Working Paper Series
ISSN 1170-487X

Using Model Trees for Classification

by Eibe Frank, Yong Wang, Stuart Inglis, Geoffrey Holmes and Ian H Witten

Working Paper 97/12
April 1997

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

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

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 successor of C4.5 (Quinlan, 1993). The next section briefly reviews two strands of related work. Section 3 explains the method we use and reviews the features that are responsible for its good performance. Experimental

results for twenty standard datasets are reported in Section 4. Section 5 summarizes the results.

2. Related work

The method of approximating the conditional class probability functions is standard procedure when applying neural networks to classification problems. Each output node of a neural network approximates the probability function of one class. However, in contrast to neural networks, where the class probability functions for all classes are approximated by one network, it is necessary to build model trees for each class separately. 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 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.

3. 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. Finally, 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. The construction and use of model trees is clearly described in Quinlan's (1992) account of the M5 scheme. A freely available implementation called M5′ is described by Wang and Witten (1997) along with further implementation details.[2]

It is necessary to elaborate briefly on two key aspects of model trees that will surface during the discussion of experimental results in Section 4. 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. However, it is possible to modify the linear models at the leaves to incorporate the smoothing operation as a final stage of model formation, so that the modified linear models automatically create smoothed predictions without any need for further adjustment. Smoothing substantially enhances the performance of model trees, and it turns out that this applies equally to their application to classification.

3.1. 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. Naturally, the class whose model tree gives the highest value is chosen as the predicted class.

3.2. 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 a well-known result in the foundations of probability theory that minimizing the expected mean square error of these samples yields an estimate of the underlying true probability whose mean squared error is also minimized (Devroye *et al.*, 1996; Breiman *et al.*, 1984).

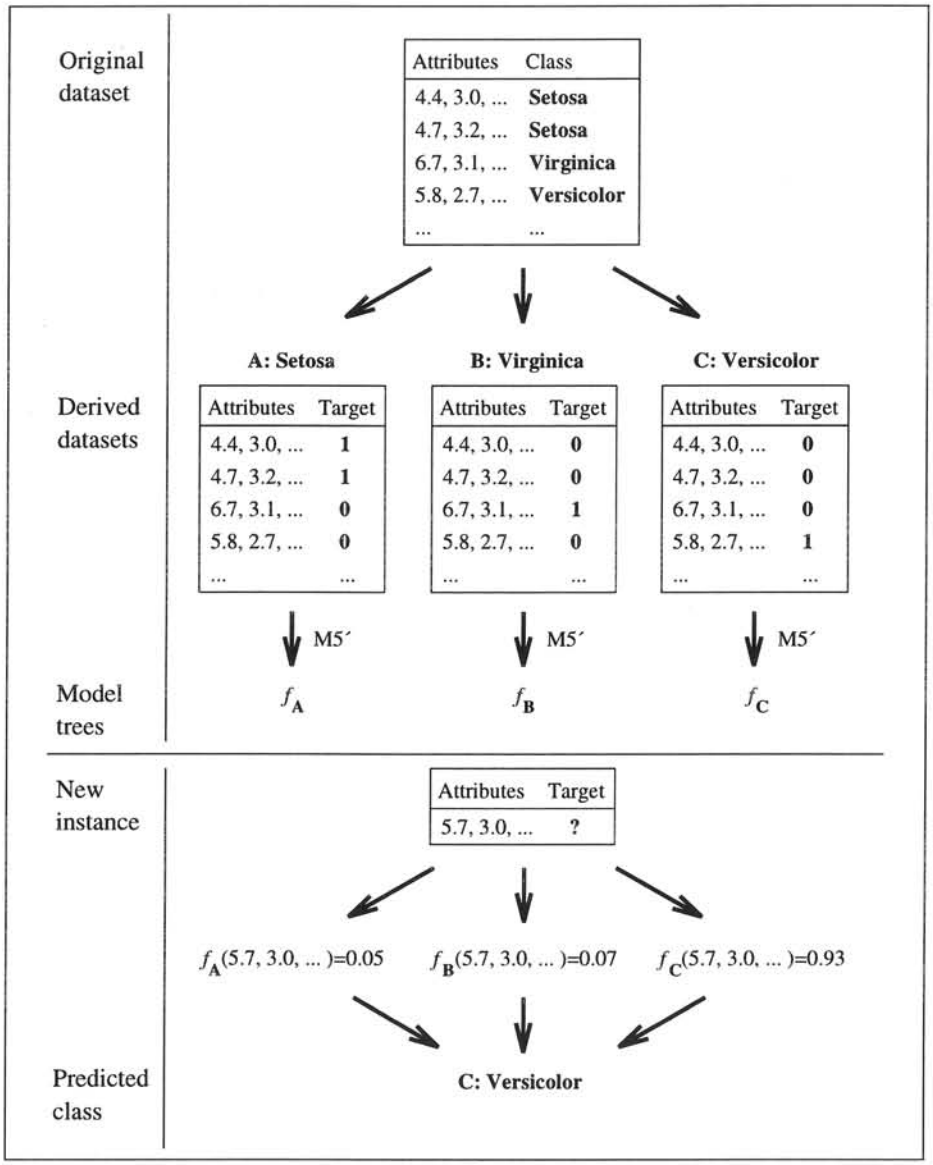
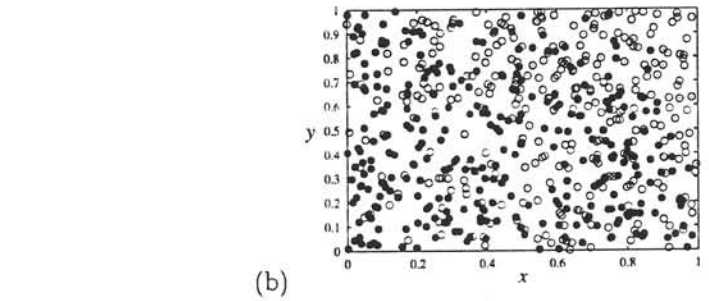
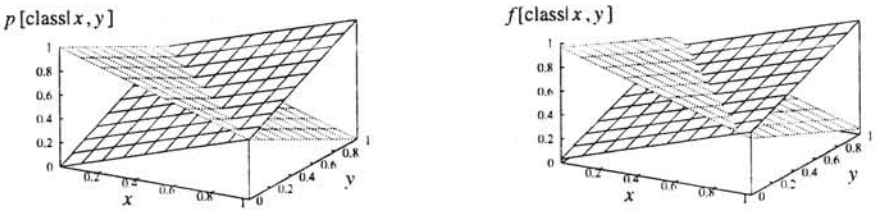


