Decision Forests with Oblique Decision Trees

Peter J. Tan and David L. Dowe

School of Computer Science and Software Engineering, Monash University,
Clayton, Vic 3800, Australia
ptan@bruce.csse.monash.edu.au

Abstract. Ensemble learning schemes have shown impressive increases in prediction accuracy over single model schemes. We introduce a new decision forest learning scheme, whose base learners are Minimum Message Length (MML) oblique decision trees. Unlike other tree inference algorithms, MML oblique decision tree learning does not over-grow the inferred trees. The resultant trees thus tend to be shallow and do not require pruning. MML decision trees are known to be resistant to overfitting and excellent at probabilistic predictions. A novel weighted averaging scheme is also proposed which takes advantage of high probabilistic prediction accuracy produced by MML oblique decision trees. The experimental results show that the new weighted averaging offers solid improvement over other averaging schemes, such as majority vote. Our MML decision forests scheme also returns favourable results compared to other ensemble learning algorithms on data sets with binary classes.

1 Introduction

Ensemble learning is one of the major advances in supervised learning research in recent years [10]. The outputs of an ensemble classifier are determined by a committee, in which a group of classifiers cast (possibly weighted) votes on final predictions. Generally, ensemble learning schemes are able to outperform single classifiers in predictive accuracy. The intuitive explanation for the success of ensemble learning is that mistakes made by individual classifiers are corrected by complementary results submitted by other classifiers in the committee.

The most widely adopted approaches are called Perturb and Combine (P&C) methods. P&C methods infer each classifier from a set of distinct training sets, which are generated by perturbing the unaltered original training set. An important prerequisite for the P&C methods is that the base learner must be unstable in that the inferred models are sensitive to even minor variation of the training set. Bagging (bootstrap aggregating) [4] and AdaBoost (adaptive boosting) [17] are two popular P&C methods implemented in many ensemble learning schemes. Both methods are able to generate diverse committees by feeding base learners with a set of distinct training sets drawn from the original training set.

Another way to create variations in the training sets is to alter the label of the target attribute of instances in the training set. Breiman proposed a scheme [3] which grows a set of decision trees by injecting random noise into the output

label of the original training set. DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples) [20] is another ensemble learning scheme which has a similar motivation. But, instead of randomly altering the output labels, the algorithm inserts the artificially constructed instances (and adds to the training set) with the aim of deliberately increasing diversity among the inferred committee members.

Decision tree inducers are unstable in that resultant trees are sensitive to minor perturbations in the training data set. Largely for this reason, decision trees are widely applied as the base learners in ensemble learning schemes. Some ensemble algorithms have implemented modified decision tree inference algorithms in order to generate diverse decision forests. Ho proposed a scheme called the random subspace method [18] for constructing decision forests. When the algorithm constructs a split at each internal node of the inferred trees, candidate features are restricted to a randomly selected subset of the original input features. Dietterich introduced another ensemble learning scheme [11] that does not rely on the instability of the decision tree inducer. Instead of picking the split with the best score on the objective function, the algorithm randomly chooses a split among a pre-defined number of candidate splits with the highest score (on the objective function). Ferri et al. [16] have an interesting scheme in which the subset of the data that was deemed most difficult to classify is put aside and then delegated to another run of the classifier.

Research on combining some of the above methods to further improve the performance of ensemble learning has also shown promising results. In the random forests scheme developed in [5], bagging and random feature selection schemes were implemented to inject randomness into the decision tree growing processes.

While there are no clear winners emerging from the above ensemble schemes, all of them reported superior "right"/"wrong" predictive accuracy compared to single classifier learning schemes. In this paper, we propose a new ensemble algorithm called decision forests with oblique decision trees. The proposed ensemble learning scheme is different from other random forests in several ways. While most ensemble learning algorithms grow deep and unpruned decision trees, the base learner in our ensemble learning is Minimum Message Length (MML) oblique decision trees, which were introduced in [28]. The paper also shows how to include simple probabilistic Support Vector Machines in the internal and/or leaf nodes of decision trees. The MML coding scheme is applied to select optimal candidate trees (with overall lower MML coding) with high probabilistic prediction accuracy (low log-loss score) and smaller tree size (lower height with fewer leaf nodes). Compared to schemes with univariate trees (which cut on only one attribute at a time), using MML (multivariate) oblique trees offers potential to greatly increase the diversity of the inferred forest. A new weighted averaging scheme is also proposed. The proposed averaging scheme is based on Bayesian weighted tree averaging but uses a modified, smoothed prior on decision trees (see sec. 3.4). In order to take advantage of the above weighted averaging scheme, a new algorithm to rapidly generate a large number of distinct oblique decision trees is introduced.

