Using J-Pruning to Reduce Overfitting in Classification Trees

Max Bramer
Faculty of Technology, University of Portsmouth, Portsmouth, UK
max.bramer@port.ac.uk
www.dis.port.ac.uk/~bramerma

Abstract The automatic induction of classification rules from examples in the form of a decision tree is an important technique used in data mining. One of the problems encountered is the overfitting of rules to training data. In some cases this can lead to an excessively large number of rules, many of which have very little predictive value for unseen data. This paper is concerned with the reduction of overfitting during decision tree generation. It introduces a technique known as *J-pruning*, based on the *J-measure*, an information theoretic means of quantifying the information content of a rule.

1. Introduction

The growing commercial importance of knowledge discovery and data mining techniques has stimulated new interest in the automatic induction of classification rules from examples, a field in which research can be traced back at least as far as the mid-1960s [1].

Most work in this field has concentrated on generating classification rules in the intermediate form of a decision tree using variants of the TDIDT (Top-Down Induction of Decision Trees) algorithm [2], [3]. The TDIDT algorithm will be described briefly in Section 2. It is well known, widely cited in the research literature and an important component of commercial packages such as *Clementine*. However, like many other methods, it suffers from the problem of *overfitting* of the rules to the training data, resulting in some cases in excessively large rule sets and/or rules with very low predictive power for previously unseen data.

This paper is concerned with the reduction of overfitting in classification trees. Following a discussion of a number of alternative approaches in Section 3, a new technique known as *J-Pruning* is introduced in Section 4. The method is a refinement of the TDIDT algorithm, which enables a classification tree to be pruned while it is being generated, by making use of the value of the *J-measure*, an information theoretic means of quantifying the information content of a rule. Results are presented for a variety of datasets.

All but two of the datasets used (*wake_vortex* and *wake_vortex2*) were either created by the author or downloaded from the on-line repository of machine learning datasets maintained at the University of California at Irvine [4].

2. Automatic Induction of Classification Rules

2.1 Example and Basic Terminology

The following example is taken from [2]. Table 1 records a golf player's decisions on whether or not to play each day based on that day's weather conditions.

Outlook	Temp (°F)	Humidity (%)	Windy	Class
sunny	75	70	true	play
sunny	80	90	true	don't play
sunny	85	85	false	don't play
sunny	72	95	false	don't play
sunny	69	70	false	play
overcast	72	90	true	play
overcast	83	78	false	play
overcast	64	65	true	play
overcast	81	75	false	play
rain	71	80	true	don't play
rain	65	70	true	don't play
rain	75	80	false	play
rain	68	80	false	play
rain	70	96	false	play

Table 1. The *golf* Training Set. What combination of weather conditions determines whether the decision is to play or not to play?

The standard terminology is to call Table 1 a *training set* and each of its rows an *instance*. Each instance comprises the values of a number of *attributes* (variables) and the value of a corresponding *classification*. Attributes can be either *categorical* (such as outlook) or *continuous* such as humidity.

One possible set of classification rules that can be derived from Table 1 is as follows:

1. IF outlook = sunny AND humidity <= 75 THEN Class = play
2. IF outlook = sunny AND humidity > 75 THEN Class = don't play
3. IF outlook = overcast THEN Class = play
4. IF outlook = rain AND windy = true THEN Class = don't play
5. IF outlook = rain AND windy = false THEN Class = play

2.2 Inducing Decision Trees: The TDIDT Algorithm

The TDIDT algorithm constructs a set of classification rules via the intermediate representation of a decision tree. At each leaf node of the tree all the corresponding instances have the same classification.

A possible decision tree, corresponding to the classification rules given in Section 2.1, is shown in Figure 1.

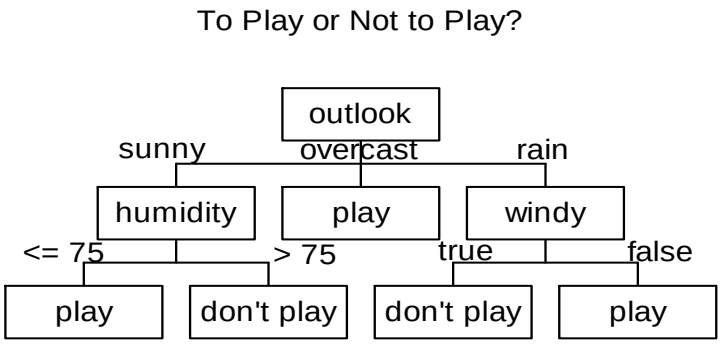

Figure 1. Decision Tree Corresponding to Table 1

The most commonly used criterion for selecting attributes to split on is probably *Information Gain* [3], which uses the information theoretic measure *entropy* at each stage to split on the attribute which maximises the expected gain of information from applying the additional test. Further details about automatic induction of classification trees are given in [5].