Figure 1. How M5′ is used for classification



(b)



(a) (c)

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

3.3. 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 adjacent regression functions at the leaves of the model tree.

4. Experimental results

Our experiments are designed to explore the application of model trees to classification by comparing their results with decision tree induction, 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?
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 model trees generated by M5′ with the pruned decision trees generated by C5.0 (Quinlan 1993). To investigate the second, we first compare the accuracy of the 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 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 M5′ are needed to generate SRT and UMT models.

4.1. Experiments

The twenty standard datasets used in the experiments are summarized in Table 1; they are taken from the UCI collection (Merz and Murphy, 1996). The first ten involve only numeric and binary attributes; the last ten involve non-binary nominal attributes as well.[3] 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 are shown. 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 5%-level according to a paired t-test. 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.

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
cleveland	303	2.3	10	3	0	2
glass (G2)	163	0.0	9	0	0	2
glass	214	0.0	9	0	0	6
ionosphere	351	0.0	33	1	0	2
iris	150	0.0	4	0	0	3
pima-indians	768	0.0	8	0	0	2
segment	2310	0.0	19	0	0	7
vehicle	846	0.0	18	0	0	4
waveform-noise	5000	0.0	40	0	0	3
audiology	226	2.0	0	61	9	24
anneal	898	65.0	6	7	25	5
autos	205	1.1	15	4	6	6
horse-colic	368	23.8	7	2	13	2
credit-rating	690	0.6	6	4	5	2
german	1000	0.0	6	3	11	2
lymphography	148	0.0	3	9	6	4
primary-tumor	339	3.9	0	14	3	21
soybean	307	6.6	0	17	18	19
zoo	101	0.0	1	15	1	7

4.2. Discussion of results

To answer the first question above, we observe from Table 2 that M5′ outperforms C5.0 in twelve datasets, whereas C5.0 outperforms M5′ in only one. (These numbers also appear, in boldface, in Table 3.) Of the ten datasets having numeric and binary attributes, M5′ is significantly more accurate on seven and significantly less accurate on none; on the remaining datasets it is significantly more accurate on five and significantly less accurate on one. 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.

The dataset for which model trees perform significantly worse than C5.0 is *autos*. Here, C5.0 generates decision trees that split on a 22-valued nominal attribute near the root: examination of M5′s model trees reveals that they fail to detect the importance of this attribute. M5′ converts a nominal attribute with n attribute values into $n-1$ binary attributes using the same procedure as employed by CART (Breiman *et al.*, 1984). But unlike CART it does this before the model tree is built, not locally at each node. We feel that this procedure, which is motivated by computational efficiency, may be a weakness of the current M5′ implementation rather than a fundamental problem with classifiers based on smoothed model trees, although this supposition has yet to be tested.

