Using J-Pruning to Reduce Overfitting of Classification Rules in Noisy Domains

Max Bramer

Faculty of Technology, University of Portsmouth, UK
Max.Bramer@bcs.org.uk
http://www.btinternet.com/~Max.Bramer

Abstract. The automatic induction of classification rules from examples is an important technique used in data mining. One of the problems encountered is the overfitting of rules to training data. This paper describes a means of reducing overfitting known as *J-pruning*, based on the *J-measure*, an information theoretic means of quantifying the information content of a rule, and examines its effectiveness in the presence of noisy data for two rule induction algorithms: one where the rules are generated via the intermediate representation of a decision tree and one where rules are generated directly from examples.

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 to date 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]. An alternative approach, which generates classification rules directly from examples, is Prism [3,4].

A problem that arises with all methods of generating classification rules is that of *overfitting* to the training data. In some cases this can result in excessively large rule sets and/or rules with very low predictive power for previously unseen data.

A method for reducing overfitting in classification rules known as *J*-pruning has previously been reported [5]. The method makes use of the value of the *J-measure*, an information theoretic means of quantifying the information content of a rule. The rules are *pre-pruned*, i.e. pruned as they are being generated.

In this paper the robustness of this technique in the presence of noise is examined. A comparison is made between the results obtained from the unpruned and J-pruned versions of both TDIDT and Prism for varying levels of noise added in a systematic fashion to three datasets from the UCI Repository of Machine Learning Datasets [6].

The use of J-pruning leads in all cases to a reduction in the number of rules generated and in many cases to an increase in predictive accuracy.

2 Automatic Induction of Classification Rules from Examples

2.1 Basic Terminology

It is assumed that there is a universe of *objects*, each of which belongs to one of a set of mutually exclusive *classes*. Objects are described by the values of a number of their *attributes*. There is a two-dimensional table of examples, known as a t*raining set*, each row of which (an *instance*) comprises the values of the attributes and the corresponding classification for a single object. The aim is to develop classification rules that enable the class to which any object in an unseen *test set* of further instances belongs to be determined from the values of its attributes. It will be assumed that the rules are to be in propositional form, each comprising a conjunction of *terms*, such as

IF x=a AND y=b AND z>34.5 AND w=k THEN Class=3

2.2 Top-Down Induction of Decision Trees

Many systems have been developed to derive classification rules of the above kind from a training set. Most (but not all) do so via the intermediate form of a decision tree constructed using a variant of the TDIDT (top-down induction of decision trees) algorithm given in Figure 1 below.

IF all cases in the training set belong to the same class
THEN return the value of the class
ELSE

(a) Select an attribute A to *split* on *
(b) Sort the instances in the training set into non-empty subsets, one for each value of attribute A
(c) Return a tree with one branch for each subset, each branch having a descendant subtree or a class value produced by applying the algorithm recursively for each subset in turn.

* When selecting attributes at step (a) the same attribute must not be selected more than once in any branch.

Fig. 1. The TDIDT Tree Generation Algorithm

The induced decision tree can be regarded as a set of classification rules, one corresponding to each branch.

The most widely used criterion for selecting attributes at step (a) is probably *Information Gain.* This uses the information-theoretic measure *entropy* to choose the attribute that maximises the expected gain of information from applying the additional test. This is the approach adopted in well-known systems such as C4.5 [2].

2.3 The Prism Algorithm

The Prism classification rule generation algorithm was developed by Cendrowska [3], primarily as a means of avoiding the generation of unnecessarily complex rules, which it was argued is an unavoidable but undesirable consequence of the use of a tree representation. The need to fit rules into such a representation requires them all to begin with a test on the value of the same attribute, even though that attribute may be irrelevant to many or most of the rules.

The Prism algorithm induces classification rules directly from a training set one rule at a time. Each rule is generated term-by-term, by selecting the attribute-value pair that maximises the probability of a chosen outcome class.

The version of Prism described in this paper is a modified form known as PrismTCS (standing for Prism with Target Class, Smallest first), which has been found to produce smaller sets of classification rules than the original form of the algorithm, with a similar level of predictive accuracy. With the original version of Prism, the training set is restored to its original state before the rules are generated for each class, thus requiring the training set to be processed once for each of the classes.

Instead PrismTCS makes use of a *target class*, which varies from one rule to the next as shown in Figure 2. With this form of the algorithm the full training set only needs to be processed once however many classes there are.

