Machine Learning, , 1–14 ()

Technical Note

Using Model Trees for Classification

EIBE FRANK eibe@cs.waikato.ac.nz
YONG WANG yongwang@cs.waikato.ac.nz
STUART INGLIS singlis@cs.waikato.ac.nz
GEOFFREY HOLMES geoff@cs.waikato.ac.nz
IAN H. WITTEN ihw@cs.waikato.ac.nz
Department of Computer Science, University of Waikato, Hamilton, New Zealand

Received Sep 9, 1997; Accepted Nov 11, 1997; Final Manuscript Nov 24, 1997

Editor: Raymond Mooney

Abstract. Model trees, which are a type of decision tree with linear regression functions at the leaves, form the basis of a recent successful technique for predicting continuous numeric values. They can be applied to classification problems by employing a standard method of transforming a classification problem into a problem of function approximation. Surprisingly, using this simple transformation the model tree inducer M5′, based on Quinlan's M5, generates more accurate classifiers than the state-of-the-art decision tree learner C5.0, particularly when most of the attributes are numeric.

Keywords: Model trees, classification algorithms, M5, C5.0, decision trees

1. Introduction

Many applications of machine learning in practice involve predicting a "class" that takes on a continuous numeric value, and the technique of model tree induction has proved successful in addressing such problems (Quinlan, 1992; Wang and Witten, 1997). Structurally, a model tree takes the form of a decision tree with linear regression functions instead of terminal class values at its leaves. Numerically-valued attributes play a natural role in these regression functions, while discrete attributes can also be handled—though in a less natural way. This is the converse of the classical decision-tree situation for classification, where discrete attributes play a natural role. Prompted by the symmetry of this situation, we wondered whether model trees could be used for classification. We have discovered that they can be turned into classifiers that are surprisingly accurate.

In order to apply the continuous-prediction technique of model trees to discrete classification problems, we consider the conditional class probability function and seek a model-tree approximation to it. During classification, the class whose model tree generates the greatest approximated probability value is chosen as the predicted class.

The results presented in this paper show that a model tree inducer can be used to generate classifiers that are significantly more accurate than the decision trees

produced by C5.0.[1] The next section explains the method we use and reviews the features that are responsible for its good performance. Experimental results for thirty-three standard datasets are reported in Section 3. Section 4 briefly reviews related work. Section 5 summarizes the results.

2. Applying model trees to classification

Model trees are binary decision trees with linear regression functions at the leaf nodes: thus they can represent any piecewise linear approximation to an unknown function. A model tree is generated in two stages. The first builds an ordinary decision tree, using as splitting criterion the maximization of the intra-subset variation of the target value. The second prunes this tree back by replacing subtrees with linear regression functions wherever this seems appropriate. Whenever the model is used for prediction a smoothing process is invoked to compensate for the sharp discontinuities that will inevitably occur between adjacent linear models at the leaves of the pruned tree. Although the original formulation of model trees had linear models at internal nodes that were used during the smoothing process, these can be incorporated into the leaf models in the manner described below.

In this section we first describe salient aspects of the model tree algorithm. Then we describe the procedure, new to this paper, by which model trees are used for classification. Some justification for this procedure is given in the next subsection, following which we give an example of the inferred class probabilities in an artificial situation in which the true probabilities are known.

2.1. Model-tree algorithm

The construction and use of model trees is clearly described in Quinlan's (1992) account of the M5 scheme. An implementation called M5′ is described by Wang and Witten (1997) along with further implementation details. The freely available version[2] of M5′ we used for this paper differs from that described by Wang and Witten (1997) only in its improved handling of missing values, which we describe in the appendix.[3] There were no other changes, and no tuning of parameters.

It is necessary to elaborate briefly on two key aspects of model trees that will surface during the discussion of experimental results in Section 3. The first, which is central to the idea of model trees, is the linear regression step that is performed at the leaves of the pruned tree. The variables involved in the regression are the attributes that participated in decisions at nodes of the subtree that has been pruned away. If this step is omitted and the target is taken to be the average target value of training examples that reach this leaf, then the tree is called a "regression tree" instead.

The second aspect is the smoothing procedure that, in the original formulation, occurred whenever the model was used for prediction. The idea is first to use the leaf model to compute the predicted value, and then to filter that value along the path back to the root, smoothing it at each node by combining it with the value predicted by the linear model for that node. Quinlan's (1992) calculation is

$$p' = \frac{np + kq}{n + k} \quad , \tag{1}$$

where p' is the prediction passed up to the next higher node, p is the prediction passed to this node from below, q is the value predicted by the model at this node, n is the number of training instances that reach the node below, and k is a constant. Quinlan's default value of $k = 15$ was used in all experiments below.

