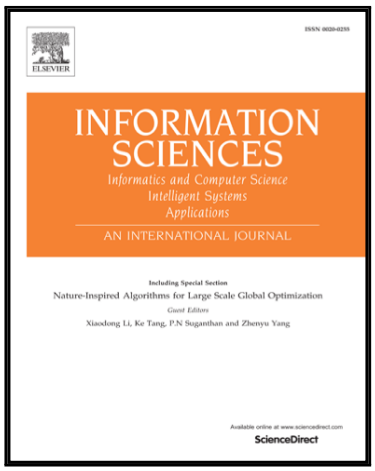

Heuristic target class selection for advancing performance of coverage-based rule learning

Han Liu, Shyi-Ming Chen, Mihaela Cocea

PII: S0020-0255(18)30945-9
DOI: https://doi.org/10.1016/j.ins.2018.12.001
Reference: INS 14101

To appear in: *Information Sciences*

Received date: 25 September 2018
Revised date: 29 November 2018
Accepted date: 1 December 2018

Please cite this article as: Han Liu, Shyi-Ming Chen, Mihaela Cocea, Heuristic target class selection for advancing performance of coverage-based rule learning, *Information Sciences* (2018), doi: https://doi.org/10.1016/j.ins.2018.12.001

Heuristic target class selection for advancing performance of coverage-based rule learning

Han Liu[a], Shyi-Ming Chen[b,*], Mihaela Cocea[c]

[a]*School of Computer Science and Informatics, Cardiff University, Queen's Buildings, 5 The Parade, Cardiff, CF24 3AA, United Kingdom*
[b]*Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan*
[c]*School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth, PO1 3HE, United Kingdom*

Abstract

Rule learning is a popular branch of machine learning, which can provide accurate and interpretable classification results. In general, two main strategies of rule learning are referred to as 'divide and conquer' and 'separate and conquer'. Decision tree generation that follows the former strategy has a serious drawback, which is known as the replicated sub-tree problem, resulting from the constraint that all branches of a decision tree must have one or more common attributes. The above problem is likely to result in high computational complexity and the risk of overfitting, which leads to the necessity to develop rule learning algorithms (e.g., Prism) that follow the separate and conquer strategy. The replicated sub-tree problem can be effectively solved using the Prism algorithm, but the trained models are still complex due to the need of training an independent rule set for each selected target class. In order to reduce the risk of overfitting and the model complexity, we propose in this paper a variant of the Prism algorithm referred to as PrismCTC. The experimental results show that the PrismCTC algorithm leads to advances in classification performance and reduction of model complexity, in comparison with the C4.5 and Prism algorithms.

*Corresponding author
Email addresses: liuh48@cardiff.ac.uk (Han Liu), smchen@mail.ntust.edu.tw (Shyi-Ming Chen), mihaela.cocea@port.ac.uk (Mihaela Cocea)

Keywords: Machine learning, Rule based systems, Rule based classification, Decision tree learning, Rule learning, Prism

1. Introduction

Rule learning is a popular form of machine learning approaches, which essentially aims at production of rule based systems [32]. In general, rule learning is undertaken through two well-known strategies, namely, 'divide and conquer' (DAC), and 'separate and conquer' (SAC). The DAC strategy is also known as Top-Down Induction of Decision Trees (TDIDT), since this strategy aims at generation of a decision tree that can be directly converted into a set of if-then rules. For example, ID3 [41] and C4.5 [43] are commonly known algorithms of TDIDT with high popularity in real-world applications. On the other hand, the SAC strategy is also referred to as the covering approach, since this strategy aims at learning a rule that covers some training instances that should then be deleted before the learning of the next rule is initiated. The representative of the covering approach is the Prism algorithm [5]. Due to the fact that the DAC strategy produces rules that are automatically represented in the form of a decision tree and rules in the form of 'if-then' can be directly generated from training instances through the SAC strategy [26], thus the former strategy is referred to as 'decision tree learning' and the latter strategy is referred to as 'rule learning' (in a narrow sense) in the rest of this paper.