The version of M5′ used for these experiments differs from that described by Wang and Witten (1997) in its handling of missing values. The original version also had

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

Dataset	C5.0	M5′	M5′ SRT	M5′ UMT
balance-scale	77.6±0.9	86.5±0.8*	75.3±1.1	79.1±0.8
cleveland	78.8±1.2	80.9±1.6*	80.3±1.7	76.1±1.9
glass (G2)	78.8±2.0	81.1±1.9*	75.1±1.2	78.1±2.3
glass	67.5±2.7	70.5±2.7	67.5±1.7	67.1±2.8
ionosphere	88.9±1.6	89.6±1.0	88.1±0.7	87.3±1.0
iris	94.5±0.7	94.5±0.7	94.0±1.0	93.9±0.8
pima-indians	74.5±1.2	76.1±0.9*	75.6±1.0	72.0±0.8
segment	96.8±0.2	97.1±0.1*	96.2±0.2	95.9±0.3
vehicle	72.9±1.3	77.3±1.1*	70.9±1.2	69.2±1.2
waveform-noise	75.4±0.5	82.2±0.3*	80.3±0.3	72.2±0.4
audiology	77.6±1.3	76.7±0.9	67.9±1.2	76.8±1.8
anneal	93.2±0.5	93.4±0.8	75.0±1.7	94.2±0.8
autos	79.9±2.5*	75.3±2.1	70.1±2.1	71.6±2.3
horse-colic	85.3±0.6	84.7±0.7	84.7±0.8	83.2±1.6
credit-rating	85.3±0.5	85.8±0.9*	85.7±0.6	82.8±0.8
german	71.2±1.0	73.0±0.8*	74.1±0.9	70.0±0.9
lymphography	75.4±2.8	79.8±1.4*	76.1±1.6	77.0±2.9
primary-tumor	41.8±1.3	45.2±1.3*	45.3±1.1	41.2±1.3
soybean	83.6±1.3	85.9±1.5*	77.1±0.9	84.0±2.2
zoo	91.8±1.1	92.1±1.3	89.3±1.5	90.5±1.3

problems on the *anneal* dataset, which contains a high percentage of missing values, producing a mean error rate of 90.2±0.8 which is significantly worse than C5.0's performance. The new method of treating missing values, which is described in the Appendix, significantly improves performance for this dataset.

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). Table 3 shows that M5′ produces significantly more accurate classifiers on thirteen datasets (second column, third row) and significantly less accurate on only one (second row, third column). Compared to C5.0's pruned decision trees, classifiers based on smoothed regression trees are significantly less accurate on nine datasets and significantly more accurate on only 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, 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 seventeen datasets and significantly less accurate ones 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=0.05): number indicates how often method in column significantly outperforms method in row

	C5.0	M5′	M5′ SRT	M5′ UMT
C5.0	–	**12**	5	2
M5′	**1**	–	1	1
M5′ SRT	9	13	–	6
M5′ UMT	12	17	10	–

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 uniformly 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.

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. 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. Wang and Witten (1997) described a variant of Breiman *et al.*'s (1984) "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, the class 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 value v^* is selected to maximize the probability that the latter test has the same effect as the former. (In Breiman's original surrogate split method another attribute was chosen to split on; in this variant the class is always used.)

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 $(L + R)/|\mathcal{S}|$. We choose v^* so that the surrogate split $s^* < v^*$ maximizes this estimate.

In the original version of M5′ described by Wang and Witten (1997), 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 $s < v$. In the current version the decision is softened 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

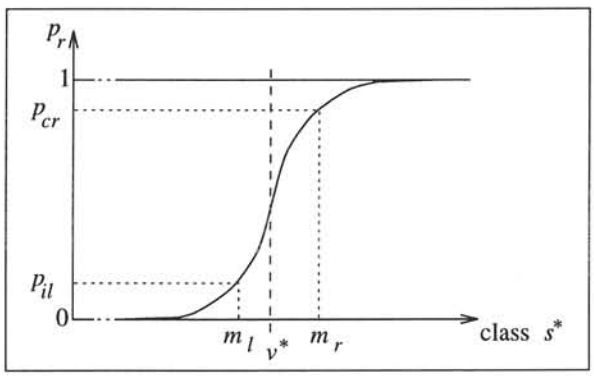


Figure A.1. How the soft step function model is fitted to the training data

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 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. see http://www.rulequest.com
2. see http://www.cs.waikato.ac.nz/~ml
3. 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.

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 Ghulum B. (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.* World Scientific, Singapore, pp. 343–348

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

Wang, Y. and Witten, I.H. (1997). "Induction of model trees for predicting continuous classes," *Proceedings European Conference on Machine Learning,* Prague, Czechoslovakia. Also available as Working Paper 96/23, Department of Computer Science, University of Waikato.