(1) Find the class with fewest instances in the training set (ignoring any with none). Call this the *target class* TC.

(2) Calculate the probability that class = TC for each possible attribute-value pair *

(3) Select the attribute-value pair with the maximum probability and create a subset of the training set comprising all instances with the selected combination (for all classes)

(4) Repeat 2 and 3 for this subset until it contains only instances of class TC. The induced rule is then the conjunction of all the attribute-value pairs selected in creating this subset

(5) Remove all instances covered by this rule from the training set

Repeat 1-5 until there are no instances remaining in the training set

* Any attribute that is part of an attribute-value pair already selected should not be used again for the same rule

Fig. 2. The PrismTCS Rule Generation Algorithm

3 Overfitting of Rules to Data

The principal problem with TDIDT, Prism 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 is 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. Alternatively, a similar level of accuracy may be achieved with a more compact set of rules.

3.1 Pruning Classification Rules to Reduce Overfitting

One approach to reducing overfitting, known as *post-pruning*, which is often used in association with decision tree generation, is to generate the whole set of classification rules and then remove a (possibly substantial) number of rules and terms, by the use of statistical tests or otherwise. An empirical comparison of a number of such methods is given in [7]. An important practical objection to post-pruning methods is that there is a large computational overhead involved in generating rules only then to delete a high proportion of them, especially if the training sets are large.

Pre-pruning a set of classification rules (or a decision tree) involves terminating some of the rules (branches) prematurely as they are being generated. Each incomplete rule such as

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

corresponds to a subset of instances currently 'under investigation'.

If not all the instances have the same classification the rule would normally be extended by adding a further term, as described previously. When following a pre-pruning strategy the subset is first tested to determine whether or not a termination condition applies. If it does not, a further term is generated as usual. If it does, the rule is *pruned*, i.e. it is treated as if no further attributes were available. Typically the rule will be treated as completed, with all the instances classified as belonging to the class to which the largest number belong.

Reference [5] reports on experiments with four possible termination conditions for pre-pruning rules as they are generated by TDIDT, e.g. truncate each rule as soon as it reaches 4 terms in length. The results obtained clearly show that pre-pruning can substantially reduce the number of terms generated and in some cases can also increase the predictive accuracy. Although they also show that the choice of pre-pruning method is important, it is not clear that (say) the same length limit should be applied to each rule, far less which of the termination conditions is the best one to use or why. There is a need to find a more principled choice of termination condition to use with pre-pruning, if possible one which can be applied completely automatically without the need for the user to select any 'threshold value' (such as the maximum

number of terms for any rule). The *J-measure* described in the next section provides the basis for a more principled approach to pre-pruning.

4 Using the J-measure to Prune Classification Rules

4.1 Measuring the Information Content of a Rule

The *J-measure* was introduced into the rule induction literature by Smyth and Goodman [8] 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 [8], 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 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 *cross-entropy* - a measure of the *goodness-of-fit* of a given rule.

In what follows, it will be taken as a working hypothesis that a rule with a high J value (i.e. high information content) is also likely to have a high level of predictive accuracy for previously unseen instances.

4.2 Using J-Pruning with TDIDT and Prism

There are several ways in which J values can be used to aid classification tree generation using TDIDT. 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.

Thus for example consider an incomplete rule

IF attrib1 = a AND attrib2 = b …. (with J-value 0.4)

which is expanded by splitting on categorical attribute *attrib3* into the three rules

IF attrib1 = a AND attrib2 = b AND attrib3 = c1 …. (with J-value 0.38)
IF attrib1 = a AND attrib2 = b AND attrib3 = c2 …. (with J-value 0.45)
IF attrib1 = a AND attrib2 = b AND attrib3 = c3 …. (with J-value 0.03)

Assuming that none of the new rules is complete (i.e. corresponds to a subset of instances with only one classification) all three would be considered as candidates for J-pruning. As the J-values of the first and third are lower than that of the original (incomplete) rule each rule would be truncated, with all the corresponding instances classified as belonging to the class to which the largest number belong. For example, the first new rule might become

IF attrib1 = a AND attrib2 = b AND attrib3 = c1 THEN Class = 5

The second new rule has a larger J-value than the original rule and in this case the TDIDT algorithm would continue by splitting on an attribute as usual.