2.3 Using Classification Trees for Prediction

Using Information Gain to select the attribute to split on at each stage of the tree generation process generally produces very compact decision trees, which is highly desirable if the training set contains all possible instances in a given domain. However, in practice the training set generally contains only a sample (often a very small sample) of all possible instances in a domain, and there is no guarantee that using Information Gain (or any other attribute selection criterion) will produce classification trees that correctly predict all previously unseen instances.

A method of estimating the classification accuracy of rules, which will be used throughout this paper, is *ten-fold cross-validation*. First, the original dataset is divided into 10 approximately equal parts. Ten runs of the TDIDT algorithm are then performed, with each of the ten parts used as a test set in turn, and the other nine used as the training set each time. The results of these 10 runs are then combined to give an average number of rules and an average percentage level of predictive accuracy.

Dataset	Instances	Rules	Classification Accuracy
diabetes	768	121.9	70.3
genetics	3190	357.4	89.2
hypo	2514	14.2	99.5
lens24	24	8.4	70.0
wake_vortex*	1714	298.4	71.8

Table 2. TDIDT with Information Gain. 10-fold Cross Validation
** The wake_vortex and wake_vortex2 datasets were obtained from National Air Traffic Services Ltd. (NATS) for use in connection with a practical classification task. The former dataset is a restricted version with only four attributes. The latter is the full version with 51 attributes, 32 of them continuous.*

The results in Table 2 and elsewhere have been generated using ***Inducer,*** one of a suite of packages developed by the author to facilitate experiments with different techniques for generating classification rules. ***Inducer*** is implemented in Java in the interests of portability and is available both as a standalone application and as an applet. Further information is given in [6] and [7]. All the experiments use the TDIDT algorithm with the Information Gain attribute selection criterion and give the average results from 10-fold cross-validation.

The results in Table 2 show that for some datasets (such as *hypo*) TDIDT with Information Gain can produce compact rulesets with high predictive accuracy. For other datasets the method can produce a large ruleset (*genetics*) or one with relatively low predictive accuracy (*lens24*) or both (*diabetes* and *wake_vortex*).

Despite its limitations, the method is widely used in practice as a benchmark against which other algorithms are evaluated and has proved its worth as a robust method on which it is difficult to improve across a wide range of datasets.

2.4 Overfitting of Rules to Data

The principal problem with TDIDT and other algorithms for generating classification rules is that of *overfitting*. Beyond a certain point, specialising a rule by adding further terms can become counter-productive. The generated rules give a perfect fit for the instances from which they were generated but in some cases are too specific to have a high level of predictive accuracy for other instances. Another consequence of excessive specificity is that there are often an unnecessarily large number of rules. A smaller number of more general rules may have greater predictive accuracy on unseen data, at the expense of no longer correctly classifying some of the instances in the original training set. Even if the level of accuracy is not improved, deriving a smaller number of rules has obvious potential benefits.

3. Using Pruning to Improve Decision Trees

3.1 Post-pruning of Decision Trees

One approach to reducing overfitting, known as *post-pruning*, is to generate the set of classification rules using TDIDT as before and then remove a (possibly substantial) number of branches and terms, by the use of statistical tests or otherwise. Quinlan [2] describes a pruning technique based on predicted error rates, which is used in his well-known system C4.5. A variety of other methods have also been tried, with varying degrees of success. An empirical comparison of a number of methods is given in [8].

An important practical objection to post-pruning methods of this kind is that there is a large computational overhead involved in generating rules only then to delete a high proportion of them. This may not matter with small experimental datasets, but 'real-world' datasets may contain millions of instances and issues of computational feasibility and scaling up of methods will inevitably become important.

