Unified Framework for Construction of Rule Based Classification Systems

Han Liu[1], Alexander Gegov[1] and Frederic Stahl[2]

Abstract Automatic generation of classification rules has been an increasingly popular technique in commercial applications such as Big Data analytics, rule based expert systems and decision making systems. However, a principal problem that arises with most methods for generation of classification rules is the overfitting of training data. When Big Data is dealt with, this may result in the generation of a large number of complex rules. This may not only increase computational cost but also lower the accuracy in predicting further unseen instances. This has led to the necessity of developing pruning methods for the simplification of rules. In addition, classification rules are used further to make predictions after the completion of their generation. As efficiency is concerned, it is expected to find the first rule that fires as soon as possible by searching through a rule set. Thus a suitable structure is required to represent the rule set effectively. In this chapter, the authors introduce a unified framework for construction of rule based classification systems consisting of three operations on Big Data: rule generation, rule simplification and rule representation. The authors also review some existing methods and techniques used for each of the three operations and highlight their limitations. They introduce some novel methods and techniques developed by them recently. These methods and techniques are also discussed in comparison to existing ones with respect to efficient processing of Big Data.

Keywords Data Mining · Machine Learning · Rule Based Systems · Rule Based Classification · Information Granularity · Big Data · Computational Intelligence

[1] Han Liu
University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, United Kingdom Email: Han.Liu@port.ac.uk
Alexander Gegov
University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, United Kingdom Email: Alexander.Gegov@port.ac.uk
[2] Frederic Stahl
University of Reading, School of Systems Engineering, Po Box 225 White Knights, RG6 6AY Reading, United Kingdom Email: F.T.Stahl@reading.ac.uk

1 Introduction

Automatic induction of classification rules has been increasingly popular in commercial applications such as Big Data analytics, rule based expert systems and predictive decision making systems. The methods of classification rule generation can be divided into two categories: ‘divide and conquer’ and ‘separate and conquer’. The former is also known as Top-Down Induction of Decision Trees (TDIDT) [1], which generates classification rules in the intermediate form of a decision tree such as ID3[1], C4.5 and C5.0. The latter is also known as covering approach [2], which generates if-then rules directly from training instances such as Prism [7]. A series of experiments have shown that Prism achieves a similar level of accuracy compared with TDIDT and can even outperform TDIDT in some cases [3].

However, a principal problem [4] that arises with most methods for generation of classification rules is the overfitting of training data. When the training data is large, this may result in the generation of a large number of complex rules. This may not only increase computational cost but also lower the accuracy in predicting further unseen instances. This has motivated the development of pruning algorithms with respect to the reduction of overfitting. Pruning methods can be subdivided into two categories- pre-pruning [8] , which truncates rules during rule generation, and post-pruning [8], which generates a whole set of rules and then remove a number of rules and rule terms, by using statistical or other tests [4]. A family of pruning algorithms are based on J-measure [12] used as information theoretic means of quantifying the theoretical information content of a rule. This is based on a working hypothesis [11] that rules with high information content (value of J-measure) are likely to have a high level of predictive accuracy. Two existing J-measure based pruning algorithms are J-pruning [4] and Jmax-pruning [5, 6], which have been successfully applied to Prism for the reduction of overfitting.

The main objective in prediction stage is to find the first rule that fires by searching through a rule set. As efficiency is concerned, a suitable structure is required to effectively represent a rule set that is generated by learning from Big Data. The existing rule representations include tree and list. Tree representation is mainly used to represent rule sets generated by ‘divide and conquer’ approach in the form of decision trees. It has root and internal nodes representing attributes and leaf nodes representing classifications as well as branches representing attribute values. On the other hand, list representation is commonly used to represent rules generated by ‘separate and conquer’ approach in the form of ‘if-then’ rules.

As the relevance of above operations, the authors have recently developed a unified framework consisting of these operations for the construction of rule based classification systems. On the other hand, it is stated in [18] that “Big Data is a popular term used to describe the exponential growth and availability of data, both structured and unstructured. And Big Data may be as important to business – and society – as the Internet has become.” This is due to the following reasons [18]:

- More data may lead to more accurate analyses.
- More accurate analyses may lead to more confident decision making.
- Better decisions can mean greater operational efficiencies, cost reductions and reduced risk.

