J-measure based hybrid pruning for complexity reduction in classification rules

J-measure Based Hybrid Pruning for Complexity Reduction in Classification Rules

HAN LIU[1], ALEXANDER GEGOV[1] and FREDERIC STAHL[2]
[1]School of Computing
University of Portsmouth
Buckingham Building, Lion Terrace, PO1 3HE Portsmouth
UNITED KINGDOM
Han.Liu@port.ac.uk
http://uk.linkedin.com/pub/han-liu/24/a13/488
Alexander.Gegov@port.ac.uk
http://www.port.ac.uk/departments/academic/comp/staff/title,3828,en.html
[2]School of Systems Engineering
University of Reading
Po Box 225 Whiteknights, Reading, RG6 6AY
UNITED KINGDOM
F.T.Stahl@reading.ac.uk
http://fredericstahl.wordpress.com

Abstract: - Prism is a modular classification rule generation method based on the 'separate and conquer' approach that is alternative to the rule induction approach using decision trees also known as 'divide and conquer'. Prism often achieves a similar level of classification accuracy compared with decision trees, but tends to produce a more compact noise tolerant set of classification rules. As with other classification rule generation methods, a principle problem arising with Prism is that of overfitting due to over-specialised rules. In addition, over-specialised rules increase the associated computational complexity. These problems can be solved by pruning methods. For the Prism method, two pruning algorithms have been introduced recently for reducing overfitting of classification rules - J-pruning and Jmax-pruning. Both algorithms are based on the J-measure, an information theoretic means for quantifying the theoretical information content of a rule. Jmax-pruning attempts to exploit the J-measure to its full potential because J-pruning does not actually achieve this and may even lead to underfitting. A series of experiments have proved that Jmax-pruning may outperform J-pruning in reducing overfitting. However, Jmax-pruning is computationally relatively expensive and may also lead to underfitting. This paper reviews the Prism method and the two existing pruning algorithms above. It also proposes a novel pruning algorithm called Jmid-pruning. The latter is based on the J-measure and it reduces overfitting to a similar level as the other two algorithms but is better in avoiding underfitting and unnecessary computational effort. The authors conduct an experimental study on the performance of the Jmid-pruning algorithm in terms of classification accuracy and computational efficiency. The algorithm is also evaluated comparatively with the J-pruning and Jmax-pruning algorithms.

Key-Words: - Data Mining, Machine Learning, Classification Rules, J-pruning, Jmax-pruning, Jmid-pruning, if-then rules, overfitting, J-measure

1 Introduction

The automatic induction of classification rules has been increasingly popular in commercial applications such as rule based expert systems, decision making systems. In this context, classification rule generation methods can be divided into two categories - 'divide and conquer' and 'separate and conquer'. The 'divide and conquer' approach, which is also known as the Top-down Induction of Decision Trees (TDIDT) [1], can be traced back to 1960s [2]. This approach induces classification rules in the intermediate form of a decision tree such as ID3, C4.5 and C5.0. The 'separate and conquer' approach, which is also known as covering approach, can be also traced back to 1960s [3]. This approach generates if-then rules directly from training instances such as Prism. A series of experiments have shown that Prism achieves with a similar level of classification

accuracy compared with TDIDT and can even outperform TDIDT in some cases [11].

As mentioned in [4], a principle problem that arises with most methods for generation of classification rules is the overfitting of training data. In some cases, this may result in the generation of a large number of complex rules. This may not only increase the computational cost but also lower the classification accuracy in predicting further unseen instances.

A suitable way to reduce overfitting of classification rules is by simplifying rules using pruning strategies. Pruning methods can be subdivided into two categories - Pre-pruning [5], which truncate rules during rule generation, and Post-pruning [5], which generate a whole set of rules and then remove a number of rules and rule terms, by using statistical or other tests [4]. For the Prism method, two pruning algorithms have been introduced recently. These are J-pruning, which is completely based on pre-pruning as the pruning action is taken during rule generation [6], and Jmax-pruning, which is a hybrid between pre-pruning and post-pruning as each rule is post-pruned before the next rule is generated [6].

The structure of this paper is as follows. Section 2 reviews the Prism method and identifies its limitations. It also discusses which of these limitations have been overcome and in what way by J-pruning and Jmax-pruning in order to identify the potential ways of resolving these issues. Section 3 introduces a novel pruning algorithm, called Jmid-pruning. This algorithm is a modified version of the J-measure based pruning and it addresses some common issues such as clashes, tie-breaking and continuous attributes that arise in classification tasks. Section 4 describes the setup of an experimental study and presents results in classification accuracy and computational efficiency. Section 5 evaluates the Jmid-pruning algorithm in comparison with J-pruning and Jmax-pruning in terms of classification accuracy and computational efficiency. Section 6 summarises the contribution of this work to real world applications and highlights related future research directions.

2 Related Work

As mentioned in Section 1, most classification rule generation methods may lead to overfitting of training data, which results in lower classification accuracy and higher computational cost. This section describes how J-pruning and Jmax-pruning can overcome the overfitting problems of Prism and also discusses other related issues of these two pruning algorithms that are discussed in the subsequent sections.

2.1 Prism Method

The Prism method was introduced by Cendrowska in [7] and the basic idea of the underlying Prism algorithm is illustrated in Figure 1. This algorithm is primarily aimed at avoiding the generation of complex rules with many redundant terms [4] such as the 'replicated subtree problem' that arises with decision trees as illustrated in Figure 2.