Holte [9] reports an empirical investigation of the accuracy of rules that classify on the basis of just a single attribute. These very simple rules perform surprisingly well compared with those produced by much more sophisticated methods. This too strongly suggests that a great deal of the effort involved in generating decision trees is either unnecessary or counterproductive and points to the potential value of a pre-pruning approach, as described in the next section, to avoid generating trees with an excessively large number of branches.

3.2 Pre-pruning of Decision Trees

Pre-pruning a decision tree involves terminating some of the branches prematurely as it is generated.

Each branch of the evolving tree corresponds to an incomplete rule such as

IF x = 1 AND z = yes AND q > 63.5 …. THEN …

and also to a subset of instances currently 'under investigation'.

If all the instances have the same classification, say c1, the end node of the branch is treated by the TDIDT algorithm as a leaf node labelled by c1. Each such completed branch corresponds to a (completed) rule, such as

IF x = 1 AND z = yes AND q > 63.5 THEN class = c1

If not all the instances have the same classification the node would normally be expanded to a subtree by splitting on an attribute, as described previously. When following a pre-pruning strategy the node (i.e. the subset) is first tested to

determine whether or not a termination condition applies. If it does not, the node is expanded as usual. If it does, the branch is *pruned*, i.e. the node is treated as a leaf node labelled with (usually) the most frequently occurring classification for the instances in the subset (the 'majority class').

The set of pre-pruned rules will classify all the instances in the training set, albeit wrongly in some cases. If the proportion of such misclassifications is relatively small, the classification accuracy for the test set may be greater than for the unpruned set of rules.

There are several criteria that can be applied to a node to determine whether or not pre-pruning should take place. Two of these are

- **Size Cutoff**. Prune if the subset contains fewer than say 5 or 10 instances
- **Maximum Depth Cutoff**. Prune if the length of the branch is say 3 or 4

Table 3 shows the results obtained from 10-fold cross-validation with a size cutoff of 5 instances, 10 instances or no cutoff (i.e. unpruned). Table 4 shows the results with a maximum depth cutoff of 3, 4 or unlimited.

Dataset	No Cutoff		5 Instances		10 Instances	
	Rules	% Acc.	Rules	% Acc.	Rules	% Acc.
breast-cancer	93.2	89.8	78.7	90.6	63.4	91.6
contact_lenses	16.0	92.5	10.6	92.5	8.0	90.7
diabetes	121.9	70.3	97.3	69.4	75.4	70.3
glass	38.3	69.6	30.7	71.0	23.8	71.0
hypo	14.2	99.5	11.6	99.4	11.5	99.4
monk1	37.8	83.9	26.0	75.8	16.8	72.6
monk3	26.5	86.9	19.5	89.3	16.2	90.1
sick-euthyroid	72.8	96.7	59.8	96.7	48.4	96.8
vote	29.2	91.7	19.4	91.0	14.9	92.3
wake_vortex	298.4	71.8	244.6	73.3	190.2	74.3
wake_vortex2	227.1	71.3	191.2	71.4	155.7	72.2

Table 3. Pre-pruning With Varying Size Cutoffs

Dataset	No Cutoff		Length 3		Length 4	
	Rules	% Acc.	Rules	% Acc.	Rules	% Acc.
breast-cancer	93.2	89.8	92.6	89.7	93.2	89.8
contact_lenses	16.0	92.5	8.1	90.7	12.7	94.4
diabetes	121.9	70.3	12.2	74.6	30.3	74.3
glass	38.3	69.6	8.8	66.8	17.7	68.7

hypo	14.2	99.5	6.7	99.2	9.3	99.2
monk1	37.8	83.9	22.1	77.4	31.0	82.2
monk3	26.5	86.9	19.1	87.7	25.6	86.9
sick-euthyroid	72.8	96.7	8.3	97.8	21.7	97.7
vote	29.2	91.7	15.0	91.0	19.1	90.3
wake_vortex	298.4	71.8	74.8	76.8	206.1	74.5
wake_vortex2	227.1	71.3	37.6	76.3	76.2	73.8

Table 4. Pre-pruning With Varying Maximum Depth Cutoffs

The results obtained clearly show that the choice of pre-pruning method is important. However, it is essentially *ad hoc*. No choice of size or depth cutoff consistently produces good results across all the datasets.