The difficulty in implementing this method is to know which classification to use when calculating the J-value of an incomplete rule. If there are only two classes the

value of J is the same whichever is taken. When there are more than two classes an effective heuristic is to generate the J-value for each of the possible classes in turn and then to use the largest of the resulting values.

Reference [5] compares the results obtained using the TDIDT algorithm both with and without J-pruning for 12 datasets, mainly taken from the UCI Repository [6]. The results were calculated using 10-fold cross-validation in each case. TDIDT was used with the Information Gain attribute selection criterion throughout.

For many of the datasets a considerable reduction in the number of rules was obtained using J-Pruning (e.g. from 357.4 unpruned to 25.9 J-pruned for *genetics* and from 106.9 unpruned to 29.6 J-pruned for *soybean*). Averaged over the 12 datasets the number of rules was reduced from 68.5 to only 19.1. The effect on the predictive accuracy of the generated rulesets varied considerably from one dataset to another, with J-pruning giving a result that was better for 5 of the datasets, worse for 6 and unchanged for one, the average being slightly lower with J-Pruning than without.

In the case of PrismTCS classification rules J-pruning takes a simpler form. At each stage of rule generation the J-value of the incomplete rule is calculated and recorded. If at any stage adding an additional term would lead to a decrease in the J-value, the term is discarded. Provided the class to which the largest number of instances belongs is the current target class, the rule is completed with all the instances classified as belonging to the target class. Otherwise the incomplete rule and the corresponding instances are discarded.

Reference [9] compares the results obtained using PrismTCS both with and without J-pruning for 12 datasets, mainly taken from the UCI Repository [6]. The results were calculated using 10-fold cross-validation in each case.

The number of rules generated for the unpruned version of the PrismTCS algorithm was on average significantly smaller than for the unpruned version of TDIDT with the same datasets. Nevertheless the use of J-pruning with PrismTCS reduced the number of rules by more than one-third, with a substantial reduction from 87.7 rules to only 25.1 for *genetics* and a halving of the number of rules from 37.3 to 16.9 for *monk2*, in both cases accompanied by an increase in predictive accuracy. The predictive accuracy was larger for the J-pruned rule sets in seven cases and smaller for only three. On average there was a small increase in predictive accuracy despite the substantially reduced number of rules.

Although these results were very promising, as were those from the experiments with TDIDT, an important criterion for evaluating any classification rule generation algorithm is its *robustness*, particularly when noise is present in the data.

5 Experiments with Noisy Datasets

Many (perhaps most) real-world datasets suffer from the problem of *noise*, i.e. inaccurately recorded attribute or classification values. Although the user of a rule generation algorithm will generally be unaware that noise is present in a particular dataset, far less the proportion of values that are affected, the presence of noise is likely to lead to an excessively large number of rules and/or a reduction in classification accuracy compared with the same data in noise-free form.

The robustness of the unpruned and J-pruned versions of the TDIDT and PrismTCS algorithms to noise was investigated using the *vote* dataset from the UCI Repository [6]. The dataset comprises information about the votes of each of the members of the US House of Representatives on 16 key measures during 1984. The dataset has 300 instances, each relating the values of 16 categorical attributes to one of two possible classifications: *republican* or *democrat*. It seems reasonable to suppose that the members' votes will have been recorded with few (if any) errors, so the *vote* dataset in its original form will be considered noise-free.

From this dataset further datasets were created by contaminating the attribute values with progressively higher levels of noise. There were eight such datasets, named *vote_10*, *vote_20*, …, *vote_80*, with the numerical suffix indicating the percentage of contaminated values.

The methodology adopted in the case of say *vote_30* was to consider the possibility of contaminating each attribute value in each instance in turn. For each value a random number from 0 to 1 was generated. If the value was less than or equal to 0.30 the attribute value was replaced by another of the valid possible values of the same attribute, selected with equal probability. The original classification was left unchanged in all cases. As the level of noise contamination increases from zero (the original dataset), through 10%, 20%, … up to 80%, it is to be expected that (with any method) the predictive accuracy of any ruleset generated will decline.

5.1 Experimental Results: TDIDT

Figure 3 shows the number of rules generated using the TDIDT algorithm (with the 'Information Gain' attribute selection criterion) in its standard 'unpruned' form and with J-pruning for each of the datasets *vote_10*, *vote_20*, … *vote_80*. Figure 4 shows the corresponding levels of predictive accuracy for the two forms of the algorithm for the nine versions of the *vote* dataset. All results were calculated using 10-fold cross-validation. The J-pruned algorithm clearly produces substantially fewer rules with at least as good predictive accuracy as the unpruned version.