IBM defines that Big Data is characterised by four Vs [19]:

- **Volume** - terabytes, petabytes, or more
- **Velocity** - data in motion or streaming data
- **Variety** - structured and unstructured data of all types - text, sensor data, audio, video, click streams, log files and more
- **Veracity** - the degree to which data can be trusted

Therefore, this chapter aims to introduce a framework for construction of rule based classification systems particularly on Big Data and to review some existing methods and techniques involved in each of the three operations namely generation, simplification and representation highlighting their limitations. The chapter also introduces some novel methods and techniques that are based on information theory and that may overcome the limitations of those methods reviewed with respect to effective and efficient processing of Big Data.

The rest of the chapter is organized as follows. Section 2 reviews Prism algorithm and identifies its limitations with respect to rule generation. It also discusses in what way J-pruning and Jmax-pruning help Prism and other rule based classifiers overcome overfitting with respect to rule simplification and the efficiency that list and tree representation achieve at prediction stage. Section 3 introduces three novel methods and techniques developed by the authors in their more recent research. It includes Information Entropy Based Rule Generation (IEBRG), Jmid-pruning and networked rule representation. The methods and techniques are discussed further in comparison to existing ones with respect to effective and efficient processing of Big Data in Section 4. Section 5 summarises the completed work reflecting the potential use to real world problems and highlights further directions.

2 Related Work

As mentioned in Section 1, a unified framework for the construction of rule based classification systems involves three operations: rule generation, rule simplification and rule representation. In this section, Prism is selected as a representative for the methods of classification rules generation with the reason that Prism is more noise-tolerant and achieves a higher predictive accuracy comparing to decision trees in some special cases but also can perform similar accuracy to decision trees in most cases. Furthermore, J-pruning and Jmax-pruning are reviewed because only the two existing pruning methods have been applied to Prism for rule simplification. In addition, rules that are generated by 'divide and conquer' approach are automatically represented in the form of decision trees and rules generated by 'separate and conquer' approach are directly represented by a linear list in the form of 'if-then' rules. However, they both have their limitations as criticised in [7] and identified by the authors respectively. These methods and techniques are described in the following subsections highlighting the limitations of them to show the motivation for the development of those novel methods and techniques which are further presented in Section 3.

2.1 Prism Method

The Prism method was introduced by Cendrowska in [7] and the basic procedure of the underlying Prism algorithm is illustrated in Fig. 1. This algorithm is primarily aimed at avoiding the generation of complex rules with many redundant terms [4] such as the 'replicated subtree problem' [7] that arises with decision trees as illustrated in Fig. 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 conditional 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 only covers instances of *class= i* in S'. 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 rule generation

Fig. 1 Basic Prism algorithm [8]

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 discretized by preprocessing the dataset prior to generating classification rules [5, 6, 8]. 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 different classes but cannot be separated further [6, 8]. However, the Inducer Software implementation [20] of Prism can handle clashes and the strategy of handling a clash is illustrated in Fig. 3.

Another problem that arises with Prism is tie-breaking, i.e. if there are two or more attribute-value pairs which have equal highest probability in a subset (see step 3 in Fig.1). The original Prism algorithm makes an arbitrary choice in step 4 as illustrated in Fig. 1 whereas the Inducer Software makes the choice using the highest total target class frequency [8].

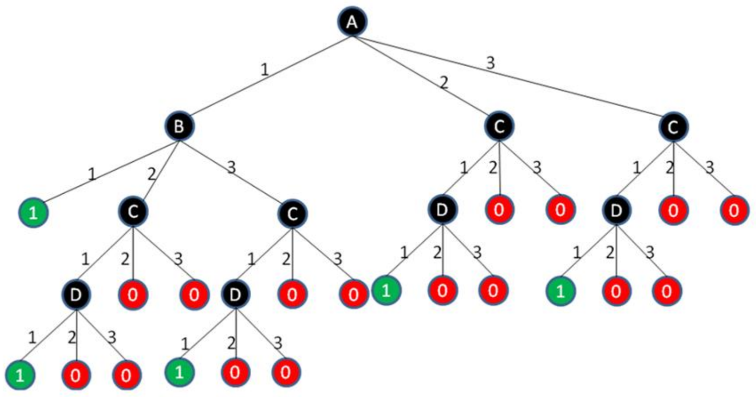

Fig. 2 Cendrowska's replicated subtree example [5, 6, 16]