This result reinforces the comment by Quinlan [2] that the problem with pre-pruning is that the 'stopping threshold' is "not easy to get right - too high a threshold can terminate division before the benefits of subsequent splits become evident, while too low a value results in little simplification". There is therefore a need to find a more principled choice of cutoff criterion to use with pre-pruning than the size and maximum depth approaches used previously, and if possible one which can be applied completely automatically without the need for the user to select any cutoff threshold value. The *J-measure* described in the next section provides the basis for a more principled approach to pre-pruning of this kind.

4. Using the J-measure in Classification Tree Generation

4.1 Measuring the Information Content of a Rule

The *J-measure* was introduced into the rule induction literature by Smyth and Goodman [10], who give a strong justification of its use as an information theoretic means of quantifying the information content of a rule that is soundly based on theory.

Given a rule of the form **If Y=y, then X=x**, using the notation of [10], the (average) information content of the rule, measured in bits of information, is denoted by J(X;Y=y). The value of this quantity is given by the equation

$$J(X;Y=y) = p(y).j(X;Y=y)$$

Thus the J-measure is the product of two terms:

- p(y) The probability that the hypothesis (antecedent of the rule) will occur - a measure of *hypothesis simplicity*

- j(X;Y=y) The *j-measure* (note the small letter 'j') or *cross-entropy* - a measure of the *goodness-of-fit* of a given rule.

The cross-entropy term is defined by the equation:

$$j(X;Y=y) = p(x|y).\log_2(\frac{p(x|y)}{p(x)}) + (1-p(x|y)).\log_2(\frac{(1-p(x|y))}{(1-p(x))})$$

Smyth and Goodman state that the j-measure is the only non-negative measure of information satisfying the requirement that "the average information from all rules should be consistent with [Shannon's] standard definition for average mutual information between two [events]".

A plot of the j-measure for various values of p(x), the *a priori* probability of the rule consequent, is given in Figure 2.

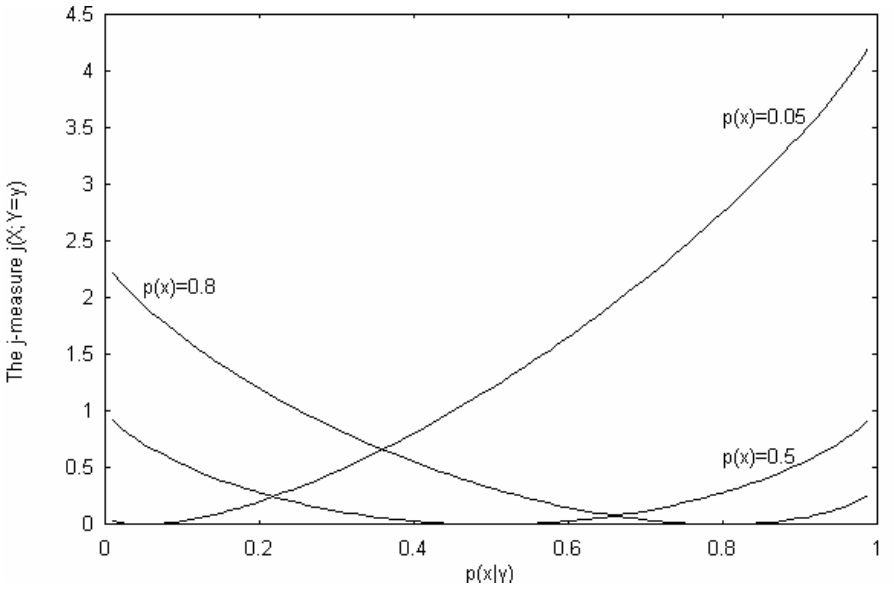

Figure 2. Plot of j-Measure for Various Values of p(x)

The J-measure has two helpful properties concerning upper bounds. First, it can be shown that the value of J(X;Y=y) is less than or equal to $p(y).\log_2(\frac{1}{p(y)})$.

The maximum value of this expression, given when p(y) = 1/e, is $\frac{\log_2 e}{e}$, which is approximately 0.5307 bits.

Second (and more important), it can be proved that the J value of any rule obtained by *specialising* the given rule by adding further terms is bounded by the value

$$J\max = p(y).\max\{p(x \mid y).\log_2(\frac{1}{p(x)}),(1-p(x \mid y)).\log_2(\frac{1}{1-p(x)})\}$$

Thus if a given rule is known to have a J value of, say, 0.352 bits and the value of Jmax is also 0.352, there is no benefit to be gained (and possibly harm to be done) by adding further terms to the left-hand side, as far as information content is concerned.