This experiment was repeated for two further datasets taken from the UCI Repository: *genetics* and *agaricus_lepiota*. The *genetics* dataset comprises 3,190 instances, each with 60 categorical attributes and 3 possible classifications. The *agaricus_lepiota* dataset comprises 5,644 instances (after those containing any missing values were removed), each with 22 categorical attributes and 2 possible classifications. These datasets were chosen partly because all the attributes were categorical. It was considered that categorical values were less likely to be wrongly (or imprecisely) recorded than continuous ones. The results of the experiments for these datasets (again calculated using 10-fold cross-validation) are given in Table 1, with values rounded to the nearest integer.

The reduction in the number of rules obtained using J-pruning increases substantially as the percentage of noise in the data increases. In the most extreme case, for *agaricus_lepiota_80*, the unpruned version of TDIDT gives 2916 rules and the J-pruned version only 19. The predictive accuracy obtained using J-pruning was better than that for the unpruned version of TDIDT in all cases where the proportion of noise exceeded 10%.

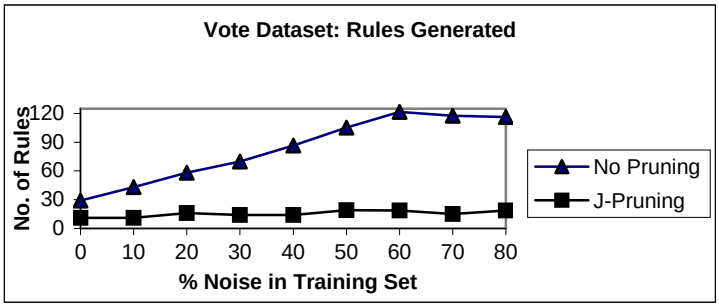

Fig. 3. Comparison of Number of Rules Generated: *vote* Dataset

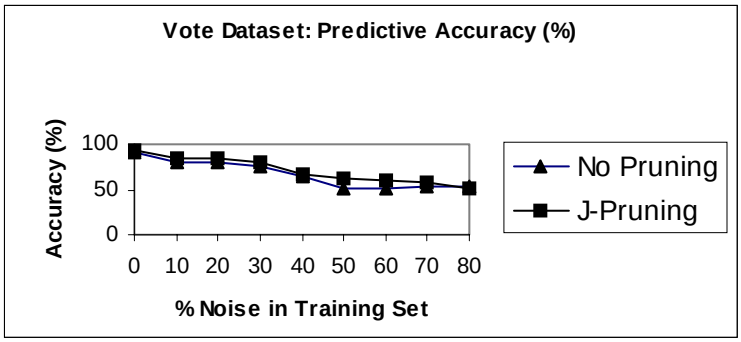

Fig. 4. Comparison of Predictive Accuracy: *vote* Dataset

Table 1. Rules Generated and Predictive Accuracy: *genetics* and *agaricus_lepiota*

	genetics				*agaricus_lepiota*			
Noise %	Rules		Accuracy (%)		Rules		Accuracy (%)	
	Un-pruned	Pruned	Un-pruned	Pruned	Un-pruned	Pruned	Un-pruned	Pruned
0	357	26	89	78	15	10	100	100
10	918	122	73	72	349	96	96	95
20	1238	158	60	67	794	128	89	91
30	1447	185	54	64	1304	149	81	86
40	1652	175	44	60	1827	159	72	80
50	1815	163	36	55	2246	167	64	76
60	1908	165	33	52	2682	167	55	71
70	1998	153	29	51	3003	184	48	67
80	2074	179	27	48	2916	19	52	74
Ave.	1490	147	49	61	1682	120	73	82

5.2 Experimental results: PrismTCS

The experiments described in Section 5.1 were repeated with the J-pruned and unpruned versions of the PrismTCS algorithm.