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] when the training data is very large. For the purpose of increasing the computational efficiency, a modified version of Prism, called PrismTCS, was developed by Bramer [10]. 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 according to its importance [5, 6, 9]. Therefore, PrismTCS is not only faster in generating rules compared with the original Prism, but also provides a similar level of classification accuracy [5, 6, 10].

```
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 target 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

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. This can be illustrated by the example below:

Rule 1: If x=1and y=1 then class= a

Rule 2: If z=1 then class= b

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 a method to choose only one classification to classify the unseen instance [8]. Such a method is known as a conflict resolution strategy. Bramer mentioned in [8] 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. However, the original Prism cannot actually introduce an order to a rule according to its importance as each of those rules with a different target class is independent from each other. 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 classification confliction well. The PrismTCS does not restore dataset to its original state unlike original Prism and thus can introduce the order to a rule for 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 proven that one class could always be selected as 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 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 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 on 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 rather than to assign the majority class, which is higher in importance, to the rule. As mentioned above, Prism may generally generate more general and less rules than a decision tree. One reason is potentially due to discarding rules. In addition, the clash may happen in two principal 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 [8].

When there is noise present in datasets, Prism may be more robust than decision trees as mentioned above. However, if the reason that a clash occurs is not due to noise and the training set covers a large amount of data, then it may result in serious underfitting of the rule set by discarding rules as it will leave many unseen instances unclassified at prediction stage. The fact that Prism would decide to discard the rules in some cases is probably because it uses the so-called 'from effect to cause' approach. As mentioned above, 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 [5, 6]. This actually not only increases the irrelevant computation cost but also results in underfitting of the rule set.

These limitations have motivated the development of a new method for the generation of classification rules which is further introduced in Section 3.1.

2.2 J-pruning and Jmax-pruning

As mentioned in Section 1, both J-pruning and Jmax-pruning are based on J-measure which was introduced by Smyth and Goodman [12] who justified the use of the J-measure as an information theoretic means of quantifying the theoretical information content of a rule.

According to the notation of [12], given a rule of the form *IF Y = y THEN X = x* can be measured in bits and is denoted by *J(X, Y=y)*.

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

$J(X;\ Y = y)$ 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:

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

The value of *cross-entropy* depends upon two values [8]:

- $p(x)$: 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, 8] 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) · (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.

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

Thus, there are no theoretical 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.

When a rule is being generated, the J-value (value of J-measure) may increase or decrease after specialising the rule by adding a new term. Both pruning algorithms (J-pruning and Jmax-pruning) expect to find the global maximum of J-value for the rule. Each rule has a complexity degree which is the number of terms. The increase of complexity degree may lead the J-value of this rule to increase or decrease. The aim of pruning algorithms is to find the complexity degree corresponding to the global maximum of J-value as illustrated in Fig. 4 using a fictitious example.

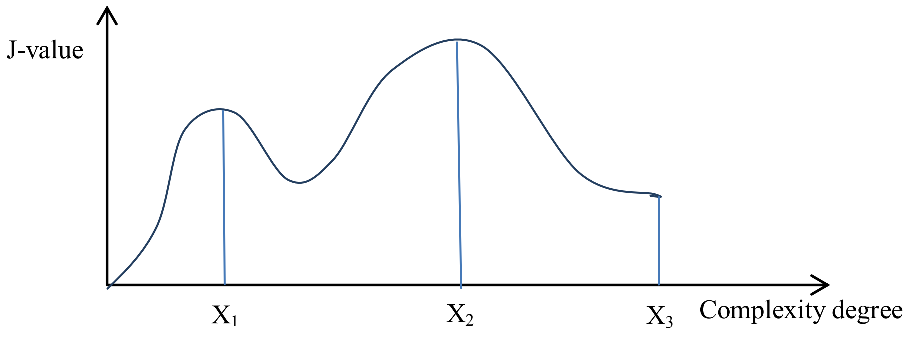

Fig. 4 Relationship between complexity degree and J-value (case 1)

However, the two pruning algorithms mentioned above search the global maximum of J-value with different strategies:

- J-pruning: monitor the change pattern of J-value and stop rule generation once it goes down. i.e it will stop rule generation when complexity degree is X_1 as illustrated in Fig.4 because the J-value is going to decrease afterwards. The final rule generated is with the complexity degree X_1 (having the first X_1 rule terms).
- Jmax-pruning: monitor and record the highest J-value observed so far until the completion of rule's generation. i.e it will stop rule generation when the complexity is X_3 as illustrated in Fig.4 and reduce the complexity degree subsequently until the degree is X_2 by removing those rule terms afterwards. The final rule is with the complexity degree X_2.

J-pruning 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 Algorithm 1.