2 Details of Some Related Ensemble Schemes

Bagging [4] relies on perturbing the training set. When unstable learning algorithms are applied as base inducers, a diverse ensemble can be generated by feeding the base learner with training sets re-sampled from the original training set.

Another type of P&C method is the AdaBoost algorithm, which is also referred to as an arcing (adaptive re-sampling and combining) algorithm by Breiman in [2]. The fundamental difference between bagging and AdaBoost is that while bagging is non-deterministic, AdaBoost is deterministic and iterative.

AdaBoost iteratively alters the probability over instances in the training set while performing the re-sampling. It works very well when the data is noise free and the number of training data is large. But when noise is present in the training sets, or the number of training data is limited, AdaBoost does not perform as well as Bagging and (see below) random forests.

Random forests use CART [6] as the base learner and combine several methods to generate a diverse ensemble. Each decision tree in a random forest is trained on a distinct and random data set re-sampled from the original training set, using the same procedure as bagging. While selecting a split at each internal node during the tree growing process, a random set of features is formed by either choosing a subset of input variables or constructing a small group of variables formed by linear combinations of input variables. Random forests [5] have achieved "right"/"wrong" predictive accuracy comparable to that of AdaBoost and much better results on noisy data sets. Breiman also claimed and showed that AdaBoost is a form of random forest (algorithm) [5].

3 Ensemble Learning with MML Random Forests

It has been shown that the performance of an ensemble classifier depends on the strength of individual classifiers and correlations among them [5]. MML oblique trees are shown to return excellent accuracies, especially on probabilistic predictions. The algorithm beats both C4.5 [24] and C5 on both "right"/"wrong" and especially probabilistic predictions with smaller trees (i.e., less leaf nodes) [28]. Because we would like to implement a Bayesian averaging scheme [31, sec. 8][29, sec. 4.8][13, sec. 6.1.4], performance of individual classifiers on probabilistic prediction is crucial. Therefore the MML oblique trees are chosen as the base learners in our algorithms. Due to the introduction of hyperplanes at internal nodes, the space of candidate trees is also hugely enlarged. This helps to increase diversity among the trees, especially for the data sets with fewer input attributes.

3.1 Minimum Message Length (MML)

The Minimum Message Length (MML) Principle [30, 32, 31, 27, 29] provides a guide for inferring the best model given a set of data. MML and the subsequent Minimum Description Length (MDL) principle [25, 19] are widely used for model

selection in various machine learning problems, and both can be thought of as operational forms of Ockham's razor [21]. For introductions to MML, see [29, 13]; and for detalis on MML, see [30, 32, 31, 22, 15, 7, 8]. For a comparison between MML and the subsequent MDL principle[25], see, e.g., [31] (which also gives a survey), other articles in that 1999 special issue of the *Computer Journal*, [8] and [29]. We apply the MML multivariate oblique coding scheme [28] when oblique decision trees in our new decision forests are grown.

3.2 Searching For Optimal Splits at Internal Nodes

The algorithm we propose here is tailored for searching for optimal two-dimensional hyperplanes (linear combinations of two input attributes). It works as follows: Firstly, a random two dimensional hyperplane is generated. Then the hyperplane is rotated by 10 degrees each time, so that a set of 18 orientations is generated. For each hyperplane in the 18 orientations, a maximum of 32 cut-points were tested and the one with the minimum total code length is recorded. The process is repeated on all candidate combinations of two input attributes. The total code length is given as below. For further details of the MML coding of oblique decision trees, please see [28].
Total code length = $Part1 + Part2$, where