Our implementation achieves exactly the same effect using a slightly different representation. As a final stage of model formation we create a new linear model at each leaf that combines the linear models along the path back to the root, so that the leaf models automatically create smoothed predictions without any need for further adjustment when predictions are made. For example, suppose the model at a leaf involved two attributes x and y, with linear coefficients a and b; and the model at the parent node involved two attributes y and z:

$$p = ax + by \qquad q = cy + dz. \tag{2}$$

We combine these two models into a single one using the above formula:

$$p' = \frac{na}{n + k}x + \frac{nb + kc}{n + k}y + \frac{kd}{n + k}z. \tag{3}$$

Continuing in this way up to the root gives us a single, smoothed linear model which we install at the leaf and use for prediction thereafter.

Smoothing substantially enhances the performance of model trees, and it turns out that this applies equally to their application to classification.

2.2. Procedure

Figure 1 shows in diagrammatic form how a model tree builder is used for classification; the data is taken from the well-known Iris dataset. The upper part depicts the training process and the lower part the testing process.

Training starts by deriving several new data sets from the original dataset, one for each possible value of the class. In this case there are three derived datasets, for the *Setosa*, *Virginica* and *Versicolor* varieties of Iris. Each derived dataset contains the same number of instances as the original, with the class value set to 1 or 0 depending on whether that instance has the appropriate class or not. In the next step the model tree inducer is employed to generate a model tree for each of the new datasets. For a specific instance, the output of one of these model trees constitutes an approximation to the probability that this instance belongs to the associated class. Since the output values of the model trees are only approximations, they do not necessarily sum to one.

In the testing process, an instance of unknown class is processed by each of the model trees, the result of each being an approximation to the probability that it belongs to that class. The class whose model tree gives the highest value is chosen as the predicted class.

