An Information-Theoretic Approach to the Pre-pruning of Classification Rules

Max Bramer
University of Portsmouth, Portsmouth, UK

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. 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. 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 and applies this to two rule induction methods: one where the rules are generated via the intermediate representation of a decision tree and one where rules are generated directly from examples.

Keywords: Knowledge Discovery, Data Mining, Classification Rules, Decision Trees

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 the examples, is Prism [3,4].

A problem that arises with all methods of generating rules is that of *overfitting* of the rules 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.

This paper describes a method for reducing overfitting in classification rules known as *J-Pruning*. 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.

Results are presented for a range of datasets using both TDIDT and Prism. 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

A more detailed description is given in [5].

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.

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

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.

Figure 1. The TDIDT Tree Generation Algorithm

entropy to choose the attribute which 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.

Reference [4] introduced a revised form of the Prism algorithm, incorporating a number of new features. The version of the algorithm used for the experiments described in this paper (PrismTCS) will be described in detail in Section 5.1.

2.4 Dealing With Clashes

Clashes occur during the classification tree/rule generation process whenever an algorithm is presented with a subset of the training set which

contains instances with more than one classification, but which cannot be processed further. Such a subset is known as a 'clash set'.

The principal cause of clashes is the presence of inconsistent data in the training set, where two or more instances have the same attribute values but different classifications. In such cases, a situation will inevitably occur during tree/rule generation where a subset with mixed classifications is reached, with no further attributes available for selection.

If this occurs at step (a) of the TDIDT algorithm the simplest way of dealing with the clash (and the method assumed in this paper) is to treat all the instances in the clash set as if they belong to the class that contains the largest number of them and generate a rule (i.e. a branch of the classification tree) accordingly.

Clashes in Prism are resolved in the following way. First the largest class for the subset of instances in the clash set is determined. If this largest class is the one for which a rule is being generated, the induced rule is treated as complete. If not, the rule is discarded and those instances in the clash set that have that classification are removed from the training set.

2.5 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.

2.6 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 [6]. 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 and a clash had occurred (see Section 2.4).

Reference [7] 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.

3. THE J-MEASURE

3.1 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 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:

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

The maximum value of the J-measure can be shown to be $\frac{\log_2 e}{e}$, which is approximately 0.5307 bits.

Further information on the J-measure and its uses is given in [8].

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.

3.2 A J-measure Interpretation of Overfitting

The results given in [7] strongly suggest that, beyond a certain point, adding further terms to rules 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.

Reference [8] gives a formula for Jmax, an upper bound on the J value of any rule that can be obtained by *specialising* a given rule by adding further terms. 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)

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 left-hand side of the rule after a single term. This would have led to more misclassified instances for the training data, but may have led to better predictive accuracy on unseen data.

4. USING J-PRUNING WITH TDIDT

4.1 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 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 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 an effective heuristic is 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 1 shows the results obtained using J-pruning with a variety of datasets and the comparative figures for unpruned rules. The results were obtained using 10-fold cross-validation [7] in each case. The Information

Gain attribute selection criterion is used throughout. The number of attributes and the percentage size of the largest class are also included for information.

Table 1. Comparison of Unpruned and J-pruned Rules for TDIDT

Dataset	Largest Class %	No. of Attributes	Number of Rules	Accuracy (%)	Number of Rules	Accuracy (%)
			No Pruning		With J-pruning	
agaricus_lepiota	52	22	24.0	100.00	16.0	99.41
breast-cancer	66	9	93.2	89.84	66.5	91.27
chess	95	7	20.0	99.38	6.2	96.44
contact_lenses	88	5	16.0	92.55	8.3	92.64
genetics	52	60	357.4	89.22	25.9	78.15
lens24	63	4	8.4	70.00	6.2	70.00
monk1	50	6	37.8	83.91	14.4	67.76
monk2	62	6	88.4	43.82	21.3	55.66
monk3	51	6	26.5	86.92	12.5	90.90
soybean	13	35	106.9	88.74	29.6	76.87
vote	61	16	29.2	91.67	11.1	94.00
zoo	41	16	13.8	97.00	10.9	89.18
AVERAGE			68.5	86.09	19.1	83.52