Execute the following steps for each classification (*class= i*) in turn and on the original training data *S*:
1. *S'=S*.
2. Remove all instances from *S'* that are covered from the rules induced so far. If *S'* is empty then stop inducing further rules
3. Calculate the probability from *S'* for *class=i* for each attribute-value pair.
4. Select the *attribute-value pair* that covers *class= i* with the highest probability and remove all instances from *S'* that comprise the selected *attribute-value pair*
5. Repeat 3 and 4 until a subset is reached that contains only instances of *class= i*. The induced rule is then the conjunction of all the *attribute-value pairs* selected.
Repeat 1-5 until all instances of *class i* have been removed

*For each rule, no one attribute can be selected twice during generation

Fig.1 Basic Prism algorithm [5]

The original Prism algorithm cannot directly handle continuous attributes as it is based on the assumption that all attributes in a training set are categorical. When continuous attributes are actually present in a dataset, these attributes should be discretised by preprocessing the dataset prior to generating classification rules [5, 6]. In addition, Bramer's Inducer Software handles continuous attributes as described in [5, 6, 8] and in Section 3.

On the other hand, the original Prism algorithm does not take clashes into account, i.e. a set of instances in a subset of a training set that are identical apart from being assigned to more than one class but cannot be separated further [6, 8]. However, the Inducer Software implementation of Prism can handle clashes and the strategy of handling a clash is illustrated in Figure 3.

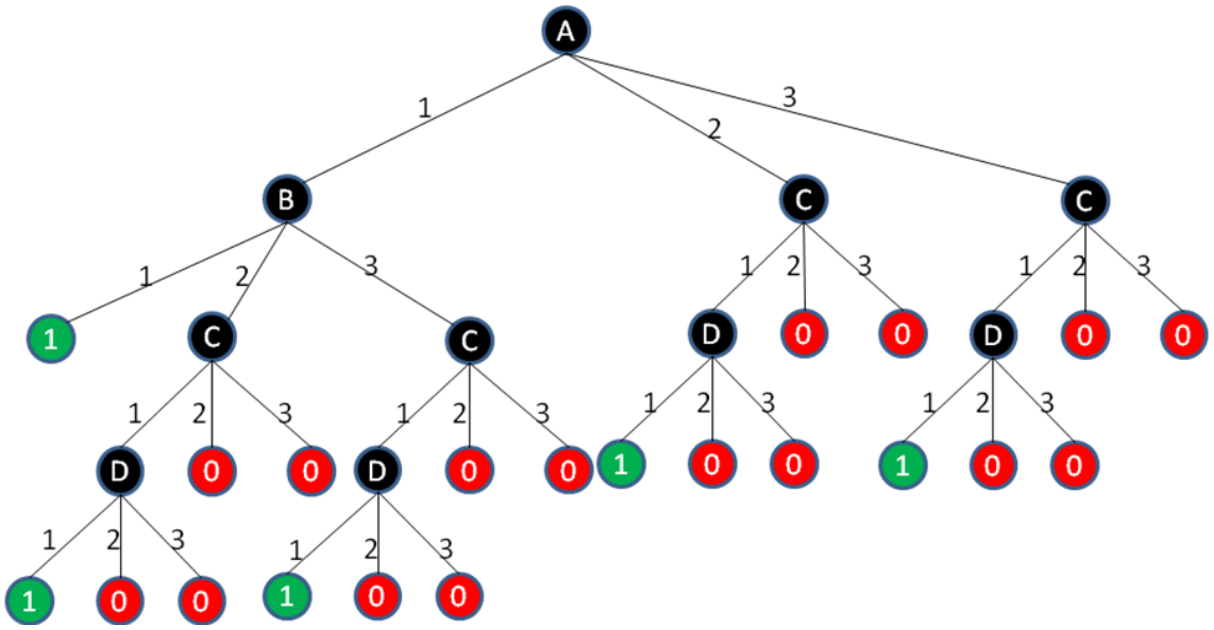

Fig.2 Cendrowska's replicated subtree example [8, 9]

Another problem that arises with Prism is tie-breaking, i.e. there are two or more attribute-value pairs which have equal highest probability in a subset. The original Prism algorithm makes an arbitrary choice in step 4 as illustrated in Figure 1 whereas the Inducer Software makes the choice using the highest total target class frequency [5].

If a clash occurs while generating rules for class i:
1. Determine the majority class for the subset of instances in the clash set.
2. If this majority class is *class i*, then compute the induced rule by assigning all instances in the clash set to class i. If it is not, discard the whole rule.
3. If the induced rule is discarded, then all instances that match the target class should be deleted from the training set before the start of the next rule induction. If the rule is kept, then all instances in the clash set should be deleted from the training data.

Fig.3 Dealing with clashes in Prism

In addition, Bramer pointed out that the original Prism algorithm always deletes instances covered by those rules generated so far and then restores the training set to its original size after the completion of rule generation for class *i* and before the start for class *i+1*. This undoubtedly increases the number of iterations resulting in high computational cost [9]. For the purpose of increasing the computational efficiency, a modified version of Prism, called PrismTCS, was developed by Bramer [14]. PrismTCS always chooses the minority class as the target class pre-assigned to a rule being generated as its consequence. Besides this, it does not reset the dataset to its original state and introduces an order to each rule for its importance [6, 8, 9]. Therefore, PrismTCS is not only faster in generating rules compared with the original Prism, but also provides a similar level of classification accuracy [6, 8].