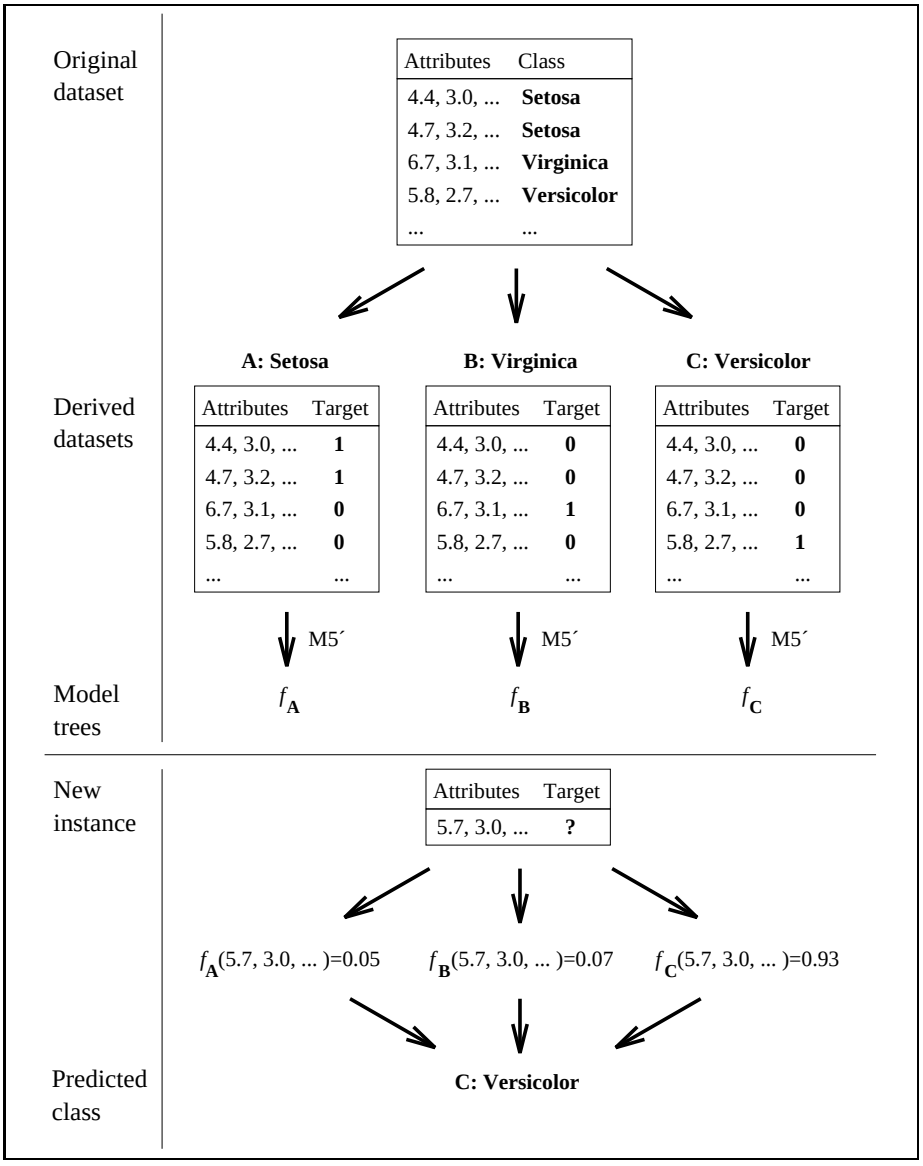

Figure 1. How M5′ is used for classification

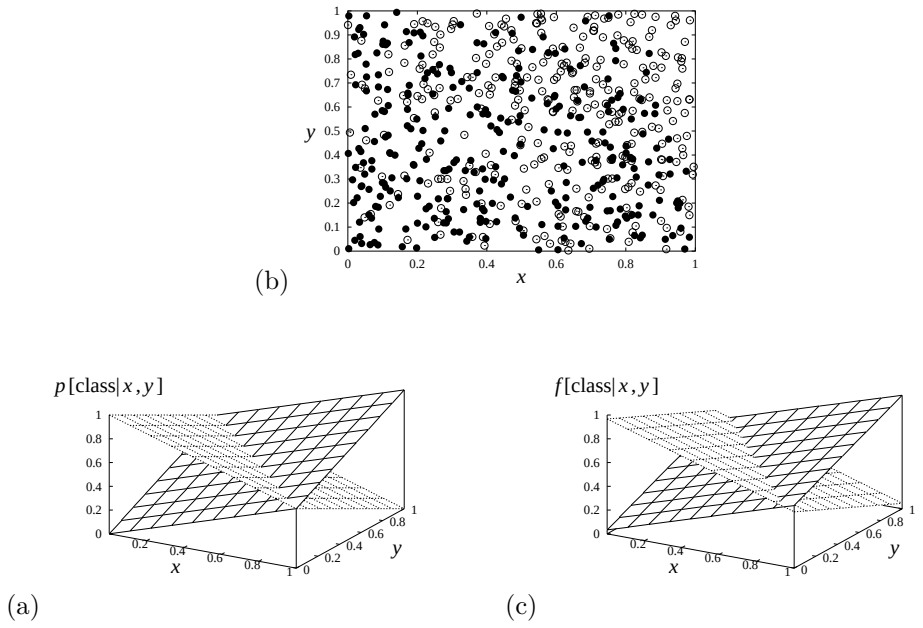

Figure 2. Example use of model trees for classification: (a) class probabilities for data generation; (b) the training dataset; (c) inferred class probabilities

2.3. Justification

The learning procedure of M5′ effectively divides the instance space into regions using a decision tree, and strives to minimize the expected mean squared error between the model tree's output and the target values of zero and one for the training instances within each particular region. The training instances that lie in a particular region can be viewed as samples from an underlying probability distribution that assigns class values of zero and one to instances within that region. It is standard procedure in statistics to estimate a probability distribution by minimizing the mean square error of samples taken from it (Devroye, Gyoerfi and Lugosi, 1996; Breiman, Friedman, Olshen and Stone, 1984).

2.4. Example

Consider a two-class problem in which the true class probabilities are linear functions of two attributes x and y, $p[\text{class}|x, y]$, as depicted in Figure 2a, summing to 1 at each point. A dataset with 600 instances is generated randomly according to these probabilities. To do this, uniformly distributed (x, y) values are chosen and the probability at that (x, y) value is used to determine whether the instance should be assigned to the first or the second class. The data generated is depicted

in Figure 2b, where the classes are represented by filled and hollow circles. It is apparent that the density of filled circles is greatest at the lower left corner and decreases towards the upper right corner; the converse is true for hollow circles.

When the data of Figure 2b is submitted to M5′ it generates two model trees. In this case the structure of the trees generated is trivial—they each consist of a single node, the root. Figure 2c shows the linear functions $f[\text{class}|x, y]$ represented by the trees. As the above discussion intimates, they are excellent approximations to the original class probabilities from which the data was generated.

The class boundary is the point of intersection of the two planes in Figure 2c, and as this example illustrates, classifiers based on model trees are able to represent oblique class boundaries. This is one reason why model trees produced by M5′ outperform the univariate decision trees produced by C5.0. Another is that M5′ smooths between regression functions at adjacent leaves of the model tree.