Further information on the J-measure and its uses is given in [11] and [12].

4.2 Using the J-measure for Rule Generation

In what follows, it will be taken as a working hypothesis that rules with high information content are also likely to have a high level of predictive accuracy for previously unseen instances.

In their system ITRULE [10] Smyth and Goodman make use of the availability of an upper bound Jmax on the J values of any possible further specialisations of a rule to generate the best *N* association rules from a given dataset, i.e. those with the highest J values. However, classification problems are normally concerned with finding all the rules necessary to make good classifications rather than, say, the best 50 rules.

The values of J for a set of rules generated from a given training set do not have any consistent range of values (apart from being between 0 and 0.5307). As an example, applying TDIDT with Information Gain to the *diabetes* and *lens24* datasets gives rulesets of 140 and 9 rules, respectively. (Note that these are obtained using the entire dataset as a single training set in each case, not from cross-validation.) The J values for the rules in the *diabetes* ruleset vary from 0.0008 to 0.1056, whereas those for the *lens24* dataset vary from 0.0283 to 0.3390. It is difficult to give any physical interpretation to these values. It would be possible to post-prune a set of rules by discarding all rules except those with the highest *N* values of the J-measure or all those with J values below a certain threshold, but in general this could lead to a large number of instances in the training set left unclassified by the remaining rules and a corresponding loss of predictive accuracy for previously unseen instances. The analysis given in the next section points towards an alternative approach, using pre-pruning.

4.3 A J-measure Interpretation of Overfitting

The results given in Section 3.2 strongly suggest that, beyond a certain point, adding further terms to rules (by splitting on additional attributes) can become counter-productive because of overfitting. Analysing successive forms of a rule using the J-measure clarifies why this happens.

Taking the l*ens24* dataset for illustration, one of the rules generated is

IF tears=2 AND astig=1 AND age=3 AND specRx=1 THEN class=3

This has a J-value of 0.028 and seems a reasonable rule. However, by looking at the way the rule develops term by term a different picture emerges.

After just one term, the rule and corresponding J and Jmax values were

IF tears=2 THEN class=3 (J=0.210, Jmax=0.531)

In general, specialising a rule by adding further terms may either increase or decrease the value of J (i.e. the information content). However the value of Jmax gives the maximum J value that any possible specialisation of the rule may achieve. In this case Jmax = 0.531, so it seems appropriate to continue developing the rule.

Adding the second term gives

IF tears=2 AND astig=1 THEN class=3 (J= 0.161, Jmax=0.295)

The J value has gone down from 0.210 to 0.161, but has the potential to increase again, possibly up to 0.295 by further specialisation.

Adding the third and fourth terms completes the picture.

IF tears=2 AND astig=1 AND age=3 THEN class=3 (J= 0.004, Jmax=0.059)

IF tears=2 AND astig=1 AND age=3 AND specRx=1 THEN class=3
(J= 0.028, Jmax=0.028)

It can be seen that adding additional terms to rules can either increase or decrease the value of J. However, the combined effect of adding the three final terms has been to lower the J value (information content) of the rule by almost a factor of 10. If we assume that the J measure is a reliable indicator of the information content and thus the predictive accuracy of a rule, it would have been better to truncate the rule after a single term (classifying all the instances in the majority class). This would have led to more misclassified instances for the training data, but might have led to better predictive accuracy on unseen data.

4.4 J-Pruning

There are several ways in which J values can be used to aid classification tree generation. One method, which will be called *J-pruning*, is to prune a branch as soon as a node is generated at which the J value is less than that at its parent.

Looking at this in terms of partially completed rules, say there is an incomplete rule for the *lens24* dataset

(1) IF tears=2 AND astig=2 ….

Splitting on attribute specRx (which has two values) would add an additional term, making the incomplete rule

(2) IF tears=2 AND astig=2 AND specRx=1 ….
or
(3) IF tears=2 AND astig=2 AND specRx=2 ….

All the instances corresponding to branch (2) have the same classification, so the rule is completed with that classification in the usual way. However the instances corresponding to branch (3) have more than one classification.

The J-pruning technique now involves a comparison between the J-value of (3) and the J-value of (1). If the former is smaller, the rule is truncated and the instances are all classified as belonging to the majority class, i.e. the class to which the largest number of instances belong. If not, the TDIDT algorithm continues by splitting on an attribute as usual.