Apart from the overfitting problem mentioned in Section 1, Prism has further limitations. For example, it may result in underfitting of training data due to over-discarding of rules when clashes occur during rule generation. It is mentioned in [5] that clashes may occur in the following two ways:

- One of the instances has at least one incorrect record for its attribute values or classification [5].
- The clash set has all instances correctly recorded but it is not possible to discriminate between or amongst them on the basis of the attributes recorded. So, it may be required to examine extra attributes not recorded in the training set [5].

If the reason that clashes occur is due to the presence of noise, the fact that Prism prefers to leave instances unclassified rather than to give a wrong classification may lead to good performance with noisy data. However, in the absence of noise, Prism may generate a rule set that *underfits* training data due to *over-discarding* of rules. Bramer mentioned in [13] that classification tasks are normally aimed at finding all relevant rules to make

good classifications rather than, say, finding the best 20 rules like Association Rule Mining. In addition, over-discarding of rules may also increase the redundant computational effort as the method takes time to generate a rule which is discarded later. This may result in a considerable waste of computational resources.

On the basis of above considerations, it is worth looking at how J-pruning and Jmax-pruning manage to reduce overfitting while also attempting to come up with a solution that avoids underfitting and redundant computational effort at the same time.

2.2 Pruning Algorithms

A working hypothesis taken from [13] is that rules with high information content (value of J-measure) are likely to have a high level of predictive accuracy. This subsection aims to review how J-pruning and Jmax-pruning use the J-measure to reducing overfitting in classification rules. It also examines whether these two pruning algorithms exploit the J-measure to its full potential or if there are potential improvements.

2.2.1 J-measure

J-measure was introduced by Smyth and Goodman [10] who strongly justified the use of the J-measure as an information theoretic means of quantifying the information content of a rule.

According to the notation of [10], given a rule of the form *IF Y = y THEN X = x* which can be measured in bits and is denoted by *J(X, Y=y)*.

$$J(X; Y = y) = p(y) \cdot j(X; Y = y) \quad (1)$$

$J(X; Y = y)$ is essentially a product of two terms as follows:

- $p(y)$, the probability that the left hand side of the rule (hypothesis) will occur.
- $j(X;Y = y)$, which is called the j-measure (with a lower case j) and measures the goodness-of-fit of a rule.

The j-measure, also known as the *cross-entropy*, is defined as:

$$j(X;Y = y) = p(x \mid y) \cdot log_2(p(x \mid y)p(x)) + (1 - p(x \mid y)) \cdot log_2((1 - p(x \mid y))(1 - p(x))) \quad (2)$$

The value of *cross-entropy* depends upon two values [5]:

- $p(x)$: the probability that the consequence (right hand side) of the rule will be matched if there is no other information given. This is known as *a priori* probability of the rule consequence.
- $p(x \mid y)$: the probability that the consequence of the rule is matched if the given antecedents are satisfied. This is also read as *a posterior* probability of x given y.

Bramer mentioned in [4, 5] that the J-measure has two very helpful properties related to upper bounds as follows:

- It can be shown that $J(X; Y = y) \leq p(y) \cdot log_2 (1/p(y))$. The maximum point of this expression can be found at $p(y) = 1/e$. This can derive a maximum value, is $(log_2 (e) \cdot (1/e)$, i.e. approximately 0.5307 bits.
- More importantly, it can be proven that the value of the J-measure is never higher than the upper bound value illustrated in equation (3) whenever a rule is specialised by adding further terms to its left hand side.

$$Jmax = p(y) \cdot max \{p(x \mid y) \cdot log_2 (1/p(x)), (1 - p(x \mid y)) \cdot log_2 (1 / 1 - p(x))\} \quad (3)$$

Thus, there are no benefits to be gained by adding further terms to a rule when the value of the J-measure of this rule is equal to its corresponding Jmax-value. The application of Jmax is illustrated in Section 3.

2.2.2 J-pruning

As mentioned in Section 1, J-pruning, based on the J-measure, is a pre-pruning method because the pruning action is taken during rule generation. It was developed by Bramer [4] and its basic idea is illustrated in Figure 4.

```
Rule r = new Rule;
Boolean rule_Incomplete = true;
Do While (rule_Incomplete){
   Term t = generate new term;
    compute J_value of r if appending t;
    IF(r.current_J_value > J_value){
      do not append t to r;
      invoke clash handling for r;
      rule_Incomplete = false;
    }ELSE{
       r.current_J_value = J_value;
       append t to r;
    }
}
```

Fig.4 J-pruning for Prism algorithms

J-pruning achieved relatively good results as indicated in [14]. However, Stahl and Bramer pointed out in [6, 8] that J-pruning does not exploit the J-measure to its full potential as this method immediately stops the generation process as soon as the J-measure goes down after a new term is added

to the rule. In fact, it is possible that the J-measure may go down and go up again after further terms are added to the rule. This indicates the pruning action may be taken too early. The fact that J-pruning may achieve relatively good results could be explained by the assumption that it does not happen very often that the J-value goes down and then goes up again. It also indicates that J-pruning may even result in underfitting due to over-generalised rules. This is because the pruning action may be taken too early resulting in too general rules generated to have high predictive accuracy. This motivated the development of a new pruning method, called Jmax-pruning, which was proposed by Stahl [6, 8], in order to exploit the J-measure to its full potential.

2.2.3 Jmax-pruning

As mentioned in Section 1, Jmax-pruning can be seen as a hybrid between pre-pruning and post-pruning. However, with regard to each generated rule, each individual rule is actually post-pruned after the completion of the generation for that rule. However, with respect to the whole classifier (whole rule set) it is pre-pruning approach as there is no further pruning required after all rules have been induced.

