm.

Classes	Dimensions	Train Size	Test Size	Global Test Error Improvement	ARPACC Test Error Improvement
2	2	2,000	2,000	0.00% s.d 0.14	0.16% s.d 0.13
3	3	3,000	3,000	-0.03% s.d 0.10	0.52% s.d. 0.42
5	5	5,000	5,000	1.00% s.d. 1.12	2.90% s.d. 1.55
6	4	2,000	2,000	0.42% s.d. 0.34	1.70% s.d. 0.39
10	8	2,000	2,000	1.72% s.d. 2.89	3.90% s.d. 2.00

The above results would appear to suggest that the performance of the ARPACC tree improves as the number of dimensions and classes is increased. We believe that this is due to the propensity of our standard classifiers to underfit problems of high dimension, with the combination tree providing us with a measured means of correcting for this underperformance.

4 Combining a Tree and a Neural Network Using ARPACC

The aim of this section is to provide a visual representation of what an ensemble classifier created using the ARPACC algorithm might look like. We begin by generating 500 training and 2,000 test observations distributed uniformly in two dimensions, and use the target probability function

$$f(1|\mathbf{x}) = \frac{cos\left(2\pi\sqrt{(x_1^2 + x_2^2)}\right) + 1}{2}$$

to randomly assign each observation to one of two categorical classes. Panel 1 in Figure 1 shows how the design sample appears in two dimensional space. Panel 2 is a contour plot of the generating function f across the same range.

To our design sample we fit a CART tree, using the *tree* function in S-Plus, and a neural net created using Venables and Ripley's neural net library for S-Plus [12]. This is a 2-3-1 neural network with 13 weights, and the fitting algorithm is allowed to run for 100 iterations.

Contour plots of the predictions made by the tree model and the neural network model are shown in panels 3 and 4 of figure 5. Here the greyscale level represents the 'confidence' level of the prediction, ranging from black (high confidence in the first class) to white (high confidence in the second class). The test error rates for the tree and the neural net, ie. the number of misclassified test examples divided by the total number of test examples, are 0.221 and 0.2225 respectively. Although these test error rates are close, a visual inspection tells us that the forms of the fitted models are in fact very different, and that both underfit. Empirically we have found that the pairing of a tree and a neural net model is a good one for combination, since they tend to complement each other well.

We now combine the two models globally using two sets of weights generated using the optimal MSE solution. The weights are 0.91 for the tree and 0.09 for the neural network. This means that the combined model heavily favours the tree over the neural network. The resultant predictions are plotted in panel 5. The test error rate of this globally combined model is 0.215, lower than that of either of the component models. Visually, the combined classifier appears a slightly better fit to the true f.

Our final model is that produced when the ARPACC algorithm uses the training data to fit a fifty node combination tree. The test error rate for this model is 0.2, a further improvement on the global combination. A visual examination of the plot in panel 6 tells us that at least some of this improvement probably results from an improved model fit in the region around the centre of the plot.

The test error results for all our models, as a fraction of points misclassified, are summarized in Table 2.

Table 2. Absolute training and test errors for the models used in the simulated example.

Model	Training error	Test Error
CART Tree	0.1280	0.2210
Neural Net	0.1780	0.2225
Global Combined Model	0.1260	0.2150
50-Node ARPACC	0.1100	0.2000