$Part1 = -2(D-1)\log(\frac{2}{\sqrt{D\|w\|^2+4}}) + \log\binom{N}{2}$, $Part2 = \sum_{l=1}^{L} Msg_l$,

$Msg_l = \frac{M-1}{2}(\log\frac{N_l}{12}+1) - \log(M-1)! - \sum_{m=1}^{M}(n_m+\frac{1}{2})\log(\frac{n_m+\frac{1}{2}}{N_l+\frac{M}{2}})$, where

D is the dimension of the hyperplane, N is the number of data at the internal node to be split, L is the number of child nodes resulting from the split (in this case L is 2), Msg_l is the code length for encoding leaf probability and data in each resultant lth leaf node, M is the number of classes in the data, n_m is the number of instances in class m in the particular leaf node and N_l is the number of data in leaf node l. For the remainder of this paper, $D = 2$.

3.3 An Efficient Algorithm to Generate a Larger Number of MML Oblique Trees

The idea behind our new rapid forest generation algorithms here is that, at each node, a specified number of viable splits (splits which improve the message length) are recorded. For each of these candidate splits, tentative splits are performed. The above procedure is recursively run on each resulting child node.

The forest growing process is divided into the following two parts (A and B):

A: In part A, a search tree is constructed in the following steps:
Start the tree with a single leaf node as the root node,

1. Generate a set of possible combinations of two input attributes.
2. Search for the best split by hyperplanes (i.e., those yielding the shortest two-part message length) constructed from each of the given combinations of two attributes for this node.

3. Record each of the candidate splits from step 2 that achieve better message length than the unsplit leaf node.
4. For each of the splits recorded in step 3, perform a tentative split.
5. Recursively apply this procedure on each of the child nodes generated in step 4, until the height of the search tree is H.

In this way, following a branch under an internal node in the search tree represents selecting a hyperplane split, and the subsequent subtrees under this branch are a union of candidate subtrees which would possibly be generated by such a split.

B: A random decision tree can thus be created by randomly picking a branch and one of the subsequent subtrees recursively. Create such a tree. Repeat this process until a pre-defined number of trees is generated.

In order to approximate the number of searches and the number of distinct trees that are able to be generated from a search tree, we assume that there are M viable candidate binary splits at each internal node to be searched. For a search tree with height H, it is easy to see that the number of searches is $S(H, M) = 2M \times S(H-1, M)$ and $S(2, M) = M$, thus $S(H, M) = 2^{H-2}M^{H-1}, H > 1$. The number of distinct trees, T, that can be generated from the search tree can be estimated by using the fact that $T(2, M) = M$ and
$T(H, M) \approx T(H-1, M)^{2M}$, thus roughly $T(H, M) \approx M^{((2M)^{H-2})}$. To keep the computational time in rein, the algorithm puts some upper limits on M and H. In our experiments, M ranges from 25 to 50 while H ranges from 4 to 5.

3.4 Weighted Averaging of Trees

Oliver and Hand proposed a Bayesian weighted tree averaging scheme [23] in which the weights are set to be proportional to the posterior probability of each tree in the forest. Given a tree, T_i, with I internal nodes, L leaves and C classes, they give the posterior probability of the tree T_i as

$$P(T_i|D) \propto \prod_{i=1}^{I} \frac{1}{ap(i)} \prod_{i=1}^{L} (1 - \frac{1}{ap(i)}) \prod_{i=1}^{L} \frac{\prod_{j=1}^{C} M_j!}{(\sum_{j=1}^{C} M_j)!} \quad \ldots (2)$$

where $ap(i)$ is the arity of the parent of node i, and M_j is the number of the instances belonging to class j (j$\in \{1, 2, \ldots, C\}$) in each leaf node. We initially implemented a similar scheme as one of our averaging methods. However, experimental results returned by this averaging method are worse than those by the simple arithmetic vote averaging. Investigation showed that even in forests with 1000 trees, there are always 2 to 3 trees dominating the majority (.8 to .9) of the weights. While such results may seem to be contradictory to Bayesianism, there are several possible explanations for this. One possible reason is that the prior of decision trees given by $P(T_i|D) \propto \prod_{i=1}^{I} \frac{1}{ap(i)} \prod_{i=1}^{L} (1 - \frac{1}{ap(i)})$ is not right for the oblique decision trees in the forest. Another possibility is that our uniform multinomial prior on the class probabilities is not flexible enough. Rather than fix our Beta/Dirichlet prior as having $\beta = 1$, we could more generally permit