The basic idea of Jmax-pruning is illustrated in Figure 5.

```
Rule r = new Rule;
Boolean rule_Incomplete = true;
term_index = 0;
Do While (rule_Incomplete){
    Term t = generate new term;
    term_index++;
    append t to r;
    compute J_value of r;
    IF(J_value > best_J_Value){
       best_J_Value = J_Value;
      best_term_index = term_index;
    }
    IF(No more rule terms can be induced){
       cut r back to rule best_term_index;
       invoke clash handling for r;
       rule_Incomplete = false;
    }
}
```

Fig.5 Jmax-pruning for Prism algorithms

A series of experiments have shown that Jmax-pruning outperforms J-pruning in some cases and perform the same as J-pruning in other cases [6, 8]. Strictly speaking, the situation that Jmax-pruning outperforms J-pruning may arise when there are two or more rule terms within the same rule that have a local maximum for the J-value. J-pruning regards the change trend of the J-value and thus stops inducing further rule terms once the first local maximum is reached, even though this may not be the global maximum. In this case, J-pruning would make Prism generate a rule with a J-value equal to the first local maximum whereas Jmax-pruning would provide the global maximum as the J-value of the rule generated. The cases in which Jmax-pruning performs the same as J-pruning could be explained by the assumption that there is only one local maximum for the J-value, or that the first local maximum encountered is also the global maximum.

However, all the explanations above are potentially based on the assumption that a rule is being discarded due to dealing with a clash after a pruning action is taken does not happen very often. As mentioned in Section 2.1, Prism prefers to discard a rule rather than assign it to a majority class when a clash occurs. Therefore, it may even lead to underfitting of the induced rule set if a pruning method attempts to reduce the overfitting by pruning rules but unfortunately results in discarding rules. If this case is taken into consideration, then it may be possible that J-pruning outperforms Jmax-pruning if J-pruning actually simplifies a rule whereas Jmax-pruning makes the simplified rule get discarded. However, it is not very likely to happen. In addition, Jmax-pruning does not take into consideration tie-breaking of the J-value, i.e. there are two or more terms with equal highest J-value. In other words, in terms of the change trend for the J-value during rule generation, there may be two or more global maxima for the J-value for different rule terms. In this case, a different decision in choosing the term with the highest J-value may end up with a different outcome. For example, if there are two terms that could be chosen and the first one is chosen, then the rule is being discarded. Otherwise, the rule is being kept. In order to prevent underfitting from happening, as many rules as possible rules that make good classifications should be kept. However, the strategy of Jmax-pruning is to simply choose the first term with highest J-value and thus it may not only result in a loss of accuracy but also increase the unnecessary computational efforts if it ends up with the rule being discarded on these grounds.

On the other hand, Jmax-pruning may be computationally relatively expensive as each rule generated by this method is post-pruned. The pruning action could be taken earlier during the rule generation and thus speed up the rule generation. This could be achieved by making use of the Jmax value as introduced in Section 2.2.1.

Let us take the lens24 dataset for illustration; one of the rules generated is shown below [13]:

If tears=2 and astig=1 and age=3 and specRx =1 then class= 3;

After adding the four terms subsequently, the corresponding J and Jmax values change in the trend as follows:

If tears=2 then class=3; (J=0.210, Jmax=0.531)

If tears=2 and astig=1 then class=3; (J=0.161, Jmax=0.295)

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 after adding the second and third term to this rule that both the J-value (0.004) and Jmax value (0.059) are lower than the J-value (0.161) after adding the second term. In this case the rule generation can be stopped by taking pruning action after the third term is added. This is because the J-value is unable to get higher than 0.161, it could only go up to 0.059 by adding any further terms and thus cannot get higher than the highest J-value (0.210). By looking at the J-value after the fourth term is added, it is obvious that the J-value cannot get higher than 0.059 by adding further rule terms, in fact it is decreasing to 0.028 after adding the 4th rule term. This shows that Jmax-pruning increases the redundant computational effort in such cases.

On the basis of considerations above, the authors propose a novel pruning method which can not only reduce overfitting of classification rules but can also avoid underfitting and unnecessary rule term inductions and their associated computational overhead.

3 Jmid-pruning

As mentioned in section 2.2, neither J-pruning nor Jmax-pruning exploit the J-measure to its full potential and they may lead to underfitting. In addition, Jmax-pruning is computationally relatively expensive. Therefore, the authors propose a novel pruning algorithm that avoids underfitting and unnecessary rule term inductions while at the same time rules are being pruned for reducing overfitting.

3.1 Essence

The Jmid-pruning is a modified version of the J-measure based pruning algorithm Jmax-pruning. The basic concept of this algorithm is illustrated in Figure 6:

```
Rule r = new Rule;
Boolean rule_Incomplete = true;
term_index = 0;
Do While (rule_Incomplete){
    Term t = generate new term;
    term_index++;
    append t to r;
    compute J_value of r;
    IF(J_value > best_J_Value){
       best_J_Value = J_Value;
      best_term_index = term_index;
      record current_marjority_class;
    }
    IF(r.current_J_value > J_value){
     compute Jmax_value of r;
     IF(best_J_value> Jmax_value){
      do not append t to r;
      cut r back to rule best_term_index;
      invoke clash handling for r;
      rule_Incomplete = false;
      } ELSE{
       r.current_J_value = J_value;
       append t to r;
      }
    } ELSE{
       r.current_J_value = J_value;
       append t to r;
   }
    IF(No more rule terms can be induced){
       cut r back to rule best_term_index;
       invoke clash handling for r;
       rule_Incomplete = false;
    }
}
```

Fig.6 Jmid-pruning for Prism algorithms

Jmid-pruning can also be seen as a hybrid between pre-pruning and post-pruning. This is because each rule may be either pre-pruned or post-pruned. For example, as mentioned in Section 2.2.3, it may be possible that both J and Jmax values are lower than the J-value given at the last stage of rule generation. If this is the case, then the generation would be stopped immediately. Thus, the rule is actually pre-pruned. Otherwise, the rule would be post-pruned.

