Induction of Modular Classification Rules by Information Entropy Based Rule Generation

Han Liu and Alexander Gegov

Abstract Prism has been developed as a modular classification rule generator following the separate and conquer approach since 1987 due to the replicated sub-tree problem occurring in Top-Down Induction of Decision Trees (TDIDT). A series of experiments have been done to compare the performance between Prism and TDIDT which proved that Prism may generally provide a similar level of accuracy as TDIDT but with fewer rules and fewer terms per rule. In addition, Prism is generally more tolerant to noise with consistently better accuracy than TDIDT. However, the authors have identified through some experiments that Prism may also give rule sets which tend to underfit training sets in some cases. This paper introduces a new modular classification rule generator, which follows the separate and conquer approach, in order to avoid the problems which arise with Prism. In this paper, the authors review the Prism method and its advantages compared with TDIDT as well as its disadvantages that are overcome by a new method using Information Entropy Based Rule Generation (IEBRG). The authors also set up an experimental study on the performance of the new method in classification accuracy and computational efficiency. The method is also evaluated comparatively with Prism.

1 Introduction

Automatic generation of classification rules has been an increasingly popular technique in commercial applications such as a decision making system. Classification rule generation approaches can be subdivided into two categories: 'divide and conquer approach' [1], which is also known TDIDT and induces rules in the intermediate form of a decision tree, and 'separate and conquer approach', also called 'covering approach', which induces 'If-Then' rules directly from training instances. Both approaches can be traced back to 1960s [2, 3] and achieve a comparable accuracy. A well-known example of classification rule generators following 'divide and conquer approach' is C4.5, a version of decision trees, introduced by Quinlan [1]. It is a widely used method but its representation has a serious potential drawback as mentioned in [4] in that a tree may contain many redundant terms (attribute-value pairs) as there are rules which cannot be easily represented by a tree structure if these rules have no common attributes. This problem is called replicated subtree problem and will be mentioned in section 2 in details. The existence of this problem motivated the development of Prism method [5]

Han Liu
University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, Email: Han.Liu@port.ac.uk

Alexander Gegov
University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, Email: Alexander.Gegov@port.ac.uk

with the aim to generate modular classification rules using the separate and conquer approach.

In this paper, section 2 introduces the replicated subtree problem and the principle of TDIDT and Prism in generating classification rules. It also describes the advantages of Prism compared with TDIDT and its disadvantages that are going to be avoided in the new method that the authors proposed. Section 3 explains the proposed method and mention about dealing with some common issues arising in classification tasks such as clash, conflict resolution and continuous attributes. Section 4 describes the setup of experimental study and presents the results. Section 5 evaluates the performance of this proposed method analysing its strengths and limitations. Section 6 summarises the contribution of the current work to real world applications and highlights future work aimed at overcoming the limitations identified in section 5.

2 Related Work

As stated in Section 1, a decision tree may result in a replicated subtree problem. The Prism algorithm has been developed to solve this problem. This section describes the principle of the two algorithms and compare them in accuracy and computational efficiency as well as identifies some limitations of Prism in the following subsections.