```
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;
    }
}
```

Algorithm 1 J-pruning for Prism algorithms

J-pruning achieves relatively good results as indicated in [4]. However, Stahl and Bramer pointed out in [5, 6] that J-pruning does not exploit the J-measure to its full potential. This is because this method immediately stops the generation process as soon as the J-measure goes down after a new term is added to the rule as illustrated in Fig.4. In fact, it is theoretically 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. A possible case is that there is only one local maximum of J-value as illustrated in Fig.5. 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 being generated. This motivated the development of a new pruning method, called Jmax-pruning, which was proposed by one of the authors of this chapter [5, 6], in order to exploit the J-measure to its full potential.

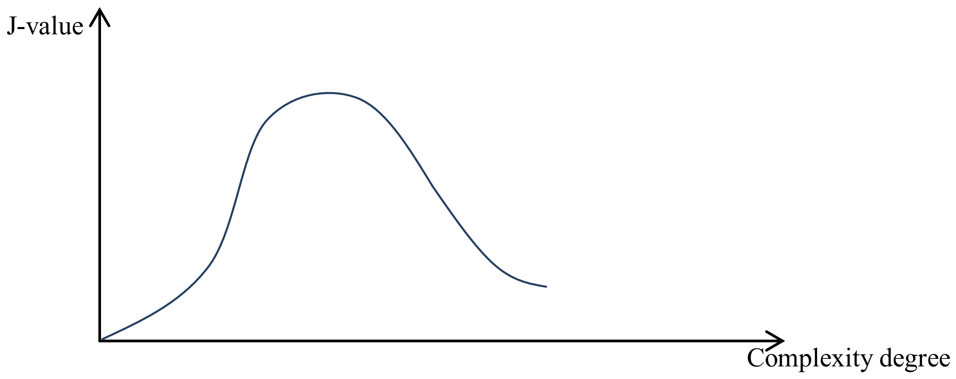

Fig. 5 Relationship between complexity degree and J-value (case 2)

Jmax-pruning can be seen as a hybrid between pre-pruning and post-pruning. 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 a 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 Algorithm 2.