The Jmid-pruning algorithm illustrated in Figure 6 does not take the tie-breaking problem into consideration. Tie-breaking may happen during rule generation in two ways:

1) Let the current rule's last added rule term be denoted t_i, and the previously added rule term be denoted t_(i-1). Then a tie break happens if J-value at t_i is less than that at t_(i-1) and Jmax-value at t_i equals J-value at t_(i-1). It is also illustrated by an example (**Rule 1**) below.
2) When there is more than one term with the highest J-value for one rule during rule generation. It is also illustrated by an example (**Rule 2**) below.

As mentioned in section 2.2.3, tie-breaking should be carefully dealt with in order to avoid the case whereby the pruning algorithm actually attempts to reduce overfitting by pruning rules but unfortunately discards the whole rule. The authors propose to keep all rules that are capable of making good classifications if adequate. Therefore, the basic idea of dealing with this tie-breaking is to see if the rule can be kept when choosing a particular term with the highest J-value as the last (best) term of the simplified rule for pruning criteria. Therefore, the first case for tie-breaking mentioned above can be handled as follows:

1) To check if the currently best term (with the highest J-value) is chosen as the last rule term of the generalized rule and then a clash set covered by this rule can have a majority class which is also the target class of this rule.
2) If so, then the rule generation could be stopped immediately. Otherwise, the rule generation would be carried out as normal generation procedure.

For example, let us see the rule:
Rule 1: If x=1 and y=1 and z=1 then class=1;

After adding first term:
If x= 1 then class= 1; (J= 0.33, Jmax= 0.55)

After adding second term:
If x=1 and y=1 then class=1; (J= 0.21; Jmax=0.33)

In this case, the Jmid-pruning would aim to check if the incomplete rule: if x=1 then class=1; covers a clash set which had 'class=1' as the majority class. If this is the case, then the generation would be stopped immediately and have the rule: if x=1 then class=1; as the finally simplified rule to be applied for predicting unseen instances. Otherwise, the rule generation would be carried out as normal.

For the second case, the tie-breaking could be handled as follows:

1) To check if the term corresponded by the first global maximum of J-value can retain the corresponding rule.
2) If so, then choose the term as the last (best) term of the generalized rule. Otherwise, check the next term corresponded by a global maximum in this way and so on.

For example, let us see the rule:
Rule 2: If a=1 and b=1 and c=1 and d=1 then class=1;

After adding the terms subsequently, the J-value changes in the way as below:
If a=1 then class=1; (J-value= 0.6)
If a=1 and b=1 then class=1; (J-value=0.4)
If a=1 and b=1 and c=1 then class=1; (J-value= 0.6)
If a=1 and b=1 and c=1 and d=1 then class=1; (J-value=0.5)

In this case, the Jmid-pruning method would aim to first check if the incomplete rule: if a=1 then class=1; covers a clash set which has the target class of this rule as the majority class. If it does, the method would choose the rule: if a=1 then class=1; as the finally simplified rule. Otherwise, the method will continue to check the rule: if a=1 and b=1 and c=1 then class=1; in the same way and so on.

3.2 Justification

The proposed Jmid-pruning aims to avoid underfitting and unnecessary computational effort. In fact, J-pruning and Jmax-pruning do not actually make use of Jmax to measure the potential search space of gaining benefits.

Let us get back to the example about the lense24 dataset as mentioned in Section 2.2.3. There is a rule generated as follows:
If tears=2 and astig=1 and age=3 and specRx =1 then class= 3;

After adding the four terms subsequently, the corresponding J and Jmax values change in the trend as follows:

If tears=2 then class=3; (J=0.210, Jmax=0.531)

If tears=2 and astig=1 then class=3; (J=0.161, Jmax=0.295)

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)

In this example, all of the three algorithms would provide the same simplified rule that is: if tears=2 then class=3; this is because the highest J-value has been given after adding the first term (tears=2). However, the computational efficiency would be different in the three methods. J-pruning would decide to stop the generation after the second term (astig=1) is added as the J-value goes down after the second term (astig=1) is added. In contrast, Jmax-pruning would stop when the rule is complete. In other words, the generation would be stopped after the fourth (last) term is added and then the terms (astig=1, age=3 and specRx=1) will be removed In addition, Jmid-pruning would decide to stop the generation after the third term is added as the value of Jmax (0.295) is still higher than the J-value (0.210) given after the first term (tears=2) is added although its corresponding J-value (0.161) decreases; however, the generation should be stopped after the third term (age=3) is added as both J (0.004) and Jmax (0.059) values are lower than the J-value (0.161) computed after the second term (astig=1) is added although the J-value could still increase up to 0.059.