2.1 Decision Tree Algorithm

Decision trees have been a popular method as a means of generating classification rules and they are based on the fairly simple but powerful TDIDT algorithm [4]. The basic idea of this algorithm can be illustrated in Figure 1.

```
IF all cases in the training set belong to the same class
THEN return the value of the class
ELSE
  (a) Select the 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 TDIDT Tree generation algorithm [4]

One popular method of attribute selection for step (a) illustrated in figure 1 is based on average entropy of an attribute [4], which is to select the attribute that can minimize the value of entropy for the current subset being separated, thus maximize information gain. Some examples of decision trees using entropy as attribute selection technique include ID3, C4.5 and C5.0.

Entropy was introduced by Shannon in [6], which is a measure of the uncertainty contained in a training set, due to the presence of different classifications [4]. It can be calculated in the following way [4]:

- To calculate the entropy for each attribute-value pair in the way that the entropy of a training set is denoted by E and is defined by the formula: $E= -\sum^{K}_{i=1}p_i \log_2 p_i$ summed over the classes for which $p_i \neq 0$ (p denotes the probability of class i) if there are k classes.
- To calculate the weighted average for entropy of resulting subsets.

For the conditional entropy of an attribute-value pair, p_i denotes the posterior probability for class i when given the particular attribute-value pair as a condition. On the other hand, for initial entropy, the p_i denotes the priori probability for class i. The information gain is calculated by subtracting the initial entropy from the average entropy of a given attribute.

2.2 Replicated Subtree Problem

In a PhD project at the Open University, Cendrowska criticised the principle of generation of decision trees which can then be converted into a set of individual rules, compared with the principle of generation of classification rules directly from training instances [4]. She comments the following:

"The decision tree representation of rules has a number of disadvantages…[Most] importantly, there are rules that cannot easily be represented by trees."

She gave an example with the following rule set:

Rule 1: If a=1 and b=1 Then class= x
Rule 2: If c=1 and d=1 Then class= x

This kind of rule set may not fit into a tree structure. However, if they are forced to fit into a tree structure, then it is required to add other terms to at least one of the two rules, which would require at least one other rule to cover instances excluded by the addition of other terms.

For the above example, we need to have four attributes a, b, c and d, each of which has three possible values 1, 2 and 3, and is to be selected for partitioning at the root node.

The rule set would be listed as follows and it is illustrated in Figure 2:

If a=1 and b=1 then class=x
If a=1 and b= 2 and c=1 and d=1 then class=x
If a=1 and b=3 and c=1 and d=1 then class=x
If a=2 and c=1 and d=1 then class= x
If a=3 and c=1 and d=1 then class=x

It was pointed out in [9] that ID3 is difficult to manipulate for expert systems as it is required to examine the whole tree in order to extract rules about a single classification. It has been partially solved by converting a tree to a set of individual rules but there are some rules that are not easily fit into a tree structure as is the replicated subtree problem mentioned above. In a medical diagnosis system, this problem may lead to unnecessary surgery [5, 9]. The reasons identified in [9] are the following:

- The decision tree is attribute oriented
- Each iteration in the generation process chooses the attribute on which to be split aiming at minimizing the average entropy. i.e. measuring the average uncertainty. However, this doesn't necessarily mean that the uncertainty for each rule is reduced.
- An attribute might be highly relevant to one particular classification and irrelevant to the others. Sometimes only one value of an attribute is relevant.

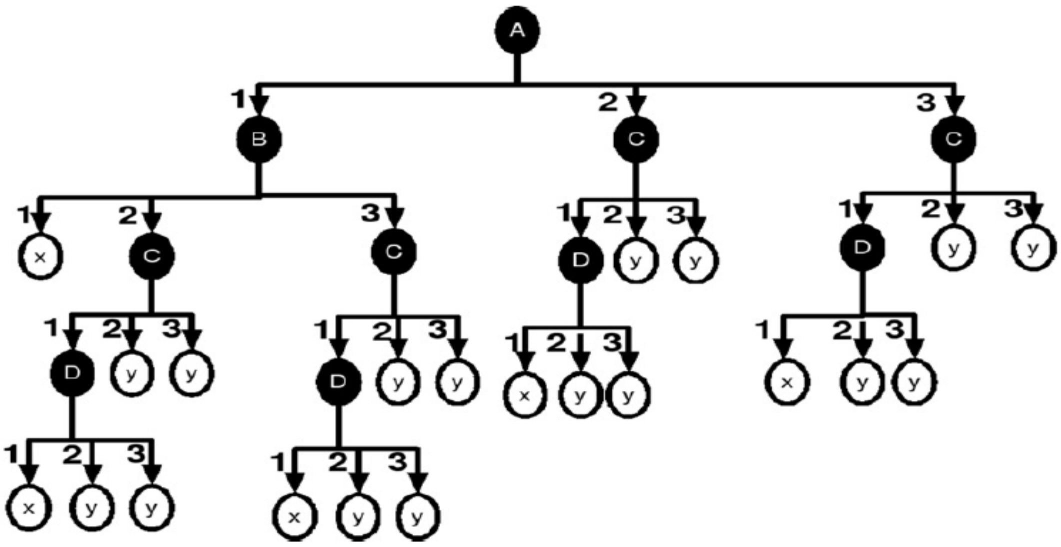

Fig. 2 Cendrowska's replicated subtree example [7, 8]

2.3 Prism Algorithm

As mentioned in section 1, the development of Prism algorithm was motivated due to the existence of the replicated subtree problem. Original Prism was introduced by Cendrowska in [5] and its basic idea can be illustrated in Figure 3.

For each classification (class= i) in turn and starting with the complete training set each time:

1. Calculate the probability class =I for each attribute-value pair.
2. Select the attribute-value pair with the largest probability and create a subset of the training set comprising all the instances with the selected attribute-value pair (for all classifications)
3. Repeat 1 and 2 for this subset 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.
4. Remove all instances covered by this rule from the training set.

Repeat 1-4 until all instances of class I have been removed

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

Fig. 3 Basic Prism Algorithm [4]