```
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;
    }
}
```

Algorithm 2 Jmax-pruning for Prism algorithms

A series of experiments have shown that Jmax-pruning outperforms J-pruning in some cases [5, 6] when there are more than one local maximum and the first one is not the global maximum as illustrated in Fig.4. However, it performs the same as J-pruning in other cases [5, 6] when there is only one local maximum as illustrated in Fig.5 or the first one of local maxima is also the global maximum .

However, 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 when Big Data is used for training. This could be achieved by making use of the Jmax value as introduced above.

On the other hand, a special case may need to be taken into account when Prism is used as the classifier. This case is referred to as tie-breaking which is if there is more than one global maximum for the J-value during rule generation as illustrated in Fig.6.

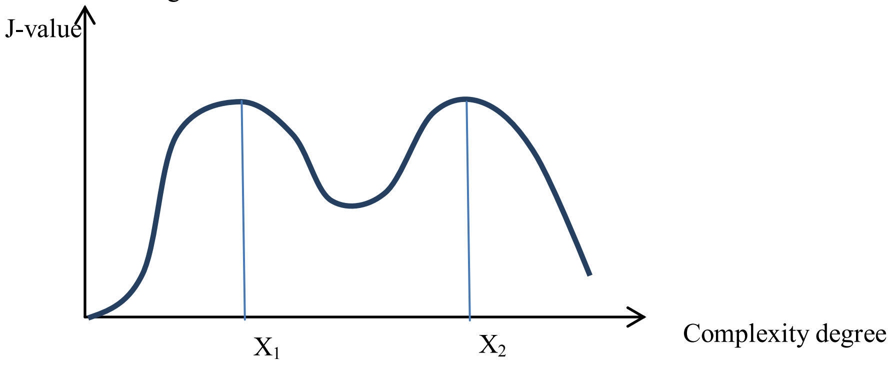

Fig. 6 Relationship between complexity degree and J-value (case 3)

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 account, it is worth to determine properly which one of the global maximum points to be chosen as the start point of pruning in order to avoid over-discarding rules. In other words, according to Fig.6, it needs to determine to choose either X_1 or X_2 as the start point for removing all rule terms afterward.

With regards to this issue, Jmax-pruning always chooses to take X_1 (the first global maximum point) as the start point of pruning and to remove all rule terms generated afterwards. It may potentially lead to underfitting as it is possible that the rule is being discarded after handling a clash if X_1 is chosen but is being kept otherwise. In addition, another type of tie-breaking may arise with the case as illustrated below:

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.

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)

However, the two cases about tie-breaking mentioned above are not very likely to happen. As the basis of above descriptions about limitations of J-pruning and Jmax-pruning, it has motivated the development of a new pruning algorithm to overcome the limitations of J-pruning and Jmax-pruning with respects to underfitting and computational efficiency. The new pruning algorithm is further introduced in Section 3.2.

2.3 Decision Tree and Linear List Representation

As mentioned in Section 1, decision tree is an automatic representation for classification rules generated by 'divide and conquer' approach. However, the representation has been criticized by Cendrowska and identified as a major cause of overfitting in [7] as illustrated in Fig.2. It was also pointed in [13] that it is required to examine the whole tree in order to extract rules about a single classification in the worst case. This drawback on representation has made it difficult to manipulate for expert systems. It has thus motivated the direct use of 'if then' rules represented by a linear list structure. However, simulation in this representation is run in linear search with the time complexity O (n) while the total number of rule terms is used as the input size (n). This is because list representation works in linear search by going through rule by rule in an outer loop; and by going through term by term for each rule in an inner loop. It implies it may have to go through the whole rule set to find the first rule that fires in the worst case. This may lead to huge computational costs when the representation is used to represent a rule set generated by learning from Big Data.

As the basis of above description about limitations of tree and list representation, it has motivated the development of a new representation of classification rules which performs a level of efficiency higher than linear time in time complexity. This new representation is further described in Section 3.3.

3 Novel Methods and Techniques

Section 2 has reviewed a representative rule generation method called Prism, two J-measure based pruning algorithms namely J-pruning and Jmax-pruning and two types of representation of classification rules namely tree and list. It has also highlighted their limitations so this section explores a novel rule generation method called Information Entropy Based Rule Generation (IEBRG); a novel J-measure based pruning algorithm called Jmid-pruning and a novel representation of classification rules called Rule Based Classification Networks.

3.1 Information Entropy Based Rule Generation

Information Entropy Based Rule Generation is a method of classification rules generation following 'separate and conquer' approach and has been recently developed in [14]. This method tends to avoid underfitting and redundant computational efforts.

3.1.1 Essence

This method is attribute-value-oriented like Prism but it uses the 'from cause to effect' approach. In other words, it does not 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 introduced by Shannon in [15]. The basic idea of IEBRG is illustrated in Fig.7 as below:

```
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 do not comprise the 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. 7 IEBRG algorithm

3.1.2 Justification

As mentioned in Section 2.1, all versions of Prism need to have a target class pre-assigned to the rule being generated. In addition, an attribute might be not relevant to each particular classification and sometimes only one value of an attribute is relevant [13]. Therefore, the Prism method chooses to pay more attention to the relationship between 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 reconstructed by Bramer in [8]. 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 PrismTCS strategy, the first rule being generated would select "class =1" as target class as it is the minority class (Frequency=4). Original Prism may select class 1 as well because it is in a smaller index. As described in [8], the first rule generated by Original Prism is "if astig=2 and tears=2 and age=1 then class=1". It indicates the computational efficiency is slightly worse than expected and the resulting rule is more complex. When Big Data is used for training, the Prism method may be even likely to generate an incomplete rule covering a clash set as mentioned in 2.1 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 in [14]. 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 complete and its rule is represented by “if tears=1 then class=3”.

In comparison to the Prism family, this algorithm may reduce significantly the computational cost when Big Data is being dealt with. In addition, in contrast to Prism, the IEBRG method deals with clashes (introduced in Section 3.1.3) by assigning a majority class in the clash set to the current rule. This may potentially reduce the underfiting of rule set 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 occurring clashes better compared with Prism.

3.1.3 Dealing with Clashes

There are two principal ways of dealing with clashes mentioned in [8] 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 [14], the authors choose ‘majority voting’ as the strategy of dealing with this problem as the objective of [14] is mainly to validate this method and to find its potential in improving accuracy and computation efficiency as much as possible.