On the basis of the description above, J-pruning would be the most efficient and Jmid-pruning is more efficient than Jmax-pruning. However, it seems J-pruning may prune rules too early in some cases as mentioned in Section 2.2.2. For example, one of the rules generated from the Soybean dataset [6, 8] is:
If temp= norm and same-lst-sev-yrs= whole-field and crop-hist= same-lst-two-yrs) then class=frog-eye-leaf-spot;

First term:
If temp= norm then class=frog-eye-leaf-spot;
(J= 0.00113, Jmax=0.02315)

Second term:
If temp= norm and same-lst-sev-yrs= whole-field then class=frog-eye-leaf-spot;
(J=0.00032, Jmax=0.01157)

Third term:
If temp= norm and same-lst-sev-yrs= whole-field and crop-hist= same-lst-two-yrs) then class=frog-eye-leaf-spot;
(J=0.00578, Jmax=0.00578)

In this case, both Jmax-pruning and Jmid-pruning would normally stop the generation when the rule is complete and take the complete rule: If temp= norm and same-lst-sev-yrs= whole-field and crop-hist= same-lst-two-yrs) then class=frog-eye-leaf-spot; as the final rule with the highest J-value (0.00578). In contrast, J-pruning would stop the generation after the second term (same-lst-sev-yrs= whole-field) is added and take the rule: If temp= norm then class=frog-eye-leaf-spot; as the final rule with a lower J-value (0.00113 instead of 0.00578).

The other potential advantage of Jmid-pruning in comparison with Jmax-pruning is that it may keep more rules than Jmax-pruning when tie-breaking on J-value happens as mentioned in Section 2.2.3 and Section 3.1. In this way, Jmid-pruning is better in avoiding underfitting of rule sets.

3.3 Dealing with Continuous Attributes

As mentioned in Section 2.1, the strategy of handling continuous attributes introduced by Bramer has been implemented in the Inducer Software [12]. The basic idea of this strategy is to sort a continuous attribute first. For example, let attribute x have these values: 3, 5, 6, 8. Then the attribute is scanned for these values in either ascending or descending order. For each attribute value, e.g. 5, two tests are used: $p(class=i|x<5)$ and $p(class=i|x\geq5)$. The largest conditional probability would be chosen for comparison with the conditional probabilities of candidate rule terms generated from other attributes.

3.4 Dealing with Clashes

As mentioned in Section 2.1, a clash set is a subset of a training set with instances that can be assigned to more than one class whereby the subset cannot be split further. In other words, the left-hand side of a complete rule may be mapped to more than one classification. The authors follow the approach introduced by Bramer in [11] that is to check if the target class pre-assigned to the current rule is also the majority class in the clash set. If it is then the rule is included in the rule set. Otherwise, the current rule is discarded and all instances that match the target class are deleted from the training set in order to prevent the same case arising during the rule generation all over again.

3.5 Conflict Resolution

There is a problem that arises with most ‘separate and conquer’ rule generation methods. This problem is known as classification conflict, i.e. one unseen instance may be covered by two or more rules with different right-hand sides (classifications) in the rule set.

Let us see the following example:
Rule 1: if x=1 and y=1 then class= 1;
Rule 2: if z=1 then class=0;

So, what should the classification be for an instance with x=1, y=1 and z=1? One rule gives class 1 and the other one gives class 0. We need to choose only one classification to classify the instance. Bramer introduced in [5] the 'take the first rule that fires' strategy that requires the rule generation method to generate the most important rules first. In this paper, the authors propose to apply Jmid-pruning to PrismTCS. As mentioned in Section 2.1, PrismTCS introduces an order to a rule for its importance. Thus, the authors choose the 'take the first rule that fires' strategy for conflict resolution.

4 Experimental Results

In this experimental study, the authors use PrismTCS as a rule generation method and J-pruning, Jmax-pruning and Jmid-pruning, respectively, to prune the rules generated by Prism TCS. In addition, all datasets used are retrieved from the UCI repository [15].

The authors compare classification accuracy and computational efficiency performed by PrismTCS with J-pruning, Jmax-pruning and Jmid-pruning respectively. For estimating the classification accuracy the authors used cross-validation [5]. Therefore the data is split into complementary subsets, whereas one subset is used as test set and the remaining subsets are used as training set. Multiple rounds are performed using different test sets and the results are averaged. With regards to efficiency, the authors choose the whole dataset as the training set to train the rule set and then use the same dataset for testing. This is because the efficiency performed in simulation stage is only concerned with the computation time taken to make a decision in assigning a class to an unseen instance but no matter what decision (class). Moreover, in this case (when the whole data set that is available is used) it would be larger and more representative in relation to the evaluation of efficiency. The efficiency performed in modeling stage is measured against the number of rules, the number of terms per rule and the number of discarded (redundant) rules. The reason for counting the number of discarded rules is that these rules use the Prism method as a computational resource and thus increase the associated cost. In this case, the number of backward steps should be counted in order to prove that Jmid-pruning may stop the rule generation earlier than Jmax-pruning. The number of backward steps is equal to the number of terms removed. For example, let us look again at the lense 24 dataset:

One of the rules generated is:

If tears=2 and astig=1 and age=3 and specRx =1 then class= 3; (number of terms: 4)

The final simplified rule would be:
If tears=2 then class=3; (number of terms: 1)

Therefore, the number of backward steps would be *4-1=3* in this case. For a rule set, the authors count the total number of backward steps for all rules. The number of redundant iterations would be twice number of backwards steps, because all discarded rule terms need to be induced and then be removed, which requires some computation as also.

The accuracy and efficiency are illustrated in Table 1-3.