The reduction in the number of rules is clearly considerable for many of the datasets (e.g. 357.4 to 25.9 for *genetics* and from 106.9 to 29.6 for *soybean*). On average the number of rules is reduced from 68.5 to only 19.1 for the 12 datasets. 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 5 of the datasets, lower for 6 and unchanged for one (the smallest dataset, *lens24*). On average the predictive accuracy is reduced by 2.5% but this may not be as important in practice as the substantial reduction in the number of rules and corresponding gain in simplicity of the classification rules generated.

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 and are difficult to overcome in that framework.

The method prunes a branch as soon as a node is generated at which the J value is less than that at its parent. It would have been better if that branch had been pruned at the parent node instead, but doing so would also have removed all other descendant branches, possibly with damaging results.

The use of a decision tree representation for rules has previously been identified as itself a major cause of overfitting [3,4]. An example is given in

[3] of two rules with no attribute in common which are equivalent to a complex decision tree almost all branches and terms of which are redundant.

It may prove more effective to incorporate J-pruning or other pre-pruning techniques into an algorithm such as Prism, which generates classification rules directly rather than through the intermediate representation of a decision tree.

5. ADDING J-PRUNING TO PRISM

5.1 PrismTCS

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. In that form the training set is restored to its original state before the rules are generated for each class, thus requiring the full training set to be processed once for each of the classes.

(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

Figure 2. The PrismTCS Rule Generation Algorithm

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.

5.2 Using the J-measure to Prune Prism Classification Rules

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 and the clash handling method described in Section 2.4 is invoked. This method is known as *J-pruning*.

The following table shows the number of rules generated and the corresponding predictive accuracy of PrismTCS with and without J-pruning for twelve datasets. The results were obtained using 10-fold cross-validation in each case.

It can be seen that the number of rules generated for the unpruned algorithm is on average significantly smaller for PrismTCS than for TDIDT. Nevertheless the use of J-Pruning with PrismTCS reduces 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 is larger for the J-pruned rule sets in seven cases and smaller for only three. On average there is a small increase in predictive accuracy despite the substantially reduced number of rules.

Table 2. Comparison of Unpruned and J-pruned Rules for PrismTCS

Dataset	Largest Class %	No. of Attributes	Number of Rules	Accuracy (%)	Number of Rules	Accuracy (%)
			No Pruning		With J-Pruning	
agaricus_lepiota	52	22	11.9	100.00	11.9	100.00
breast-cancer	66	9	37.8	93.99	33.2	95.28
chess	95	7	7.9	99.38	3.6	97.98
contact_lenses	88	5	8.6	88.73	6.6	89.82
genetics	52	60	87.7	90.88	25.1	93.48
lens24	63	4	6.5	55.00	6.4	58.33
monk1	50	6	14.0	87.56	10.8	88.46
monk2	62	6	37.3	50.18	16.9	59.71
monk3	51	6	15.2	85.32	13.6	83.65
soybean	13	35	64.9	90.63	51.5	92.98
vote	61	16	21.6	92.33	11.0	91.00
zoo	41	16	10.4	92.09	9.7	92.09
AVERAGE			27.0	85.51	16.69	86.90

6. CONCLUSIONS

This paper has demonstrated the potential value of using the information-theoretic J-measure as the basis for reducing overfitting by pre-pruning rules as they are generated. The J-pruning technique illustrated works well in practice for a range of datasets, both with the decision tree representation of TDIDT and even more successfully with PrismTCS, an algorithm that directly generates classification rules from examples. Unlike many other possible measures, the J-measure has a sound theoretical foundation as a measure of the information content of rules.

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

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

[7] 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.

[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