The difficulty in implementing the above method is that the value of J depends partly on the class specified in the rule consequent, but when the partial rules (incomplete branches) are generated there is no way of knowing which class that will eventually be. A branch may of course be extended by TDIDT to have a large descendent subtree, obtained by subsequent splittings on attributes, with many leaf nodes each of which has its own classification.

If the rules had been truncated at (1) there are 3 possible ways in which all the instances could have been assigned to a single class. These are listed below with the corresponding values of J and Jmax

IF tears=2 AND astig=2 THEN class=1 (J = 0.223, Jmax = 0.431)
IF tears=2 AND astig=2 THEN class=2 (J = 0.084, Jmax = 0.084)
IF tears=2 AND astig=2 THEN class=3 (J = 0.063, Jmax = 0.236)

There are 3 possible ways in which the instances corresponding to (3) could be assigned to a single class:

IF tears=2 AND astig=2 AND specRx=2 THEN class=1 (J=0.015, Jmax=0.108)
IF tears=2 AND astig=2 AND specRx=2 THEN class=2 (J=0.042, Jmax=0.042)
IF tears=2 AND astig=2 AND specRx=2 THEN class=3 (J=0.001, Jmax=0.059)

If there are only two classes the value of J is the same whichever is taken. When there are more than two classes the J values will generally not all be the same. One possibility would be always to use the J value of the majority class, but in practice it has been found to be more effective to use the largest of the possible J values in each case. Thus the J values for branches (1) and (3) are taken to be 0.223 and 0.042 respectively. Since the value for (3) is lower than for (1), J-pruning takes place and branch (3) is truncated.

Table 5 shows the results obtained using J-pruning with a variety of datasets and the comparative figures for unpruned rules.

Dataset	No J-Pruning		With J-Pruning	
	Rules	% Accuracy	Rules	% Accuracy
breast-cancer	93.2	89.8	66.5	91.3
contact_lenses	16.0	92.5	8.3	92.6
crx	127.5	79.4	20.4	85.4
diabetes	121.9	70.3	6.4	75.1
genetics	357.4	89.2	25.9	78.2
glass	38.3	69.6	9.4	63.5
hepatitis	18.8	82.0	4.5	81.2
hypo	14.2	99.5	7.6	99.3
iris	8.5	95.3	5.7	94.7
lens24	8.4	70.0	6.2	70.0
monk1	37.8	83.9	14.4	67.8
monk2	88.4	43.8	21.3	55.7
monk3	26.5	86.9	12.5	90.9
sick-euthyroid	72.8	96.7	6.8	97.8
vote	29.2	91.7	11.1	94.0
wake_vortex	298.4	71.8	12.0	73.5
wake_vortex2	227.1	71.3	12.4	72.5

Table 5. Comparison of Unpruned and J-pruned Rules

The reduction in the number of rules is clearly considerable for many of the datasets (e.g. from 121.9 to 6.4 for *diabetes* and from 298.4 to 12.0 for *wake_vortex*). This again confirms that the basic (unpruned) form of TDIDT leads to substantial overfitting of rules to the instances in the training set. The predictive accuracy is higher with J-pruning for 10 of the datasets, lower for 6 and unchanged for one (the smallest dataset, *lens24*). There are large increases in accuracy for *crx* and *monk2* and large decreases in accuracy for *genetics* and *monk1*.

The predictive accuracy obtainable from a dataset depends on many factors, including the appropriateness of the choice of attributes, so large improvements should not necessarily be expected from J-pruning (or any other form of pruning). However there are obvious benefits from a large reduction in the number of rules even when there is no gain in accuracy.

4.5 Limitations of the Decision Tree Representation

The method of using the J-measure for pre-pruning adopted here has limitations that relate directly to the use of the decision tree representation imposed by TDIDT.

Suppose that the branch shown as a solid line in Figure 3 has been developed by TDIDT as far as node N_0. (Other irrelevant branches are shown as dotted lines.) The decision needed is whether or not to develop the branch further by splitting on an additional attribute, giving nodes N_1, N_2 and N_3.