5 Evaluation

The results in Table 1 show that Jmid-pruning may perform same as or even better than J-pruning and /or Jmax-pruning in terms of classification accuracy. In the four datasets namely 'vote', 'weather', 'lung-cancer' and 'ionosphere', all the three methods perform with the same accuracy. This could be explained by the fact that there is only one local maximum (or two or more local maxima but only one global maximum which is also the first local maximum) for the J-value in terms of the relationship between the J-value and rule complexity. In addition, if there is more than one global maximum but the first one which is also the first local maximum corresponds a rule term which can retain the corresponding rule after dealing with a clash, then it is also possible for the three methods to perform with the same accuracy. However, there are three cases on the contact-lenses, lense24 and breast-cancer datasets for which Jmax-pruning and Jmid-pruning perform with the same accuracy but J-pruning performs worse than the other two methods. The possible explanation for this is that there is more than one local maximum point but the first one is not the global maximum and the only one global maximum or the first one of them corresponds a rule term which can retain the rule. Besides, there may still be special cases which show that J-pruning performs better than Jmax-pruning but worse than Jmid-pruning. This is possibly due to the fact that Jmax-pruning discards one or more rules when dealing with clashes and results in underfitting

although J-pruning retains rules with probably lower J-values. However, Jmid-pruning retains rules successfully by dealing with tie-breaking of the J-value (as mentioned in Section 3.1) in a heuristic way.

Table 1 Classification accuracy in percentage

Dataset	J-pruning	Jmax-pruning	Jmid-pruning
Vote	97%	97%	97%
Weather	83%	83%	83%
contact-lenses	80%	85%	85%
Lense24	67%	75%	75%
breast-cancer	55%	58%	58%
car	74%	74%	78%
lung-cancer	95%	95%	95%
Iris	67%	77%	82%
segment	53.1%	53.3%	53.8%
ionosphere	87%	87%	87%

Table 2 shows that PrismTCS with Jmid-pruning may generate a rule set with similar level of rule complexity or even fewer but more general rules in comparison with J-pruning and Jmax-pruning. However, Table 3 shows that Jmid-pruning may perform better compared with Jmax-pruning in terms of computational efficiency. It can be seen by looking at the number of backward steps that Jmid-pruning needs a smaller number of iterations than Jmax-pruning to make Prism stop generating rules. Therefore, Jmid-pruning is computationally more efficient. There is a special case with the 'car' dataset that shows that Jmax-pruning increases not only the number of iterations (backward steps) but also the number of discarded rules and thus it increases the unnecessary computational effort. In some cases, it may also increase the level of underfitting of rule sets to training data due to over-discarding of rules when dealing with clashes. On the other hand, in terms of the number of discarded rules, Table 3 does not actually show that Jmid-pruning may perform with slightly different computational efficiency in comparison with the other two pruning methods. This is because the experiments are done using relatively small datasets. However, there is still a case with the 'car' dataset showing that Jmax-pruning may make Prism discard slightly more rules in comparison with Jmid-pruning.

Table 2 Number of rules and terms per rule

Dataset	J-pruning		Jmax-pruning		Jmid-pruning	
	Count(rules)	Ave(terms)	Count(rules)	Ave(terms)	Count(rules)	Ave(terms)
Vote	2	2.5	5	4.2	2	2.5
weather	3	1.67	3	1.7	3	1.67
contact-lenses	3	1.67	3	1.67	3	1.67
Lense24	4	1.5	4	2.25	4	2.0
breast-cancer	8	1.125	7	1.0	7	1.0
car	3	1.0	3	1.0	3	1.0
lung-cancer	4	1.0	4	1.0	4	1.0
Iris	5	1.0	5	1.0	5	1.0
segment	11	1.09	13	1.69	10	1.0
ionosphere	2	1.0	2	1.0	2	1.0

6 Conclusion

This paper reviews the Prism rule generation method and two J-measure based pruning algorithms J-pruning and Jmax-pruning. It also identifies some limitations of Prism and the two associated pruning algorithms. A novel pruning algorithm called Jmid-pruning that is also based on the J-measure is proposed and validated. The experimental study shows that Jmid-pruning can avoid underfitting and redundant computational effort while also reducing overfitting of classification rules. In most cases, Jmid-pruning makes the Prism method generate a rule set with a similar level of complexity with J-pruning and Jmax-pruning. In some cases, Jmid-pruning may also cause Prism to generate fewer but more general rules than J-pruning and/or Jmax-pruning. In addition, in some special cases, Jmid-pruning stops rule generation earlier than Jmax-pruning as mentioned in Section 3.2. This avoids redundant effort in removing terms subsequently from a rule. However, the authors still need to validate the Jmid-pruning method using larger datasets in terms of the number of discarded rules. In addition, the Jmid-pruning may also work well with decision trees as J-pruning has also been applied successfully to decision tree induction with good results. Therefore, the authors will validate this pruning method in pruning classification rules generated by decision trees. Moreover, the possible relationship between the J-value and rule complexity has to be investigated further in order to find heuristic strategies for selecting the best J-measure based pruning algorithm for a particular case study. Furthermore, Jmid-pruning will be applied to some

well-known methods and validated in some particular application areas comparing other techniques used. For example, some techniques have been applied to integrated bio-systems for modelling and control such as fuzzy modelling [19], genetic algorithm [18, 20], asymptotic stabilization [17] and observer based on current estimation [21]. Jmid-pruning may potentially contribute to development of integrated bio-systems in accuracy and efficiency by exploiting more potential of some of those techniques above. For example, it may be applied to fuzzy rule based modelling for simplifying rules in order to reduce overfitting and computation costs. Besides, the four versions of PrismTCS (PrismTCS without pruning and with the three pruning methods introduced in this paper) may be embedded into four intelligent agents respectively working in cooperative strategies [16] that could be applied to a pool of different learning systems such as the Pocket Data Mining system [22, 23].