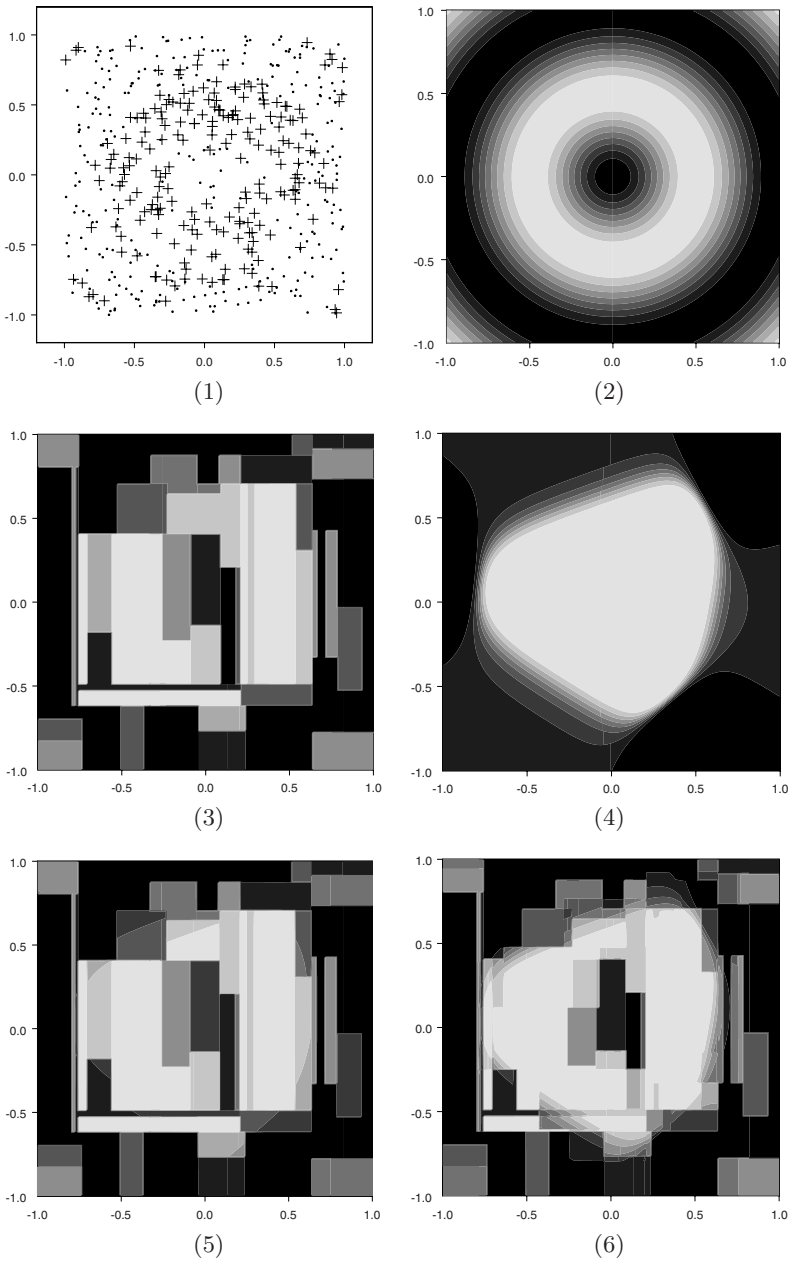

Fig. 1. An example of local classifier combination. (1) 500 data points in 2 dimensions randomly mapped to 2 classes with probabilities generated via a smooth cosine function. (2) The underlying function f. (3) A CART tree fitted to the training data. (4) A 2-3-1 neural network fitted to the training data. (5) The optimal global combined classifier. (6) A local combination as fitted by a 50-node ARPACC tree.

5 Conclusions and Future Work

In this paper we have proposed the use of optimal weights within local regions of the measurement space, have introduced solutions for more than two classes and singular error matrices, and have experimented with the ARPACC algorithm as one means of defining a suitable feature space partition.

We have also experimented with a pruning method for ARPACC based on the minimal cost-complexity pruning methods used for CART. At present our Matlab implementation of this proves computationally expensive, and we hope to improve it in the future. We also plan to experiment with ensembles of more than two classifiers, and to produce test error results for real classification problems.

Acknowledgements

RAM was supported in this work by Shell Research Ltd. and research grant number 0130322X from the Engineering and Physical Sciences Research Council.

References

1. J. M. Bates and W. J. Granger. The combination of forecasts. *Operational Research Quarterly*, 20:451 – 468, 1969.
2. H. Blockeel and J. Struyf. Frankenstein classifiers: Some experiments on the sisyphus data set. *Proceedings of IDDM 2001*, pages 1 – 12, 2001.
3. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, U.S., 1984.
4. J. P. Dickinson. The combination of short term forecasts. *Proc. Univ. of Lancaster Forecasting Conference*, 1972.
5. J. P. Dickinson. Some comments on the combination of forecasts. *Operational Research Quarterly*, 26:205 – 210, 1975.
6. E. Frank, Y. Wang, S. Inglis, G. Holmes, and I. Witten. Using model trees for classification. *Machine Learning*, 32:63–76, 1998.
7. D. J. Hand. *Construction and Assessment of Classification Rules*. John Wiley & Sons, Chichester, 1997.
8. S. Hashem and B. Schmeiser. Improving model accuracy using optimal linear combinations of trained neural networks. *IEEE Transactions on Neural Networks*, 6:792 – 794, 1995.
9. T. M. Joergensen and C. Linneberg. Feature weighted ensemble classifiers - a modified decision scheme. In *Multiple Classifier Systems (MCS) 2001*, pages 218 – 227. Springer-Verlag, 2001.
10. M. S. Kamel and N. M. Wanas. Data dependence in combining classifiers. In *Multiple Classifier Systems (MCS) 2003*, pages 1 – 14. Springer-Verlag, 2003.
11. J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3), 1998.
12. W. Venables and B. Ripley. *Modern Applied Statistics with S-Plus*. Springer-Verlag, 1994.
13. K. Woods, W. P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machie Intelligence*, 19:405 – 410, 1997.