3. Experimental results

Our experiments are designed to explore the application of model trees to classification by comparing their results with decision tree induction and linear regression, and determining which of their components are essential for good performance. Specifically, we address the following questions:

1. How do classifiers based on model trees compare to state-of-the-art decision trees, and to classifiers based on simple linear regression?

2. How important are (a) the linear regression process at the leaves, and (b) the smoothing process?

To answer the first question, we compare the accuracy of classifiers based on the smoothed model trees generated by M5′ with the pruned decision trees generated by C5.0: we will see that M5′ often performs better. However, the performance improvement might conceivably be due to other aspects of the procedure: M5′ converts a nominal attribute with n attribute values into $n - 1$ binary attributes using the procedure employed by CART (Breiman *et al.*, 1984), and it generates one model tree for each class. To test this we ran C5.0 using exactly the same encodings, transforming each nominal attribute into binary ones using the procedure employed by M5′ and generating one dataset for each class, and then building a decision tree for each dataset and using the class probabilities provided by C5.0 to arbitrate between the classes. We refer to the resulting algorithm as C5.0′. We also report results for linear regression (LR) using the same input/output encoding.

To investigate the second question, we first compare the accuracy of classifiers based on model trees that are generated by M5′ with ones based on smoothed *regression* trees (SRT). As noted above, regression trees are model trees with constant functions at the leaf nodes; thus they cannot represent oblique class boundaries. We apply the same smoothing operation to them as M5′ routinely applies to model trees. Then we compare the accuracy of classifiers based on the (smoothed) model trees of M5′ with those based on *unsmoothed* model trees (UMT). Because

Table 1. Datasets used for the experiments

Dataset	Instances	Missing values (%)	Numeric attributes	Binary attributes	Nominal attributes	Classes
balance-scale	625	0.0	4	0	0	3
breast-w	699	0.3	9	0	0	2
glass (G2)	163	0.0	9	0	0	2
glass	214	0.0	9	0	0	6
heart-statlog	270	0.0	13	0	0	2
hepatitis	155	5.6	6	13	0	2
ionosphere	351	0.0	33	1	0	2
iris	150	0.0	4	0	0	3
letter	20000	0.0	16	0	0	26
pima-indians	768	0.0	8	0	0	2
segment	2310	0.0	19	0	0	7
sonar	208	0.0	60	0	0	2
vehicle	846	0.0	18	0	0	4
vote	435	5.6	0	16	0	2
waveform-noise	5000	0.0	40	0	0	3
zoo	101	0.0	1	15	0	7
anneal	898	0.0	6	14	18	5
audiology	226	2.0	0	61	8	24
australian	690	0.6	6	4	5	2
autos	205	1.1	15	4	6	6
breast-cancer	286	0.3	0	3	6	2
heart-c	303	0.2	6	3	4	2
heart-h	294	20.4	6	3	4	2
horse-colic	368	23.8	7	2	13	2
hypothyroid	3772	5.5	7	20	2	4
german	1000	0.0	6	3	11	2
kr-vs-kp	3196	0.0	0	35	1	2
labor	57	3.9	8	3	5	2
lymphography	148	0.0	3	9	6	4
primary-tumor	339	3.9	0	14	3	21
sick	3772	5.5	7	20	2	2
soybean	683	9.8	0	16	19	19
vowel	990	0.0	10	2	1	11

a smoothed regression tree is a special case of a smoothed model tree, and an unsmoothed tree is a special case of a smoothed tree, only very minor modifications to the code for M5′ are needed to generate SRT and UMT models.

3.1. Experiments

Thirty-three standard datasets from the UCI collection (Merz and Murphy, 1996) were used in the experiments: they are summarized in Table 1. The first sixteen involve only numeric and binary attributes; the last seventeen involve non-binary nominal attributes as well.[4] Since linear regression functions were designed for numerically-valued domains, and binary attributes are a special case of numeric

attributes, we expect classifiers based on smoothed model trees to be particularly appropriate for the first group.

Table 2 summarizes the accuracy of all methods investigated. Results give the percentage of correct classifications, averaged over ten ten-fold (non-stratified) cross-validation runs, and standard deviations of the ten are also shown. The same folds were used for each scheme. Results for C5.0 are starred if they show significant improvement over the corresponding result for M5′, and *vice versa*. Throughout, we speak of results being "significantly different" if the difference is statistically significant at the 1% level according to a paired two-sided t-test, each pair of data points consisting of the estimates obtained in one ten-fold cross-validation run for the two learning schemes being compared.