Table 3 Number of discarded rules and backward steps

Dataset	J-pruning	Jmax-pruning		Jmid-pruning	
	Count(discarded rules)	Count(discarded rules)	count(backward steps)	Count(discarded rules)	count(backward steps)
Vote	4	4	154	4	5
weather	1	2	3	1	1
contact-lenses	1	1	4	1	2
Lense24	2	1	5	2	3
breast-cancer	1	2	1	2	1
car	12	46	207	12	10
lung-cancer	0	0	0	0	0
Iris	0	0	0	0	0
segment	5	3	7	4	6
ionosphere	0	0	0	0	0

References:

[1] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufman, 1993.

[2] E.B. Hunt, P.J. Stone and J. Martin, *Experiments in Induction*, Academic Press, New York, 1966.

[3] R.S. Michalski, On the Quasi-Minimal solution of the general covering problem, in: *Proceedings of the Fifth International Symposium on Information Processing*, Bled, Yugoslavia, pp. 125–128.

[4] M.A. Bramer, Using J-Pruning to Reduce Overfitting of Classification Rules in Noisy Domains. *Proceedings of 13th International Conference on Database and Expert Systems Applications— DEXA 2002*, Aix-en-Provence, France, September 2–6, 2002.

[5] M.A. Bramer, *Principles of Data Mining*. London: Springer, 2007.

[6] F. Stahl and M.A. Bramer, Jmax-pruning: A facility for the information theoretic pruning of modular classification rules. *Knowledge-Based Systems 29* (2012) 12-19.

[7] J. Cendrowska, PRISM: an algorithm for inducing modular rules, *International Journal of Man-Machine Studies 27* (1987) 349–370.

[8] F. Stahl and M.A. Bramer., Induction of modular classification rules: using Jmax-pruning. In: *In Thirtieth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 14-16 December 2011, Cambridge.

[9] F. Stahl and M.A. Bramer, Computationally efficient induction of classification rules with the PMCRI and J-PMCRI frameworks. *Knowledge-Based Systems 35* (2012) 49-63.

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

[11] M.A. Bramer, Automatic induction of classification rules from examples using N-Prism, *Research and Development in Intelligent Systems*, vol. XVI, Springer-Verlag, Cambridge, 2000, pp. 99–121.

[12] M.A. Bramer, Inducer: a public domain workbench for data mining, *International Journal of Systems Science 36* (2005) 909–919.

[13] M.A. Bramer, Using J-Pruning to Reduce Overfitting in Classification Trees. In: *Research and Development in Intelligent Systems XVIII*. Springer-Verlag, 2002, pp. 25-38.

[14] M.A. Bramer, An information-theoretic approach to the pre-pruning of classification rules, in: *B.N. M Musen, R. Studer (Eds.), Intelligent Information Processing*, Kluwer, 2002, pp. 201–212.

[15] C.L. Blake, C.J. Merz, *UCI repository of machine learning databases*, Technical Report, University of California, Irvine, Department of Information and Computer Sciences, 1998.

[16] F. Neri, Cooperative evolutive concept learning: an empirical study. *WSEAS Transaction on Information Science and Applications*, WSEAS Press (Wisconsin, USA), issue 5, vol. 2, 2005, pp. 559-563.

[17] N. Dimitrova and M. Krastanov, On the Asymptotic Stabilization of an Anaerobic Digestion Model with Unknown Kinetics. In: S. *Vassileva and F. Neri (Eds.), WSEAS Transaction on Systems: the Special Issue on Modelling and Control of Integrated Bio-Systems*, issue 7, vol. 11, July 2012, pp.244-255.

[18] T. Slavov and O. Roeva, Application of Genetic Algorithm to Tuning a PID Controller for Glucose Concentration Control. In: S. *Vassileva and F. Neri (Eds.), WSEAS Transaction on Systems: the Special Issue on Modelling and Control of Integrated Bio-Systems*, issue 7, vol. 11, July 2012, pp.223-233.

[19] S.Vassileva**,** Advanced Fuzzy Modeling of Integrated Bio-Systems. In: S. *Vassileva and F. Neri (Eds.), WSEAS Transaction on Systems: the Special Issue on Modelling and Control of Integrated Bio-Systems*, issue 7, vol. 11, July 2012, pp.234-243.

[20] M. Angelova, P. Melo-Pinto and T. Pencheva, Modified Simple Genetic Algorithms Improving Convergence Time for the Purposes of Fermentation Process Parameter Identification. In: S. *Vassileva and F. Neri (Eds.), WSEAS Transaction on Systems: the Special Issue on Modelling and Control of Integrated Bio-Systems*, issue 7, vol. 11, July 2012, pp.256-267.

[21] K. Robenack and N. Dingeldey, Observer Based Current Estimation for Coupled Neurons. In: S. *Vassileva and F. Neri (Eds.), WSEAS Transaction on Systems: the Special Issue on Modelling and Control of Integrated Bio-Systems*, issue 7, vol. 11, July 2012, pp. 268-281.

[22] F. Stahl, M. Gaber, H. Liu, M. Bramer and P. Yu, Distributed classification for pocket data mining. *In: Proceedings of the 19th*

International Symposium on Methodologies for Intelligent Systems (ISMIS 2011), 28-30 June, 2011, Warsaw, Poland.

[23] F. Stahl, M. Gaber, P. Aldridge, D. May, H. Liu, M. Bramer and P. Yu, Homogeneous and Heterogeneous Distributed Classification for Pocket Data Mining. *In: A. Hameurlain, J. Küng, and R. Wagner (eds.) Transactions on large-scale data and knowledge-centered systems V. Lecture Notes in Computer Science (7100)*. Springer, pp. 183-205.