3.1.4 Dealing with Tie-breaking on Conditional Entropy and Conflict

The tie-breaking problem on conditional entropy 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 well match the selection condition. In the IEBRG method, this problem may occur when two or more attribute-value pairs have the same smallest entropy value. The strategy is the same as the one applied to Prism by taking the one with the highest total frequency as introduced by Bramer [8].

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. Consider 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 second rule can be represented in the following way:

Rule 2: if x=1 and y≠1 then class=2;

This may indicate that after the first rule has been generated, all instances covered by the rule have been deleted from training set; then the two conditions ‘x=1’ and ‘y=1’ cannot be met simultaneously any more. Thus the first rule is more important than the second one.

3.2 Jmid-pruning

The authors have recently mentioned in [16] that 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 developed a novel pruning algorithm that avoids underfitting and unnecessary rule term inductions while at the same time rules are being pruned for reducing overfitting [16].

3.2.1 Essence

The Jmid-pruning is a modified version of the J-measure based pruning algorithm Jmax-pruning. It not only monitors and records the highest J-value observed so far but also measures the potentially highest J-value that may be achieved eventually by making use of the Jmax value highlighted in Section 2.2 in comparison to Jmax-pruning. The basic concept of this algorithm is illustrated in Algorithm 3.

```
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;
    }
            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{
         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;
    }
}
```

Algorithm 3 Jmid-pruning for Prism algorithms

3.2.2 Justification

The Jmid-pruning aims to avoid underfitting and unnecessary computational effort especially when Big Data is used for training. In fact, J-pruning and Jmax-pruning do not actually make use of Jmax value to measure the potential search space of gaining benefits.

Let us consider an example [11] using the lense24 dataset. 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 when the training data is large as mentioned in Section 2.2. For example, one of the rules [5, 6] generated from the Soybean data-set [17] 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 Jmid-pruning may get more rules not being discarded later when tie-breaking on J-value happens as mentioned in Section 2.2. In this way, Jmid-pruning is better in avoiding underfitting of rule sets.

3.3 Rule Based Classification Networks

As mentioned in Section 2.3, both tree and list representations have their individual limitations. The authors have recently developed a networked representation of classification rules called rule based classification networks.

3.3.1 Essence

Let us see a set of rules based on Boolean logic below:

If x1=0 and x2=0 then class=0;

If x1=0 and x2=1 then class=0;

If x1=1 and x2=0 then class=0;

If x1=1 and x2=1 then class=1;

The corresponding networked representation is illustrated in Fig.8. In this representation, x1=1 and x2=1 are supposed to be the two inputs respectively for simulation (prediction). Thus both 'x1' and 'x2' layers get green node labelled 1 and red node labelled 0 because each node in the layer x1 represents a value of attribute x1 and so does each node in layer x2. In addition, the two digits labelled to each of the connections between the nodes in layer x1 and x2 represent the index of rule and rule term respectively. In other words, the two digits '11' as illustrated below indicates it is for the first rule and the first term of the rule. It can be seen from the list of rules above that the first term of the first rule is 'x1=0'. However, the input value of x1 is 1 so the connection is coloured red as this condition is not met. In contrast, the connections labelled '31' and '41' respectively are both coloured green as the condition 'x1=1' is met. The same principle is also applied to the connections between the nodes in layer 'x2' and 'Rule Index'. As the two inputs are 'x1=1 and 'x2=1', the connections '31', '41' and '42' are coloured green and the node labelled 3 is green in the layer 'Rule Index' as well as the output is 1 in the layer 'Class'.

3.3.2 Justification

For Rule Based Classification Networks, simulation process is run by going through rule terms in divide and conquer search (i.e. only going through those terms that fire). The total number of terms is used as the input size of data (n) as same as used in linear list representation and thus the efficiency is O (log (n)). As can be seen from Fig.8, it only takes three steps (going through connections '31', '41' and '42') to find the first rule that fires (the rule index is 3). This is because the input value of x1 is 1 and thus the connections '11' and '21' can be ignored. In the second layer, it is only concerned with connection '42' as the input value of x2 is 1 and thus 'the connections '12' and'32' can be ignored. In addition, the connection '22' is ignored as well because the connection '21' is already discarded and thus it is not worth to go through the connection '22' any more. As the basis of above descriptions, it indicates that it is not necessary to examine the whole network in order to find the rules that fire. In practice, it may significantly speed up the process of simulation when the corresponding rule set is generated by learning from Big Data.