The original Prism is based on the assumption that all attributes in a training set are categorical. When there are actually continuous attributes, one way to deal with that is discretisation of attribute values prior to generating rules as described in [4, 7, 8] such as ChiMerge [10]. In addition, Bramer's Inducer software [11] provides implementation that deals with continuous attributes as described in [4, 7, 8]. On the other hand, the original version of Prism doesn't take clash problem into account. However, the Inducer software implementations of Prism algorithm provide the strategy of dealing with a clash set as the following [4]:

If a clash occurs while generating the 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 majority 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.

Fig. 4 Dealing with clashes in Prism

Another problem considered in Prism called tie-breaking is that there are two or more attribute-value pairs which have the equally highest probability. The original Prism decides to choose arbitrarily whereas Bramer improved it by choosing the one with the highest total frequency [4].

In addition, with the motivation of improving the computation efficiency, Bramer pointed out that the original Prism always deletes instances covered by rules generated so far and then resets the training set to its original state once the generation for a target class is finished. This undoubtedly increases the number of iterations resulting in high computational cost [12]. Therefore, Bramer developed two new versions of Prism called PrismTC, which chooses the majority class as the target class for each rule being generated, and PrismTCS, which chooses the minority class as the target class for each rule being generated, respectively. Both versions select a new class as the target class after a rule generated and do not reset the dataset to its original state as well as set an order in which rule is being applied to predict unseen instances. Prism TCS does not restore the training set to its original state and thus is faster than Original Prism but also provides a similar level of classification accuracy [8, 12]. However, Bramer's experiments show that PrismTC doesn't compare well against Original Prism and PrismTCS [8, 12].

2.4 Comparing Prism with Decision Tree Learning

Bramer described in [13] a series of experiments to compare the performance of Prism against that of TDIDT on a number of datasets. He concluded the following:

- Prism algorithm may give classification rules at least as good as those generated from TDIDT algorithms.
- There are generally fewer rules but also fewer terms per rule generated, which is likely to aid their comprehensibility to domain experts and users. This result

indicates that Prism generally performs consistently better than TDIDT in terms of accuracy.

- The main difference is that Prism generally prefers to leave a test instance unclassified rather than to give it an incorrect classification.
- The reasons why Prism is more noise tolerant than TDIDT may be due to the generation of fewer terms per rule in most cases.
- Prism generally has higher computational efficiency than TDIDT and it may be further improved by parallelisation.

As mentioned in section 2.2, there is a case that only one value of an attribute is relevant to a particular classification and that ID3 (a version of decision tree) is not taken into consideration. Deng pointed out in [9] that the Prism method is attribute-value-oriented and pays much attention to the relationship between an attribute-value pair and a particular classification, thus generating fewer but more general rules than a decision tree learning method.

2.5 Limitations of Prism

Prism algorithm also has some disadvantages. One of them is that the Original version of Prism may generate a rule set which may result in a classification confliction in predicting unseen instances. Let us see the example below:

Rule 1: If x=1and y=1 then class= a
Rule 2: If z=1 then class= b

Therefore, what should the classification be for an instance with x=1, y=1 and z=1? One rule gives class a, the other one gives class b. We need to give a method of choosing only one classification to classify the unseen instance [4]. This method is known as a conflict resolution strategy. Bramer mentioned in [4] that Prism uses the ‘take the first rule that fires’ strategy in dealing with the conflict problem and therefore it is required to generate the most important rules first as much as possible. However, the original Prism cannot actually introduce an order to a rule according its importance as each of those rules with a different target class is independent of the others. As mentioned above, this version of Prism would restore 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 indicates the rule generation for each class may be done in parallel so the algorithm cannot directly rank the importance among rules. Thus the ‘take the first rule that fires’ strategy may not deal with the confliction well. The PrismTCS doesn’t restore the dataset to its original state unlike original Prism and thus can introduce the order to a rule according to its importance. This problem is partially resolved but PrismTCS may potentially lead to underfitting of a rule set. PrismTCS always chooses the minority class in the current training set as the target class of the rule being generated. Since the training set is never restored to its original size as mentioned above, it can be proved that one class may always be selected as the target class until all instances of this class have been deleted from the training set because the instances of