β to have a prior distribution of the rough form of $\frac{3}{2\sqrt{\beta}(1+\sqrt{\beta})^4}$ or $\frac{e^{-\frac{\beta}{\pi}}}{\pi\sqrt{\beta}}$, whose purpose is to maintain a mean equal (or ideally close) to 1 while permitting the boosting-like effect of small values of β close to 0. Another possible explanation lies in the fact that there are high correlations between the predictions submitted by the trees with the top posterior probabilities, despite their distinct tree structures. Another rather viable and simple explanation is that, like earlier studies (e.g. [23]), we have not been sampling from the posterior distribution. This could be corrected by a judicious choice of importance sampling distribution.

In this paper, we retain the uniform multinomial prior and attempt to approximate the correct weights for decision trees generated from the real posterior probabilities, proposing a new (approximate) decision tree averaging scheme for our decision forests algorithm. Because there are an indefinite number of trees in the posterior space and the real distribution of the posterior probabilities is unknown, three assumptions have been made. First, assume that the distribution of the weights of the sampled trees is like a unit Normal distribution. Secondly, we assume that the posterior $P(T_i|D)$ of the inferred trees can be sampled from the range (-R, R) (R is set to 3.5 in our tests), given that $\int_{-R}^{R} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \approx 1$. Lastly, we assume that the order of the real posterior probabilities of inferred trees is identical of that of the posterior probabilities obtained from (2). Then the weight of a decision tree, $w(T_i)$, is approximated in this scheme as follows:
1. Generate a set of weights $\{w_1, w_2, \ldots, w_S\}$, so that $w_i = \int_{x_i}^{x_i+\Delta x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$, where $x_i = -R + (i-1)\Delta x$, $\Delta x = \frac{2R}{S}$
2. Normalize and sort $\{w_1, w_2, \ldots, w_S\}$ so that $w_1 \leq w_2 \ldots \leq w_S$ and $\sum w_i = 1$,
3. Sort the set of trees $\{T_1, T_2, \ldots, T_S\}$ so that
$P(T_1|D) \leq P(T_2|D) \ldots \leq P(T_S|D)$
4. Set the weights for the trees T_i, $w(T_i) = w_i$.

In an ideal situation, doing Bayesian averaging involves integration of the posteriors over the whole model space, so $P(y|x) = \int_{\theta \in \Theta} P(y|\theta, x) P(\theta|D) d\theta$, where $P(\theta|D)$ is the posterior probability derived from the training data D. However, there are two obstacles in Bayesian averaging. The first is that when the model space is huge, such as when the models are decision trees, performing a whole integration is impractical. Another obstacle is that the posterior probabilities of inferred decision trees are difficult to calculate, due to unknown marginal probabilities of data x.

An MML decision tree inference scheme results in an optimal discretized tree space where the relative posterior probabilities of two inferred trees is given by comparing the MML two-part message lengths. The proposed weighted averaging scheme above in effect "smooths out" the weights of trees, making the weights of sampled trees vary more slowly than the rate suggested by the MML two-part message length. The motivation of such an algorithm is to compensate the bias introduced by the tree inference algorithm. As our MML tree inference algorithm prefers trees with high predictive accuracy, such a process of growing an ensemble of decision trees can be regarded as performing a form of importance sampling

in the posterior space. The integration becomes
$P(y|x) = \int_{\theta \in \Theta} P(y|\theta, x) \frac{P(\theta|D)}{G(\theta)} G(\theta) d\theta$, where $\frac{P(\theta|D)}{G(\theta)} \propto 2^{-msglen}$, where $msglen$ is the two-part message length of the tree model and data (although, for our predictive purposes, it might have been slightly better to use [29, sec. 3.1.7] I_0 rather than I_1 in the leaf nodes). In this paper, $G(\theta)$ is a Gaussian function which takes the two-part message length of the trees as input. Our final goal is to devise an MML inference algorithm to find a model class and inferred parameters for the optimal $G(\theta)$ (using training data). In this way, with limited number of inferred decision trees we are able to approximate the integration of the posteriors more accurately in the whole model space.