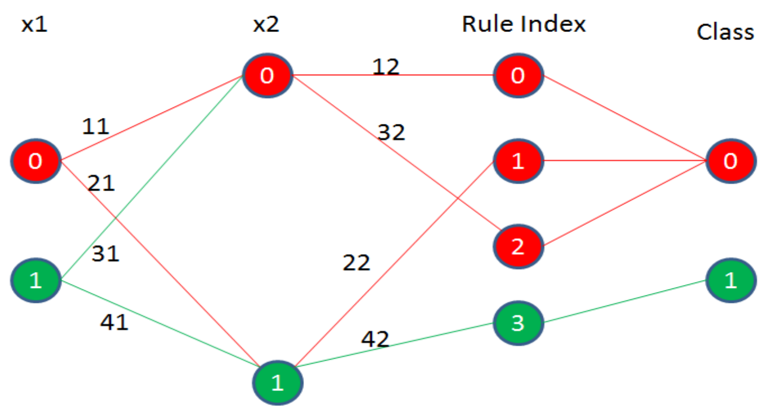

Fig. 8 Rule Based Classification Networks

4 Comparative Validation and Discussion

The authors have recently validated experimentally IEBRG against Prism [14] and Jmid-pruning against J-pruning and Jmax-pruning [16] in terms of classification accuracy and computational efficiency. They have also theoretically validated Rule Based Classification Networks against decision tree and linear list representations in terms of time complexity. With regards to classification accuracy, the authors use cross validation and check the overall accuracy, i.e. the proportion of correct classifications. With regards to computational efficiency in training stage, the authors check the number of rules and the average number of rule terms in order to reflect approximately the total number of iterations conducted during training stage. If a method generates more general and fewer rules, it indicates that the method needs less number of iterations and thus is more efficient in theory. In addition, the authors also check the time complexity using BigO notation to measure the computational efficiency in testing stage. If the complexity is lower, it indicates that the representation may make the predication on unseen instances perform more efficiently. For example, linear time is worse than logarithmic time in computational efficiency.

With regards to IEBRG, the authors conducted experiments on 10 datasets available from UCI repository [17]; they are Vote, Weather, Contact-lenses, Lense24, Breast-cancer, Nurse, Car, Lung-cancer, Kr-vs-kp and Iris. The experimental results show that that IEBRG algorithm outperforms Prism in both accuracy and efficiency in most cases. In the classification accuracy, IEBRG performs a bit worse than Prism in one case (on Vote dataset) only. However, it even slightly outperforms Prism in three cases (on Nurse, Iris and Kr-vs-kp). In the computational efficiency, IEBRG generates more general and fewer rules in most cases. In three cases (on Lung-cancer, Nurse and Car datasets), IEBRG generates more rules than Prism. However, Prism discarded large number of rules in two of these cases (on Nurse and Car datasets). Therefore, it still shows Prism is computationally more expensive than IEBRG as discarded rules also need to conduct computation for their generation although they are eventually discarded.

With regards to Jmid-pruning, the authors conducted experiments on 10 UCI datasets namely, Vote, Weather, Contact-lenses, Lense24, Breast-cancer, Car, Lung-cancer, Iris, Segment and ionosphere. The experimental results show Jmid-pruning leads PrismTCS to perform a similar level of classification accuracy in comparison with J-pruning and Jmax-pruning in most cases but outperforms the two algorithms in some cases. With regards to efficiency, 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, 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 seems likely to be computationally more efficient when training data is very large.

With regards to Rule Based Classification Networks, the authors validated the representation theoretically using BigO notation. As mentioned above, the network representation could achieve that simulation process is run in divide and conquer search and the efficiency is O (log (n)). In contrast, list representation could only achieve a linear search process for the same purpose and the efficiency is O (n). For the purpose of predictive modelling, the network representation may contribute as many quicker decisions as possible in prediction stage in expert systems. The difference to listed rule representation in the efficiency can be significant when Big Data is used to generate a rule set.