this minority class covered by the current rule generated should be removed prior to generating the next rule. This case may result in that the majority class in the training set may not be necessarily selected as the target class to generate a list of rules until the termination of the whole generation process. In this case, there is not even a single rule having the majority class as its consequence (right hand side of this rule). In some implementations, this problem has been partially solved by assigning a default class (usually the majority class) in predicting unseen instances when there is not a single rule that can cover this instance. However, this should be based on the assumption that the training set is complete. Otherwise, the rule set may still underfit the training set as the conditions of classifying instances to the other classes are probably not strong enough. On the other hand, if a clash occurs, both the original Prism and PrismTCS would prefer to discard the whole rule than to assign the majority class to the rule. As mentioned above, Prism may generally generate more general and fewer rules than decision tree learning methods. One reason is potentially due to discarding rules. In addition, the clash may happen in two main ways as follows:

1) One of the instances has at least one incorrect record for its attribute values or its classification [4].
2) The clash set has both (or all) instances correctly recorded but it is impossible to discriminate between (or among) them on the basis of the attributes recorded and thus it may be required to examine further values of attributes [4].

When there is noise present in datasets, Prism may be more tolerant than decision tree learning as mentioned above. However, if the reason why a clash occurs is not due to noise, then it may result in underfitting of the rule set by discarding rules as it will leave many unseen instances unclassified in prediction stage. Prism would decide to discard the rules in some cases is probably because it uses the so-called 'from outcome to cause' approach. As mentioned in section 2.3, each rule being generated should be pre-assigned a target class and then the conditions should be searched by adding terms (antecedents) until the adequacy conditions are met. Sometimes, it may not necessarily receive adequacy conditions even after all attributes have been examined. This indicates the current rule covers a clash set that contains instances of more than one class. If the target class is not the majority class, this indicates the search of causes is not successful so the algorithm decides to give up by discarding the incomplete rule and deleting all those instances that match the target class in order to avoid the same case to happen all over again [7, 8]. This actually not only increases the irrelevant computation cost but also results in underfitting the rule set.

3 Information Entropy Based Rule Generation Method

As mentioned in Section 2, Prism has some obvious limitations. For this reason, a new rule generation method IEBRG using the separate and conquer approach is introduced. This method avoids the underfitting of data sets and does not lead to any redundant computational effort.

3.1 *Essence of the Method*

This method is attribute-value-oriented like Prism but it uses the 'from cause to outcome' approach. In other words, it doesn't have a target class pre-assigned to the rule being generated. The main difference with respect to Prism is that IEBRG focuses mainly on minimising the uncertainty for each rule being generated no matter what the target class is. A popular technique used to measure the uncertainty is information entropy as mentioned in section 2.1. Once the antecedents added so far can reduce the entropy of the subset to 0, there is no uncertainty remaining in the subset to decide a classification assigned to this rule as the consequence. The basic idea of IEBRG with using entropy can be illustrated in Figure 5.

1. Calculate the conditional entropy of each attribute-value pair in the current subset
2. Select the attribute-value pair with the smallest entropy to spilt on, i.e. remove all other instances that contain no attribute-value pair.
3. Repeat step 1 and 2 until the current subset contains only instances of one class (the entropy of the resulting subset is zero).
4. Remove all instances covered by this rule.

Repeat 1-4 until there are no instances remaining in the training set.

* For each rule, no one attribute can be selected more than once during generation.

Fig. 5 IEBRG algorithm

3.2 Justification of the Method

As mentioned in Section 2.5, all versions of Prism need to have a target class pre-assigned to the rule being generated. As mentioned in section 2.2, an attribute might be not relevant to some particular classifications. Sometimes only one value of an attribute is relevant. Therefore, the Prism method chooses to pay main attention to the relationship between an attribute-value pair and a particular class. However, the class to which the attribute-value pair is highly relevant is probably unknown, as can be seen from the example in Table 1 below with reference to the lens 24 dataset. This dataset shows that P (class=3|tears=1) =1 illustrated by the frequency table for attribute "tears". The best rule generated first would be "if tears=1 then class=3".

Table 1 Lens 24 dataset example