The nature of decision tree learning leads to parallel growth of different branches that can be converted into several rules, i.e., the DAC strategy enables different rules to be learned in parallel. Since decision tree learning starts from attribute selection for the root node, all rules must have this selected attribute as the common attribute. The constraint on the common attribute is likely to result in a decision tree containing redundant parts (i.e., the replicated sub-tree problem) [5]. In this case, after the decision tree is transformed into a set of rules, some of these rules would have redundant terms. In order to achieve the elimination of replicated sub-trees, it has become necessary to develop rule

learning algorithms that follow the SAC strategy [18]. Although the use of the Prism algorithm can effectively lead to the elimination of the model redundancy (replicated sub-trees), the model trained by using this algorithm still shows high complexity, i.e., the model consists of a large number of complex rules, because of the nature of the Prism algorithm that training an independent set of rules is required for each selected target class.

In this paper, we propose a new version of the Prism algorithm, which is referred to as PrismCTC. The acronym 'CTC' stands for competitive target class. The key feature of the proposed PrismCTC algorithm is the involvement of a trained strategy of the selection of the target class c_i for learning each rule r_i (that is added into the final rule set as the i^{th} rule) through the competition among all classes $c[t]$, whereas the Prism algorithm simply involves a fixed strategy of the target class selection, i.e., each of the k classes in a data set is selected as the target class for learning independently one of the k required sets of rules. This paper has the following contributions:

- We propose a variant of the Prism algorithm, i.e., PrismCTC, which employs statistical measures as heuristics for target class selection in a trained strategy.
- We compare the PrismCTC algorithm with the Prism and C4.5 algorithms in terms of classification accuracy and model complexity. The experimental results show that the PrismCTC algorithm leads to considerable advances in classification accuracy compared with the C4.5 and Prism algorithms.
- We analyze in depth the difference between Prism and PrismCTC in terms of the impacts of their learning strategies on the model complexity. The experimental results prove that the PrismCTC algorithm involving a trained strategy of the target class selection leads to reduction of the model complexity, i.e., the PrismCTC algorithm produces a smaller number of simpler rules, in comparison with both C4.5 and Prism.

The rest of this paper is organized as follows. In Section 2, we provide the theoretical preliminaries on the essence of decision tree learning and rule learning. Several popular measures of rule quality are also introduced in this section. In Section 3, we provide a review of related work on heuristic learning of decision trees and if-then rules, and analyze in depth the advantages and disadvantages of some popular algorithms. In Section 4, we give detailed description of the procedure of the proposed PrismCTC algorithm, which is based on the heuristic functions of several rule quality measures. In Section 5, we conduct experiments to validate the proposed PrismCTC algorithm and show experimental results for evaluation of the algorithm performance. In Section 6, this paper is concluded by giving a summary of the contributions and providing some suggestions on further directions.

2. Preliminaries

In this section, we introduce concepts of rule based systems and illustrate the essence of decision tree learning and rule learning. Several measures of rule quality [32] are explained in depth.

2.1. Rule based systems

A rule based system is essentially a set of rules. In general, rule based systems can be used in the machine learning context for various practical tasks, such as classification, regression and association. Therefore, the type of rules can be specialized into classification rules, regression rules and association rules, depending on the type of machine learning tasks. The following rule based system, which is made up of four classification rules, is provided below for illustrative purpose:

- Rule 1: $x_1 = 0 \wedge x_2 = 0 \rightarrow class = 0$;
- Rule 2: $x_1 = 0 \wedge x_2 = 1 \rightarrow class = 0$;
- Rule 3: $x_1 = 1 \wedge x_2 = 0 \rightarrow class = 0$;

- Rule 4: $x_1 = 1 \wedge x_2 = 1 \rightarrow class = 1$;

Each rule consists of two main parts: rule antecedent and rule consequent. According to the above example, the left hand side (before the arrow symbol) of each rule is the rule antecedent, e.g., '$x_1 = 0 \wedge x_2 = 0$', whereas the right hand side (after the arrow symbol) of each rule is the rule consequent, e.g., '$class = 0$'. On the other hand, each rule antecedent is expressed as a single rule term or a conjunction of multiple rule terms, e.g., '$x_1 = 0$' and '$x_2 = 0$' are the two rule terms that make up the antecedent of Rule 1 in the form of '$x_1 = 0 \wedge x_2 = 0$'. In other words, each of the rule terms shown in the rule antecedent indicates a condition for this rule to fire, and the rule consequent indicates the classification outcome when all the conditions are met. For example, Rule 1 implies that an unseen instance would be classified to the class '0' if the instance meets the two conditions '$x_1 = 0$' and '$x_2 = 0$'. In machine learning, each learned rule would cover some training instances, which generally means that each of the instances meets all the conditions shown in the antecedent of the rule.