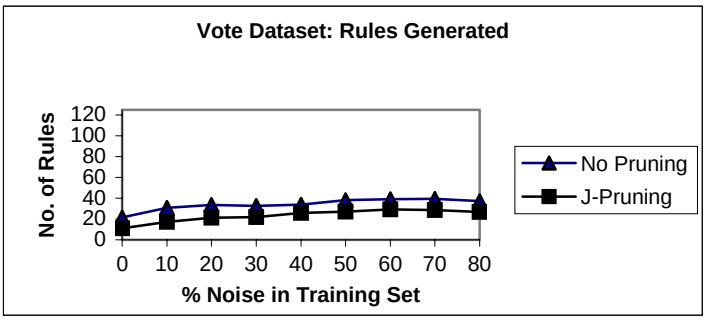

Fig. 5. Comparison of Number of Rules Generated: *vote* Dataset

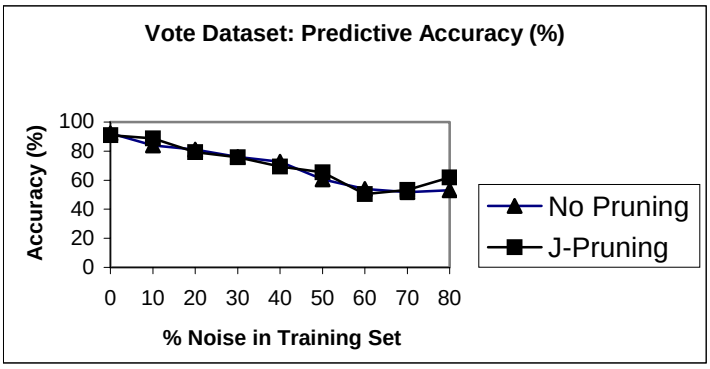

Fig. 6. Comparison of Predictive Accuracy: *vote* Dataset

Table 2. Rules Generated and Predictive Accuracy: *genetics* and *agaricus_lepiota*

	genetics				*agaricus_lepiota*			
Noise %	Rules		Accuracy (%)		Rules		Accuracy (%)	
	Un-pruned	Pruned	Un-pruned	Pruned	Un-pruned	Pruned	Un-pruned	Pruned
0	88	25	91	93	12	12	100	100
10	438	264	71	78	177	68	96	97
20	380	269	64	70	238	131	92	93
30	399	249	58	63	312	183	87	89
40	388	270	57	59	441	242	81	81
50	399	285	48	55	556	264	74	74
60	393	264	46	48	641	252	69	70
70	393	230	41	41	676	239	62	64
80	399	204	38	39	625	167	63	69
Ave.	364	229	57	61	408	173	80	82

The results for the *vote* dataset are shown in Figures 5 and 6. Table 2 gives the results for *genetics* and *agaricus_lepiota*. In the case of the *vote* dataset, the increase in the number of rules as the percentage of noise increases is much less steep than for TDIDT. However, the use of J-pruning gives not only a smaller number of rules but in most cases slightly better predictive accuracy.

The *genetics* dataset is clearly highly sensitive to even a small amount of noise. However, for both this dataset and *agaricus_lepiota* using J-pruning not only reduces the number of rules, in most cases substantially, but also gives a small improvement in predictive accuracy.

6 Conclusions

Although a comparison between the TDIDT and PrismTCS algorithms is not the principal aim of this paper, it would seem that PrismTCS is more robust to noise than TDIDT. Although increasing the percentage of noise inevitably leads to an increase in the number of rules and a reduction in predictive accuracy with both algorithms, the effect is generally more extreme with TDIDT.

The J-pruning technique is a valuable means of reducing overfitting for both TDIDT and PrismTCS, which is robust in the presence of noise. Using J-pruning will generally lead to a substantial reduction in the number of classification rules generated for both algorithms. This will often be accompanied by a gain in predictive accuracy. In general, the advantage gained by using J-pruning becomes more pronounced as the proportion of noise in a dataset increases.

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. Cendrowska, J. (1987). PRISM: an Algorithm for Inducing Modular Rules. International Journal of Man-Machine Studies, 27, pp. 349-370
4. 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
5. Bramer, M.A. (2002). Using J-Pruning to Reduce Overfitting in Classification Trees. In: Research and Development in Intelligent Systems XVIII. Springer-Verlag, pp. 25-38.
6. 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
7. Mingers, J. (1989). An Empirical Comparison of Pruning Methods for Decision Tree Induction. Machine Learning, 4, pp. 227-243
8. 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
9. Bramer, M.A. (2002). An Information-Theoretic Approach to the Pre-pruning of Classification Rules. Proceedings of the IFIP World Computer Congress, Montreal 2002.