4 Experimental Results

4.1 Data Sets

We ran our experiments on 13 data sets from the UCI data repository [1], 12 of which are from Breiman's random forest paper [5]. We added BREAST-WINS as an additional two-class problem. For each data set, 10 independent 10-fold cross-validation tests were performed. A summary of the data sets is shown in table 1.

Table 1. Summary of the data sets.

Data set	Size	Continuous Atb.	Discrete Atb.	Classes
Breast	286	0	9	2
Breast-Wins	699	0	9	2
Bupa	345	6	0	2
Cleveland	303	5	8	2
Ecoli	336	7	0	8
German	1000	3	21	2
Glass	214	9	0	6
Image	2310	19	0	7
Ionosphere	351	34	0	2
Pima	768	8	0	2
Sat-Images	6435	36	0	6
Sonar	208	60	0	2
Vowel	990	10	0	11

4.2 Comparing and scoring probabilistic predictions

In addition to the traditional right/wrong accuracy, we are also keen to compare the probabilistic performance [27, sec 5.1] [14, 12, 21, 26] of the ensemble algorithms. In many domains, like oncological and other medical data, not only the class predictions but also the probability associated with each class is essential. In some domains, like finance, strategies heavily rely on accurate probabilistic predictions. To compare probabilistic prediction performances, we use a metric

called probabilistic costing (or log loss), defined as $P_{cost} = -\sum_{i=1}^{N} \log(p_i)$, where N is the total number of test data and p_i is the probability assigned by the model to the true (correct) class associated with test instance i. The probability p_j of an instance belonging to class j in a leaf node was estimated by $\hat{p_j} = \frac{N_j+1}{(\sum N_j)+M}$, where N_j is the number of instances belonging to the class j in the leaf node, M is the number of classes. The probabilistic costing is equivalent to the accumulated code length of the total test data encoded by an inferred model. As the optimal code length can only be achieved by encoding with the real probability distribution of the test data, it is obvious that a smaller probabilistic costing indicates a better performance on overall probabilistic prediction.

Although we don't do this here, Dowe showed (in detailed and explicit private communication after a paper at the Australian AI'2002 conference) how to modify this to include a Bayesian prior on the multinomial states by simply subtracting (a multiple of) the entropy of the prior (plus an optional but unnecessary constant).

And, rather than just numerically calculate the log-loss probabilistic bit costing on test data, if we knew the true underlying model and wished to infer a probabilistic Bayesian (or causal) network [7, 8], we could quite easily calculate a Kullback-Leibler distance between the true model and such an inferred model. We would do this by looking at each combination of states (or variable values at each node, or "micro-state") in turn, looking at the Kullback-Leibler distance between the true probability and the inferred probability for each such micro-state, and then summing these Kullback-Leibler distances weighted by the true probabilities of each micro-state in turn. (These weights add up to 1.)

4.3 Comparisons with Other Ensemble Algorithms

We also compare results from our ensemble learning schemes with two other prominent ensemble learning algorithms - AdaBoost and Random Forest - for which we also run 10 independent 10-fold cross-validations tests (averaging over 10x10=100 runs) on the 13 data sets. The random forests algorithm was implemented by weka3.4.6 [33]. At each test, a random forest with 1000 decision trees, whose internal node contains a linear combination of 2 input attributes attributes, was grown. C5.0 is a commercial version of the C4.5 [24] program. To obtain the results from AdaBoost, we ran our tests by using the built-in AdaBoost function of C5.0 with a maximum of 1000 iterations or until convergence. In most cases, AdaBoost finished before 1000 runs due to diminutive or no gain in the subsequent runs. The results for MML forests are returned by forests with 1000 trees. MML oblique trees are averaged by weighted averaging of class probabilities, recalling section 3.4.