Class Label	Tears=1	Tears=2
Class=1	0	4
Class=2	0	5
Class=3	12	3
total	12	12

This indicates that the attribute-value "tears=1" is only relevant to class 3. However, this is actually not known before the rule generation. According to the PrismTCS strategy, the first rule being generated would select "class =1" as the target class as it is the minority class (Frequency=4). Original Prism may select class 1 as well as it is of a smaller index. As described in [4], the first rule generated by Original Prism is "if astig=2 and tears=2 and age=1 then class=1". Therefore, the computational performance is slightly worse than expected and the resulting rule is more complex. Sometimes, the Prism method may even generate an incomplete rule reaching a clash as mentioned in 2.5 if the target class assigned is not a good fit to some of those attribute-value pairs in the current training set. Then the whole rule may be discarded resulting in underfitting and redundant computational effort.

In order to find a better strategy for reducing the computational cost, the authors proposed the method mentioned in section 3.1. In this technique, the first iteration of the rule generation process for the "lens 24" dataset can make the resulting subset's entropy reach 0. Thus the first rule generation is compete and its rule is represented by "if tears=1 then class=3".

In comparison with the Prism family, this algorithm may reduce significantly the computational cost. In addition, in contrast to Prism, the IEBRG method deals with clashes (introduced in section 3.3) by assigning the majority class in the clash set to the current rule. This may potentially reduce the underfiting of rule sets thus reducing the number of unclassified instances although it may increase the number of misclassified instances. On the other hand, the IEBRG may also have the potential to avoid clashes occurring better than Prism.

3.3 Dealing with Clashes

There are two principal ways of dealing with clashes mentioned in [4] as follows:

1) Majority voting: to assign the most common classification of the instances in the clash set to the current rule.
2) Discarding: to discard the whole rule currently being generated

In this paper, the authors choose 'majority voting' as the strategy of dealing with this problem as the objective of this paper is mainly to validate this method and find its potential in improving accuracy and computation efficiency as much as possible.

3.4 Dealing with Tie-breaking and Conflict

The tie-breaking problem is solved by deciding which attribute-value pair is to be selected to split the current subset when there are two or more attribute-value pairs that equally match the selection condition. In the IEBRG method, this problem may occur when two or more attribute-value pairs have the equally smallest entropy. The strategy is the same as the one applied to Prism by taking the one with the highest total frequency as introduced by Bramer in [4].

The classification conflict problem may occur to modular classification rule generator such as Prism. Similarly, the IEBRG may also face this problem. The authors choose the 'take the first rule that fires' strategy which is already mentioned in section 2.3 because this method may potentially generate the most important rules first. Let us see the example below:

Rule 1: if x=1 and y=1 then class= 1;
Rule 2: if x=1 then class=2;

This seems as if there is a conflict problem but the two rules can be ordered as rule 1 is more important. In other words, the two rules can be represented in the following two ways:

Rule 2: if x=1 and y≠1 then class=2;

This indicates that after the first rule has been received, the term ‘x=1’ can directly reduce the entropy of the subset to 0 as all uncertainty has been removed from the current set after removing those instances covered by the first rule. Thus the first rule is more important than the second one.

4 Experimental Setup and Results

In this experimental study, the authors use 10 datasets retrieved from the UCI repository [14] as illustrated by table 2 and 3. The authors set to compare the classification accuracy and computation efficiency performed by Prism and IEBRG. For accuracy measure, the authors choose to use cross-validation [4]. With regards to efficiency, the authors choose the whole dataset as training set to build the model and then use the same dataset to do testing. The efficiency is measured in terms of the number of rules and the number of terms (antecedents) per rule during modelling stage. For the Prism algorithm, the authors also count the number of discard rules as they actually rise the computational cost. The accuracy and efficiency are illustrated in Tables 2 and 3, respectively.

Table 2 Classification accuracy: Prism vs IEBRG

Dataset	Prism	IEBRG
Lens24	75%	75%
Vote	96%	93%
Weather	67%	71%
Contact-lenses	67%	75%
Breast-cancer	72%	75%
Lung-cancer	74%	78%
Nurse	43%	61%
Car	70%	71%
Kr-vs-kp	65%	84%
Tic-tac-toe	67%	46%

Table 3 Computational efficiency: Prism vs IEBRG

Dataset	Prism			IEBRG	
	Count(rules)	Count(terms) per rule	Count(discard rules)	Count(rules)	Count(terms) per rule
Lens24	9	2.25	1	4	1.25
Vote	19	5.74	1	7	1.71
Weather	3	2.0	1	2	1.0
Contact-lenses	4	2.75	2	4	1.4
Breast-cancer	12	1.33	0	7	1.0
Lung-cancer	3	1.0	0	4	1.0
Nurse	70	7.9	272	121	6.99
Car	5	4.0	55	23	3.96
Kr-vs-kp	77	6.74	0	9	1.0
Tic-tac-toe	164	8.06	88	91	6.65

5 Evaluation