Table 3 shows how the different methods compare with each other. Each entry indicates the number of datasets for which the method associated with its column was significantly more accurate than the method associated with its row.

3.2. Discussion of results

To answer the first question above, we observe from Table 2 that M5′ outperforms C5.0 in fifteen datasets, whereas C5.0 outperforms M5′ in five. (These numbers also appear, in boldface, in Table 3.) Of the sixteen datasets having numeric and binary attributes, M5′ is significantly more accurate on nine and significantly less accurate on none; on the remaining datasets it is significantly more accurate on six and significantly less accurate on five. These results show that classifiers based on the smoothed model trees generated by M5′ are significantly more accurate than the pruned decision trees generated by C5.0 on the majority of datasets, particularly those with numeric attributes.

Table 3 shows that C5.0′ is significantly less accurate than C5.0 on twelve datasets (first column, last row) and significantly more accurate on five (first row, last column). It is significantly less accurate than M5′ on seventeen datasets and significantly more accurate on three. These results show that the superior performance of M5′ is not due to the change in input/output encoding.

We complete our discussion of the first question by comparing simple linear regression (LR) to M5′ and C5.0. Table 3 shows that LR performs significantly worse than M5′ on seventeen datasets and significantly worse than C5.0 on eighteen. LR outperforms M5′ on eleven datasets and C5.0 on fourteen. These results for linear regression are surprisingly good. However, on some of the datasets the application of linear regression leads to disastrous results and so one cannot recommend this as a general technique.

To answer the second of the above two questions, we begin by comparing the accuracy of classifiers based on M5′ with ones based on smoothed regression trees (SRT) to assess the importance of the linear regression process at the leaves (which the former incorporates but the latter does not). Table 3 shows that M5′ produces significantly more accurate classifiers on twenty-three datasets and significantly less accurate ones on only two. Compared to C5.0's pruned decision trees, classifiers based on smoothed regression trees are significantly less accurate on fifteen datasets

Table 2. Experimental results: percentage of correct classifications, and standard deviation

Dataset	C5.0	M5′	LR	M5′ SRT	M5′ UMT	C5.0′
balance-scale	77.6±1.0	86.4±0.7*	86.7±0.3	75.3±1.1	78.8±0.9	78.9±0.7
breast-w	94.5±0.3	95.3±0.3*	95.8±0.1	94.3±0.5	94.2±0.4	94.5±0.3
glass (G2)	78.7±2.1	81.8±2.2	70.4±0.4	75.5±1.7	79.3±2.3	78.8±2.2
glass	67.5±2.6	70.5±2.8	60.0±1.3	67.6±1.6	67.8±2.7	70.0±2.0
heart-statlog	78.7±1.4	82.2±1.0*	83.7±0.4	79.9±1.8	78.4±1.5	78.6±1.4
hepatitis	79.3±1.2	81.9±2.2*	85.6±1.5	79.6±1.5	78.8±3.0	79.7±1.1
ionosphere	88.9±1.6	89.7±1.2	86.6±0.5	88.2±0.7	87.3±1.0	88.9±1.6
iris	94.5±0.7	94.7±0.7	82.7±0.9	94.0±1.0	93.9±0.8	94.7±0.7
letter	88.0±0.2	90.3±0.1*	55.5±0.1	86.3±0.2	86.7±0.1	87.5±0.1
pima-indians	74.5±1.2	76.2±0.8*	77.2±0.5	75.7±1.0	72.0±0.7	74.5±1.2
segment	96.8±0.2	97.0±0.2	84.5±0.1	96.2±0.2	95.9±0.3	95.7±0.2
sonar	74.7±2.8	78.5±3.4*	75.6±1.8	78.0±2.4	75.8±2.7	74.7±2.8
vehicle	72.9±1.2	76.5±1.3*	75.7±0.5	70.9±1.2	69.3±1.2	72.0±1.0
vote	96.3±0.6	96.2±0.3	95.6±0.0	95.6±0.0	95.9±0.5	96.4±0.5
waveform-noise	75.4±0.5	82.0±0.2*	85.9±0.2	80.3±0.3	72.3±0.4	75.2±0.5
zoo	91.8±1.1	92.1±1.3	94.2±1.8	89.3±1.5	90.5±1.3	89.1±1.4
anneal	98.7±0.3	98.8±0.2	93.1±0.2	97.3±0.1	98.5±0.2	99.0±0.2
audiology	76.5±1.4	76.7±1.0	68.6±1.6	67.9±1.2	76.8±1.8	73.9±0.9
australian	85.3±0.5	85.8±0.9	51.1±3.6	85.7±0.7	82.8±0.9	83.8±1.1
autos	80.0±2.5*	74.4±1.9	59.0±1.5	70.0±2.2	71.7±1.8	75.6±1.7
breast-cancer	73.3±1.6*	69.6±2.3	70.0±1.5	72.9±1.0	67.5±2.4	68.8±1.7
heart-c	76.8±1.4	80.9±1.4*	85.0±0.4	79.7±1.6	76.3±1.3	78.8±1.6
heart-h	79.8±0.9	79.0±0.8	81.9±1.0	79.2±1.1	76.9±1.5	77.5±1.3
horse-colic	85.3±0.6	84.6±0.7	82.7±0.7	84.5±0.9	83.4±1.5	84.5±0.6
hypothyroid	99.5±0.0*	96.6±0.1	90.9±3.1	95.6±0.1	96.2±0.2	99.4±0.1
german	71.2±1.0	72.9±0.7*	75.4±0.6	74.1±0.9	69.9±0.8	71.6±1.4
kr-vs-kp	99.5±0.1*	99.4±0.1	94.0±0.1	98.5±0.1	99.3±0.1	99.4±0.1
labor	78.1±4.8	79.7±4.6	87.4±6.1	71.4±3.6	77.9±3.6	76.8±4.5
lymphography	75.4±2.8	79.8±1.4*	83.6±1.3	76.1±1.6	77.5±2.9	75.9±2.2
primary-tumor	41.8±1.3	45.1±1.6*	47.2±0.9	45.1±1.3	41.4±1.2	40.3±2.1
sick	98.8±0.1*	98.3±0.1	92.3±2.5	98.2±0.0	98.6±0.1	98.9±0.1
soybean	91.3±0.5	92.5±0.5*	87.3±0.6	88.4±0.5	91.3±0.5	92.3±0.5
vowel	79.8±1.3	81.7±1.1*	43.1±1.0	73.9±1.5	78.3±0.8	78.1±1.0