The results for the "Right"/"Wrong" classification accuracies are shown in table 2. For the 8 data sets with binary classes (asterisked, recalling Table 1), in terms of "right/wrong" accuracy, the MML forests perform best on 5 out of 8 sets, second best on 1 and worst on 2; and on logarithm of probability (P_{cost})

Table 2. "Right"/"Wrong" classification accuracies and probabilistic costing results for forests with MML oblique trees, AdaBoost and random forests. [An asterisk (*) denotes that the target attribute of the data set is binary.]

Data set	MML Forest "R/W"	C5 AdaBoost "R/W"	Random Forest "R/W"	MML Forest P_{cost}	C5 AdaBoost P_{cost}	Random Forest P_{cost}
Breast*	73.4	72.9	71.0	23.4	23.6	N/A
Breast-Wins*	97.4	96.3	96.7	10.2	18.0	N/A
Bupa*	67.9	68.5	73.0	30.9	30.4	N/A
Cleveland*	83.4	80.6	82.5	17.7	19.0	N/A
Ecoli	85.9	84.0	86.2	26.4	45.6	N/A
German*	74.1	75.4	77.1	74.7	74.5	N/A
Glass	67.1	74.9	80.8	28.9	30.3	N/A
Image	92.0	97.9	98.1	156.3	176.5	N/A
Ionosphere*	93.8	93.6	92.8	12.5	12.5	N/A
Pima*	76.2	75.3	75.4	53.8	56.2	N/A
Sat-Images	83.3	90.7	91.5	411.4	574.3	N/A
Sonar*	81.9	80.8	84.0	12.5	13.3	N/A
Vowel	64.3	89.9	97.0	217.7	176.5	N/A

bit costing (recall sec. 4.2), the MML forests win 5, tie on 1 and lose the other 2. The results on the 5 multiple-class data sets are interesting. In "right/wrong" scoring, random forests win 4 out of 5 and come second in 1 out of 5. But in logarithm of probability scoring, MML forests win 4 out of 5 cases. The most probable explanation is that the base learner, MML oblique decision trees [28], does not perform well in "right/wrong" scoring on data sets with multiple classes. One way to improve the performance on data sets with multiple classes might possibly be to convert a multiple class learning problem into a set of binary class learning problems, another would be to implement ideas from sec. 3.4.

As mentioned in section 4.2, we are keen to compare the performance on probabilistic predictions of ensemble learning algorithms. To obtain P_{cost} for C5.0 AdaBoost, the probabilistic prediction for each test instance is calculated by arithmetically averaging the probabilistic predictions submitted by each iteration. Unfortunately, we are unable to obtain probabilistic costing for random forests from weka [33]. Table 2 shows that MML forests have achieved better (lower) or equal probabilistic costing in 9 out of 13 data sets compared to C5.0 AdaBoost. The superior performance on probabilistic prediction of the MML forests can be attributed to the fact that both the base learner algorithm and the averaging scheme are well-suited to probabilistic predictions.

5 Conclusions

An ensemble classifier using shallow oblique decision trees, our new ensemble learning algorithm achieves favourable results on data sets with binary classes.

Our novel random decision tree generating scheme is capable of constructing a decision forest with a large number of distinct highly performing decision trees. The proposed weighted averaging scheme exploits the potential of using Bayesian weighted averaging to improve the predictive accuracy of ensemble classifiers, especially on probabilistic predictions and any predictions involving binary output classes. It is reasonable to believe that replacing the preliminary model with well developed and more elaborate models (such as mixtures of Normal distributions or other distributions) to approximate the posterior probabilities of the inferred trees can only further enhance the results from this kind of weighted averaging. The significance of the results could be improved by using advice from [9] and our own ideas from sec. 3.4 - including sampling from the posterior (via an importance sampling distribution) and generalising the prior.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
2. L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–824, Jun. 1998.
3. L. Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learing*, 40:229–242, 2000.
4. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. Leo Breiman. Random forests. *Machine Learning*, 45(1):5, 2001.
6. Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification And Regression Trees*. Wadsworth & Brooks, 1984.
7. Joshua W. Comley and David L. Dowe. Generalised Bayesian networks and asymmetric languages. In *Proc. Hawaii International Conference on Statistics and Related Fields*, 5-8 June 2003.
8. Joshua W. Comley and David L. Dowe. Chapter 11, Minimum Message Length and generalized Bayesian networks with asymmetric languages. In P. Grünwald, M. A. Pitt, and I. J. Myung, editors, *Advances in Minimum Description Length: Theory and Applications*, pages 265–294. M.I.T. Press, Apr 2005. Final camera-ready copy submitted Oct. 2003.
9. Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, January 2006.
10. Thomas G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
11. Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.
12. D. L. Dowe, G.E. Farr, A.J. Hurst, and K.L. Lentin. Information-theoretic football tipping. In N. de Mestre, editor, *Third Australian Conference on Mathematics and Computers in Sport*, pages 233–241. Bond University, Qld, Australia, 1996. http://www.csse.monash.edu.au/∼footy.
13. D. L. Dowe, S. Gardner, and G. R. Oppy. Bayes not Bust! Why simplicity is no problem for Bayesians. *British Journal for the Philosophy of Science*, forthcoming.
14. D. L. Dowe and N. Krusel. A decision tree model of bushfire activity. In *(Technical report 93/190) Dept. Comp. Sci., Monash Uni., Clayton, Australia*, 1993.