Moreover, since rule based operations can be involved in different kinds of machine learning tasks, such as classification, regression and association, there are different constraints for different kinds of rules in terms of the rule antecedent and consequent. In particular, for a classification rule, the antecedent can involve a single rule term or a conjunction of multiple rule terms, but the consequent of this rule can only involve one rule term that reflects a discrete output, i.e., the output is the class label that is assigned to an unseen instance which meets all the conditions shown in the rule antecedent. The same constraint also applies to regression rules with the only difference that the output indicated in the rule consequent must be continuous (numerical). Association rules can be viewed as the generalization of classification and regression rules, since both the antecedent and the consequent of an association rule can involve a single rule term or a conjunction of multiple rule terms (e.g., $x_1 = 1 \wedge x_2 = 1 \rightarrow y_1 = 1 \wedge y_2 = 0$), i.e., an association rule has no specific constraint in terms of its antecedent and consequent.

2.2. Procedure of decision tree learning

Decision tree learning essentially involves a recursive process of attribute selection. In particular, the training of a decision tree starts from selecting an attribute to label the root node, leading to n branches, where n is the number of values of the selected attribute. Each of the n branches may end at an internal node towards growing the tree by recursively training a sub-tree in the same way as the attribute selection for the root node. Also, each branch may end at a leaf node that is provided with a class label. The leaf node indicates that the growth of this branch should be terminated, due to the case that the last growth of this branch has made it achieved that all the training instances covered by this branch belong to the same class.

In fact, the attribute selection for the root node would lead to the division of the training set into its n subsets, and each subset contains instances that meet the condition expressed as an attribute-value pair. For example, if attribute a, which has two values 1 and 2, is selected for the root node, then the root node would lead to two branches and the training set would be partitioned into two subsets s_1 and s_2. In this context, s_1 contains all instances that meet the condition '$a = 1$' and all instances in s_2 meet the condition '$a = 2$'. Similarly, for each internal node, the attribute selection would also lead to partitioning of the corresponding training subset into its several sub-subsets.

When a branch appears to have a leaf node, it typically indicates that the stopping criteria of growing this branch has been normally satisfied, and this branch can already be converted into a consistent rule that covers training instances all belonging to the same class. In real-world applications, it could happen that a tree branch gets a leaf node without meeting the above criteria, due to the case that the maximum length of the branch has been reached, i.e., the length of each branch cannot be higher than the data dimensionality (the number of attributes given in a data set). When the maximum length of a tree branch has been reached, it usually means that each of the attributes has been selected for partitioning a training subset and labelling the corresponding internal node in this branch, but unfortunately it is still not achievable to make

Input : A set of training instances T

Output: A decision tree DT

```
1  if the stopping criterion is satisfied then
2  |  create a leaf node that covers all the remaining training instances
3  end
4  else
5  |  select the best attribute A_i according to some heuristics
6  |  label the current node with A_i
7  |  for each value v_{i.j} of the attribute A_i do
8  |  |  label an outgoing edge with value v_{i.j}
9  |  |  recursively train a sub-tree on a subset of training instances that
   |  |   meet the condition 'A_i = v_{i.j}'
10 |  end
11 end
```

Fig. 1. Decision tree learning algorithm [19]

this rule cover training instances of the same class. In this case, the leaf node is labelled with the majority class (i.e., labelling the leaf node with the class to which the majority of the covered training instances belong) and the growth of this branch is then terminated by setting the stopping criteria as satisfied. The whole procedure of the decision tree learning algorithm is shown in Fig. 1.