Table 2 shows that IEBRG performs better than Prism in classification accuracy in most cases. Table 2 also shows that the classification accuracy performed by IEBRG in 'Tic-tac-toe' dataset is slightly worse than that done by Prism. The reason is not entirely clear but it may be due to the presence of noisy data as Prism is generally more tolerant to noise as mentioned in section 2.4. The authors need to take further investigation about the tolerance to noisy data for IEBRG. In addition, if the noise is actually present in a training set, this may be handled by feature selection techniques and pruning strategies.

In terms of computational efficiency, Table 3 shows that IEBRG performs better than Prism. It is obvious from the comparison of number of rules and number of terms per rule that IEBRG generates fewer but more general rules than Prism in most datasets. Although IEBRG generates more rules than Prism in three datasets namely 'Lung-cancer', 'Nurse' and 'Car', it is obvious from the comparison of the number of terms per rule and number of discarded rules that IEBRG gives more general rules than Prism in two of the three cases and Prism discards more rules in 'Nurse' and 'Car' datasets. In both datasets, Prism not only increases significantly the redundant computational effort but also losses accuracy compared with IEBRG due to discarding many rules. This also shows that IEBRG performs better than Prism in dealing with clashes by assigning the majority class to a rule rather than discarding rules if the reason why clashes occur is not due to the presence of noise. This is probably because the attributes recorded in the dataset are not sufficient and the uncertainty remaining in the set may become 0 if one extra attribute is added to be examined. In this case, majority voting would potentially be the dominant strategy through looking at the results shown by table 2 and 3. On one hand, this may avoid underfitting of rule sets. On the other hand, this can increase the probability of getting correct classifications. However, it is also dependent on application domains. For example, in domains where the decision making system operates under uncertainty, it may be required to leave instances unclassified such as safety critical systems.

6 Conclusion

This paper has reviewed TDIDT and Prism as well as identified some limitations of Prism. A new modular rule generation method, called IEBRG, has been proposed and validated. The experimental study has shown that IEBRG has the potential to avoid underfitting of rule sets and to generate fewer but more general rules as well as to perform relatively better in dealing with clashes. IEBRG can also generate the most important rules first so that it can effectively avoid the classification conflict problem by taking the 'take the first rule that fires' strategy. However, the experiments are all based on datasets that contain no continuous attributes. Therefore, the authors will make further experiments against continuous attributes which can be dealt with by some strategies such as ChiMerge [10] and another strategy as described in [4, 7, 8]. Furthermore, the authors will investigate the overcoming effects of noise by using some pruning methods such as J-pruning, which has been applied to Prism [15] and decision trees [16], and Jmax-pruning, which has been applied to Prism [7, 8]. Both methods have shown good performance in experimental studies and are based on J-measure, which was introduced into the classification rules literature by Smyth and Goodman [17] who strongly justified the use of J-measure as information theoretic means of quantifying the information content of a rule.

7 References

1. J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.
2. E.B. Hunt, P.J. Stone, J. Marin, 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, 1969.
4. Bramer, M.A. (2007). Principles of Data Mining. London: Springer.
5. J. Cendrowska, PRISM: an algorithm for inducing modular rules, International Journal of Man-Machine Studies 27 (1987) 349–370.
6. C. Shannon (1948). A mathematical theory of communication, Bell System Technical Journal, vol.27, no.3, 379-423. Fonn.
7. Stahl, F and Bramer, Max (2011) Induction of modular classification rules: using Jmax-pruning. In: In Thirtieth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge.
8. Stahl, F and Bramer, Max (2012) Jmax-pruning: A facility for the information theoretic pruning of modular classification rules. Knowledge-Based Systems 29 (2012) 12-19.
9. Deng.X. A Covering-based Algorithm for Classification: PRISM. CS831: Knowledge Discover in Databases, 2012.
10. R. Kerber, Chimerge: Discretization of numeric attributes. In: AAAI'92 Proceedings of the tenth national conference on Artificial intelligence, pp. 123-128, 1992.
11. M.A. Bramer, Inducer: a public domain workbench for data mining, International Journal of Systems Science 36 (2005) 909–919.
12 Stahl, F and Bramer, Max (2012). Computationally efficient induction of classification rules with the PMCRI and J-PMCRI frameworks. . Knowledge-Based Systems 35 (2012) 49-63.
13. 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.
14. 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.
15. Bramer, M.A. (2002). 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.
16. 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.
17. 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.