15. D. L. Dowe and C. S. Wallace. Kolmogorov complexity, minimum message length and inverse learning. In *14th Australian Statistical Conference (ASC-14)*, page 144, Gold Coast, Qld, Australia, 6-10 July 1998.
16. Cesar Ferri, Peter Flach, and Jose Hernandez-Orallo. Delegating classifiers. In *Proc. 21st International Conference on Machine Learning*, pages 106–110, Banff, Canada, 2004.
17. Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1996.
18. Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998.
19. Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-based Decision Tree Pruning. In *The First International Conference on Knowledge Discovery & Data Mining*, pages 216–221. AAAI Press, 1995.
20. Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Journal of Information Fusion (Special Issue on Diversity in Multiple Classifier Systems)*, 6(1):99–111, 2004.
21. S. L. Needham and D. L. Dowe. Message length as an effective Ockham's razor in decision tree induction. In *Proc. 8th International Workshop on Artificial Intelligence and Statistics*, pages 253–260, Key West, Florida, U.S.A., Jan. 2001.
22. J. J. Oliver and C. S. Wallace. Inferring Decision Graphs. In *Workshop 8 International Joint Conference on AI (IJCAI)*, Sydney, Australia, August 1991.
23. Jonathan J. Oliver and David J. Hand. On pruning and averaging decision trees. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 430–437. Morgan Kaufmann, 1995.
24. J.R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, U.S.A., 1992. The latest version of C5 is available from http://www.rulequest.com.
25. J.J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
26. P. J. Tan and D. L. Dowe. MML inference of decision graphs with multi-way joins. In *Lecture Notes in Artificial Intelligence (LNAI) 2557 (Springer), Proc. 15th Australian Joint Conf. on AI*, pages 131–142, Canberra, Australia, 2-6 Dec. 2002.
27. P. J. Tan and D. L. Dowe. MML inference of decision graphs with multi-way joins and dynamic attributes. In *Lecture Notes in Artificial Intelligence (LNAI) 2903 (Springer), Proc. 16th Australian Joint Conf. on AI*, pages 269–281, Perth, Australia, Dec. 2003.
28. P. J. Tan and D. L. Dowe. MML inference of oblique decision trees. In *Lecture Notes in Artificial Intelligence (LNAI) 3339 (Springer), Proc. 17th Australian Joint Conf. on AI*, pages 1082–1088, Cairns, Australia, Dec. 2004.
29. C. S. Wallace. *Statistical and Inductive Inference by Minimum Message Length*. Springer, 2005.
30. C. S. Wallace and D. M. Boulton. An Information Measure for Classification. *Computer Journal*, 11:185–194, 1968.
31. C. S. Wallace and D. L. Dowe. Minimum Message Length and Kolmogorov Complexity. *Computer Journal*, 42(4):270–283, 1999.
32. C. S. Wallace and P. R. Freeman. Estimation and Inference by Compact Coding. *Journal of the Royal Statistical Society. Series B*, 49(3):240–265, 1987.
33. Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.