As mentioned above, the authors' recent research is mainly concerned with accuracy and efficiency. The veracity is a measure of reliability leading to more accurate analyses and confident decision making as mentioned in Section 1. However, the accuracy can indicate the uncertainty existed in a model built based on a dataset. In addition, a data set may contain missing values or noise (incorrect records). Different strategies in dealing with the issues may lead to different predictive accuracy. In classification area, each algorithm may perform a particular level of tolerance to the presence of missing values or noise. As the basis of above descriptions, veracity is subject to data based modelling techniques in the authors' research. The higher level of predictive accuracy is more likely to introduce the higher degree to which the data can be trusted. In detail, rule generation method can provide a level of predictive accuracy and pruning algorithms may help improve the accuracy.

On the other hand, volume is a measure of data scalability leading to a particular level of computational efficiency. The data scalability could be reflected by its dimensionality, average number of attribute values and the number of instances. In the authors' research, pruning algorithms may speed up the process of modelling.

The proper selection of model representations may speed up the process of simulation. Besides, the dimensionality issue can be resolved by using feature selection techniques such as entropy [15] and information gain [8] which are both based on information theory pre-measuring uncertainty present on data. In other words, it aims to remove those irrelevant attributes. When a dataset contains a large number of instances, it is possibly required to take advantage of sampling methods to choose those most representative instances. However, the authors have not yet taken feature selection and sampling into use in their current research but will do so further when large scale data is used.

5 Conclusions

This chapter has summarised the authors' more recent research in the area of rule based classification including generation, simplification and representation of classification rules. The authors have also introduced a unified framework for the construction of rule based classification systems by merging the three operations mentioned above systematically. The potential contribution to effective and efficient processing of Big Data has been discussed in the terms of volume and veracity. However, those validations are made theoretically or experimentally on some relatively small data in classification accuracy and computational efficiency. Therefore, the authors will further extend the validations onto large scale datasets and evaluate the novel methods more empirically in the concern of Big Data. They will also incorporate ensemble learning concepts and feature selection techniques with respects to the improvement of accuracy and efficiency in order to overcome the limitations that arise when Big Data is present and to make the approach more computationally intelligent.

References

1. Quinlan, J.R.: C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.
2. Michalski, R.S.: 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.
3. Bramer, M.A.: 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.
4. Bramer, M.A.: 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. Stahl, F., Bramer, M.A.: Jmax-pruning: A facility for the information theoretic pruning of modular classification rules. Knowledge-Based Systems 29 (2012) 12-19.
6. Stahl, F., Bramer, M.A.: 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.
7. Cendrowska, J.: PRISM: an algorithm for inducing modular rules, International Journal of Man-Machine Studies 27 (1987) 349–370.
8. Bramer, M.A.: Principles of Data Mining. London: Springer, 2007.
9. Stahl, F., Bramer, M.A.: Computationally efficient induction of classification rules with the PMCRI and J-PMCRI frameworks. Knowledge-Based Systems 35 (2012) 49-63.
10. Bramer, M.A.: 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.
11. Bramer, M.A.: Using J-Pruning to reduce overfitting in classification trees. In: Research and Development in Intelligent Systems XVIII. Springer-Verlag, 2002, pp. 25-38.
12. Smyth, P., Goodman, R.M.: Rule induction using information theory. In: *G.* Piatetsky-Shapiro and W.J. Frawley (eds.), Knowledge Discovery in Databases. AAAI Press, 1991, pp. 159-176.
13. Deng, X.: A Covering-based Algorithm for Classification: PRISM. CS831: Knowledge Discover in Databases, 2012.
14. Liu, H., Gegov, A.: Induction of modular classification rules by Information Entropy Based Rule Generation. V. Sgurev, R. Yager, J. Kacprzyk (Eds.), 'Innovative Issues in Intelligent Systems', Springer. (in print)
15. Shannon, C.: A mathematical theory of communication, Bell System Technical Journal, vol.27, no.3, (1948) 379-423. Fonn.
16. Liu, H., Gegov, A., Stahl, F.: J-measure based hybrid pruning for complexity reduction in classification rules. WSEAS Transaction on Systems: Issue 9, Volume 12, September 2013, pp.433-446.
17. Bache, K., Lichman, M.: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.
18. What is Big Data?. http://www.sas.com/big-data/. 7 December 2013.
19. Master Data Management for Big Data. http://www-01.ibm.com/software/data/infosphere/mdm-big-data/. 7 December 2013.
20. Bramer, M.A.: Inducer: a public domain workbench for data mining. International Journal of Systems Science, 36(14):909–919, 2005.