and significantly more accurate on five. These results show that linear regression functions at leaf nodes are essential for classifiers based on smoothed model trees to outperform ordinary decision trees.

Finally, to complete the second question, we compare the accuracy of classifiers based on M5′ with classifiers based on unsmoothed model trees (UMT). Table 3 shows that M5′ produces significantly more accurate classifiers on twenty-five datasets and significantly less accurate classifiers on only one. Comparison with C5.0's pruned decision trees also leads to the conclusion that the smoothing process is necessary to ensure high accuracy of model-tree based classifiers.

Table 3. Results of paired t-tests (p=0.01): number indicates how often method in column significantly outperforms method in row

	C5.0	M5′	LR	M5′ SRT	M5′ UMT	C5.0′
C5.0	–	**15**	14	5	0	5
M5′	**5**	–	11	2	1	3
LR	18	17	–	16	14	17
M5′ SRT	15	23	14	–	11	12
M5′ UMT	14	25	14	12	–	12
C5.0′	12	17	14	8	1	–

4. Related work

Neural networks are an obvious alternative to model trees for classification tasks. When applying neural networks to classification it is standard procedure to approximate the conditional class probability functions. Each output node of a neural network approximates the probability function of one class. In contrast to neural networks where the probability functions for all classes are approximated by a single network, with model trees it is necessary to build a separate tree for each class. Model trees offer an advantage over neural networks in that the user does not have to make guesses about their structure and size to obtain accurate results. They can be built fully automatically and much more efficiently than neural networks. Moreover, they offer opportunities for structural analysis of the approximated class probability functions, whereas neural networks are completely opaque.

The idea of treating a multi-class problem as several two-way classification problems, one for each possible value of the class, has been been applied to standard decision trees by Dietterich and Bakiri (1995). They used C4.5 (Quinlan, 1993), the predecessor of C5.0, to generate a two-way classification tree for each class. However, they found that the accuracy obtained was significantly inferior to the direct application of C4.5 to the original multi-class problem—although they were able to obtain better results by using an error-correcting output code instead of the simple one-per-class code.

Smyth, Gray and Fayyad (1995) retrofitted a decision tree classifier with kernel density estimators at the leaves in order to obtain better estimates of the class probability functions. Although this did improve the accuracy of the class probability estimates on three artificial datasets, the classification accuracies were not significantly better. Moreover, the resulting structure is opaque because it includes a kernel function for every training instance. Torgo (1997) also investigated fitting trees with kernel estimators at the leaves, this time regression trees rather than classification trees. These could be applied to classification problems in the same manner as model trees, and have the advantage of being able to represent non-linear class boundaries rather than the linear, oblique, class boundaries of model trees. However, they suffer from the incomprehensibility of all models that employ kernel estimators. An important difference between both Smyth *et al.* (1995) and Torgo (1997), and the M5 model tree algorithm, is that M5 smooths between the models

at adjacent leaves of the model tree. This substantially improves the performance of model trees in classification problems, as we saw.