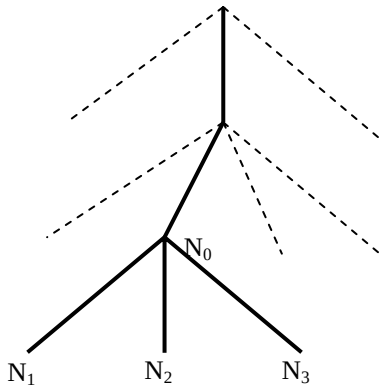

Figure 3. A Partially Generated Decision Tree

Suppose that the J values of nodes N_0, N_1 and N_2 are 0.25, 0.4 and 0.005 respectively, so that the left-hand branch is developed by further splitting and the middle branch is J-pruned. Suppose also that all the instances corresponding to node N_3 have the same classification, so that TDIDT treats it as a leaf node in the usual way. As far as the middle branch, which is J-pruned, is concerned it would have been better if the branch had been terminated a level earlier, i.e. at N_0 (with a higher value of J). However doing so would also have eliminated the left-hand and right-hand branches, which would clearly have been disadvantageous.

One possibility in this case would be to try combining the middle branch with either the left-hand or the right-hand branch. However there are many other possible situations that can arise and it is difficult to deal with all of them satisfactorily within the decision tree framework.

The use of a decision tree representation for rules has previously been identified as a major cause of overfitting ([7], [13]). An example is given in [13] of two rules with no attribute in common which lead to a complex decision tree almost all branches and terms of which are redundant. Further reductions in overfitting are likely to come from incorporating J-pruning or other pre-pruning techniques into algorithms such as Prism [13] that generate classification rules directly rather than through the intermediate representation of decision trees.

5. Conclusions

This paper has demonstrated the potential value of using the information-theoretic J-measure as the basis for reducing overfitting by pre-pruning branches during classification tree generation. The J-pruning technique illustrated works well in practice for a range of datasets. Unlike many other possible measures, the J-measure has a sound theoretical foundation as a measure of the information content of rules.

The decision tree representation of TDIDT is widely used and it is therefore desirable to find methods of pre-pruning that work well with this representation. However, the decision tree representation is itself a source of overfitting. For substantial further improvements techniques that work with algorithms that directly generate classification rules not classification trees will probably be necessary and the J-pruning method would appear to be well suited to this.

References

[1] Hunt, E.B., Marin J. and Stone, P.J. (1966). Experiments in Induction. Academic Press

[2] Quinlan, J.R. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann

[3] Quinlan, R. (1986). Induction of Decision Trees. Machine Learning, 1, pp. 81-106

[4] Blake, C.L. and Merz, C.J. (1998). UCI Repository of Machine Learning Databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science

[5] Bramer, M.A. (1997). Rule Induction in Data Mining: Concepts and Pitfalls Data Warehouse Report, No. 10, pp. 11-17 and No. 11, pp. 22-27

[6] Bramer, M.A. (2000). Inducer: a Rule Induction Workbench for Data Mining. In Proceedings of the 16th IFIP World Computer Congress Conference on Intelligent Information Processing (eds. Z.Shi, B.Faltings and M.Musen) Publishing House of Electronics Industry (Beijing), pp. 499-506

[7] Bramer, M.A. (2000). Automatic Induction of Classification Rules from Examples Using N-Prism. In: Research and Development in Intelligent Systems XVI. Springer-Verlag, pp. 99-121

[8] Mingers, J. (1989). An Empirical Comparison of Pruning Methods for Decision Tree Induction. Machine Learning, 4, pp. 227-243

[9] Holte, R.C. (1993). Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. Machine Learning, 11, pp. 63-90

[10] Smyth, P. and Goodman, R.M. (1991). Rule Induction Using Information Theory. In: Piatetsky-Shapiro, G. and Frawley, W.J. (eds.), Knowledge Discovery in Databases. AAAI Press, pp. 159-176

[11] Nazar, K. and Bramer, M.A. (1997). Concept Dispersion, Feature Interaction and Their Effect on Particular Sources of Bias in Machine Learning. In Hunt, J. and Miles, R. (eds.), Research and Development in Expert Systems XIV, SGES Publications.

[12] Nazar, K. and Bramer, M.A. (1999). Estimating Concept Difficulty With Cross-Entropy. In Bramer, M.A. (ed.), Knowledge Discovery and Data Mining, Institution of Electrical Engineers, London.
[13] Cendrowska, J. (1987). PRISM: an Algorithm for Inducing Modular Rules. International Journal of Man-Machine Studies, 27, pp. 349-370