Also closely related to our method are linear regression and other methods for finding linear discriminants. On comparing our experimental results with those obtained by ordinary linear regression, we find that although for many datasets linear regression performs very well, in several other cases it gives disastrous results because linear models are simply not appropriate.

5. Conclusions

This work has shown that when classification problems are transformed into problems of function approximation in a standard way, they can be successfully solved by constructing model trees to produce an approximation to the conditional class probability function of each individual class. The classifiers so derived outperform a state-of-the-art decision tree learner on problems with numeric and binary attributes, and, more often than not, on problems with multivalued nominal attributes too.

Although the resulting classifiers are less comprehensible than decision trees, they are not as opaque as those produced by statistical kernel density approximators. The expected time taken to build a model tree is log-linear in the number of instances and cubic in the number of attributes. Thus model trees for each class can be built efficiently if the dataset has a modest number of attributes.

Acknowledgments

The Waikato Machine Learning group, supported by the New Zealand Foundation for Research, Science, and Technology, has provided a stimulating environment for this research. We thank the anonymous referees for their helpful and constructive comments. M. Zwitter and M. Soklic donated the lymphography and the primary-tumor dataset.

Appendix

Treatment of missing values

We now explain how instances with missing values are treated in the version of M5′ used for the results in this paper. During testing, whenever the decision tree calls for a test on an attribute whose value is unknown, the instance is propagated down both paths and the results are combined linearly in the standard way (as in Quinlan, 1993). The problem is how to deal with missing values during training.

To tackle this problem, Breiman *et al.* (1984) describe a "surrogate split" method in which, whenever a split on value v of attribute s is being considered and a particular instance has a missing value, a different attribute s^* is used as a surrogate to split on instead, at an appropriately chosen value v^*—that is, the test $s < v$ is

replaced by $s^* < v^*$. The attribute s^* and value v^* are selected to maximize the probability that the latter test has the same effect as the former.

For the work described in this paper, we have made two alterations to the procedure. The first is a simplification. Breiman's original procedure is as follows. Let $\mathcal{S}$ be the set of training instances at the node whose values for the splitting attribute s are known. Let $\mathcal{L}$ be that subset of $\mathcal{S}$ which the split $s < v$ assigns to the left branch, and $\mathcal{R}$ be the corresponding subset for the right branch. Define $\mathcal{L}_*$ and $\mathcal{R}_*$ in the same way for the surrogate split $s^* < v^*$. Then the number of instances in $\mathcal{S}$ that are correctly assigned to the left subnode by the surrogate split $s^* < v^*$ is $L = |\mathcal{L} \cap \mathcal{L}_*|$, and $R = |\mathcal{R} \cap \mathcal{R}_*|$ is the corresponding number for the right subnode. The probability that $s^* < v^*$ predicts $s < v$ correctly can be estimated as $(L + R)/|\mathcal{S}|$. v^* is chosen so that the surrogate split $s^* < v^*$ maximizes this estimate. Whereas Breiman chooses the attribute s^* and value v^* to maximize this estimate, our simplification is to always choose the surrogate attribute s^* to be the class (but to continue to select the optimal value v^* as described). This stratagem was reported in Wang and Witten (1997).

The second difference is to blur the sharp distinctions made by Breiman's procedure. According to the original procedure, a (training) instance whose value for attribute s is missing is assigned to the left or right subnode according to whether $s^* < v^*$ or not. This produces a sharp step-function discontinuity which is inappropriate in cases when $s^* < v^*$ is a poor predictor of $s < v$. Our modification, which is employed by the version of M5′ used in the present paper, is to soften the decision by making it stochastic according to the probability curve illustrated in Figure A.1. The steepness of the transition is determined by the likelihood of the test $s^* < v^*$ assigning an instance to the incorrect subnode, and this is assessed by considering the training instances for which the value of attribute s is known.

First we estimate the probability p_r that $s^* < v^*$ assigns an instance with a missing value of s to the rightmost subnode; the probability of it being assigned to the left node is just $1 - p_r$. The probability that an instance is *incorrectly* assigned to the *left* subnode by $s^* < v^*$ can be estimated as $p_{il} = 1 - L/|\mathcal{L}_*|$; likewise the probability that it is *correctly* assigned to the *right* subnode is $p_{cr} = R/|\mathcal{R}_*|$. Let m_l be the mean class value over the instances in $\mathcal{L}_*$, and m_r the corresponding value for $\mathcal{R}_*$. We estimate p_r by a model of the form

$$p_r = \frac{1}{1 + e^{-ax+b}},$$

where x is the class value, and a, b are chosen to make the curve pass through the points (m_l, p_{il}) and (m_r, p_{cr}) as shown in Figure A.1. This has the desired effect of approximating a sharp step function if $s^* < v^*$ is a good predictor of $s < v$, which is when $p_{il} \approx 0$ and $p_{cr} \approx 1$, or when the decision is unimportant, which is when $m_l \approx m_r$. However, if the prediction is unreliable—that is, when p_{il} is significantly greater than 0 or p_{cr} is significantly less than 1—the decision is softened, particularly if it is important—that is, when m_l and m_r differ appreciably.

During training, an instance is stochastically assigned to the right subnode with probability p_r. During testing, surrogate splitting cannot be used because the class

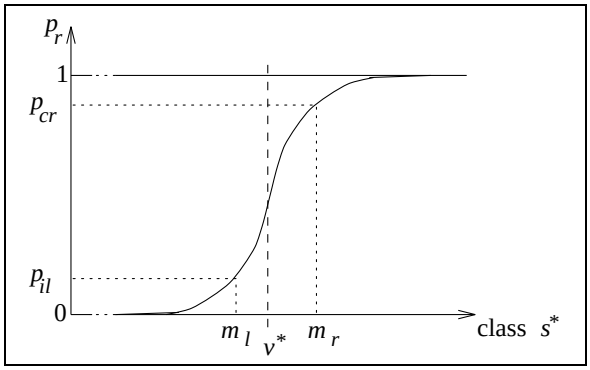

Figure A.1. How the soft step function model is fitted to the training data

value is, of course, unavailable. Instead an instance is propagated to both left and right subnodes, and the resulting outcomes are combined linearly using the weighting scheme described in Quinlan (1993): the left outcome is weighted by the proportion of training instances assigned to the left subnode, and the right outcome by the proportion assigned to the right subnode.

Notes

1. C5.0 is the successor of C4.5 (Quinlan, 1993). Although a commercial product, a test version is available from http://www.rulequest.com.
2. See http://www.cs.waikato.ac.nz/∼ml
3. For a realistic evaluation on standard datasets it is imperative that missing values are accommodated. If we removed instances with missing values, half the datasets in the lower part of Table 1 would have too few instances to be usable.
4. Following Holte (1993), the G2 variant of the *glass* dataset has classes 1 and 3 combined and classes 4 to 7 deleted, and the *horse-colic* dataset has attributes 3, 25, 26, 27, 28 deleted with attribute 24 being used as the class. We also deleted all identifier attributes in the datasets.

References

Breiman, L., Friedman, J.H., Olshen, R.A. and Stone, C.J. (1984). *Classification and regression trees.* Belmont CA: Wadsworth.

Devroye, L., Gyoerfi, L. and Lugosi, G. (1996). *A probabilistic theory of pattern recognition.* New York: Springer-Verlag.

Dietterich, T.G. and Bakiri G. (1995). "Solving multiclass learning problems via error-correcting output codes," *Journal of AI research*, 2, 263–286.

Holte, R.C. (1993). "Very simple classification rules perform well on most commonly used datasets," *Machine Learning*, 11, 63–91.

Merz, C.J. and Murphy, P.M. (1996). *UCI Repository of machine learning data-bases* [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Quinlan, J.R. (1992). "Learning with continuous classes," *Proceedings Australian Joint Conference on Artificial Intelligence* (pp. 343–348). World Scientific, Singapore.

Quinlan, J.R. (1993). *C4.5: Programs for machine learning.* Morgan Kaufmann.

Smyth, P., Gray, A. and Fayyad, U. (1995). "Retrofitting Decision tree classifiers using Kernel Density Estimation," *Proceedings International Conference on Machine Learning* (pp. 506–514). San Francisco, CA: Morgan Kaufmann.

Torgo, L. (1997). "Kernel Regression Trees," *Proceedings of the poster papers of the European Conference on Machine Learning.* University of Economics, Faculty of Informatics and Statistics, Prague.

Wang, Y. and Witten, I.H. (1997). "Induction of model trees for predicting continuous classes," *Proceedings of the poster papers of the European Conference on Machine Learning,* University of Economics, Faculty of Informatics and Statistics, Prague.