In terms of the heuristic attribute selection indicated in line 5 of the decision tree learning algorithm shown in Fig. 1, entropy [28] and information gain [26] are popularly used as the heuristics for the ID3 and C4.5 algorithms. In this context, the best attribute selected for each node needs to be capable of contributing to minimizing the entropy and maximizing the information gain.

2.3. Procedure of rule learning

Rule learning involves the selection of attribute-value pairs on an iterative basis. In particular, at each iteration of learning a single rule, an attribute-

value pair is selected to specialize the rule antecedent, i.e., the specialization of the rule antecedent is achieved by adding the selected attribute-value pair (rule term) as a part of the rule antecedent until the stopping criteria of learning this rule has been satisfied. Once the rule antecedent has been finalized after the above specialization, the rule would normally have covered some training instances that belong to the same class. In this case, the learning of the above rule is considered as completed, and it is then required to find all the instances that are covered by this rule and delete these instances from the training set, in order to initiate the learning of the next rule from the remaining instances. All the learned rules need to be added into a rule set that is eventually used as a rule based classifier. In this context, the production of the rule set is considered as completed, when each instance in the training set has been covered by one or more of the rules in the rule set. The whole procedure of the rule learning algorithm is shown in Fig. 2.

Input : A set of training instances T

Output: An ordered set of rules RS

```
1 while T ≠ φ do
2 |   generate a single rule R from the training set T
3 |   delete all instances covered by rule R
4 |   if the generated rule R is not good then
5 |   |   generate the majority rule and empty the training set T
6 |   end
7 |   add rule R into the rule set RS
8 end
```

Fig. 2. Rule learning algorithm[19]

The step shown in line 2 of the rule learning algorithm shown in Fig. 2 indicates that each rule is learned through iteratively selecting an attribute-value pair towards specializing the antecedent of this rule. The 'if-then' statement shown in line 4 indicates that the quality of the learned rule is not yet good

enough, but the learning of the rule has to be stopped due to the case that the maximum length of the rule has been reached, i.e., the length of a rule is essentially the number of rule terms that make up the rule antecedent, which must not be higher than the data dimensionality (the number of attributes in the data set). When the maximum length of a rule is really reached, it would usually indicate that all the attribute-value pairs have been evaluated for specializing the antecedent of this rule, but this rule still cannot fulfill to cover some training instances that belong to the same class. In this case, it is an usual practice to provide the rule consequent with the majority class, i.e., the class shown in the rule consequent is the one to which the majority of the covered training instances belong, and the learning of this rule is then completed by setting the stopping criteria as satisfied, similar to the setting of the stopping criteria of growing a decision tree branch when the maximum length of a branch is reached.

The nature of rule learning driven by the Prism algorithm is to iteratively select an attribute-value pair '$A_i = v_{i.j}$' for specializing the antecedent of a rule, while the rule consequent is predefined by assigning a target class c_t.

In terms of the heuristic section of attribute-value pairs, the conditional probability $P(class = c_t|A_i = v_{i.j})$ is incorporated into the Prism algorithm as the heuristic for iteratively selecting an attribute-value pair. In particular, while the target class c_t is assigned as the rule consequent, the attribute-value pair '$A_i = v_{i.j}$', which is selected to specialize the rule antecedent, needs to be capable of maximizing the conditional probability $P(class = c_t|A_i = v_{i.j})$.

In the whole procedure of the Prism algorithm shown in Fig. 3, T represents the original training set and T' is a subset of T. Each instance in T is expressed as e; AS represents an attribute set that contains d attributes; A_i represents an attribute in AS, where i is the attribute index; each value of attribute A_i is expressed as $v_{i.j}$, where j is the attribute-value index; c_t represents one of the k classes predefined in the training set T; p_{max} represents the maximum conditional probability of the target class c_t obtained by selecting the best attribute-value pair '$A_w = v_{w.b}$'; RS represents a set of rules that is finally used

Input : a training set T

Output: a rule set RS

Initialize: $T' = T$, $T'' = T'$, $t = 0$, $p_{max} = 0$;
for $t < k$ **do**
  while $\exists e : e \in T' \wedge e \in c_t$ **do**
    while $\exists e : e \in T' \wedge e \notin c_t$ **do**
      $i = 0$; $j = 0$; $p_{max} = 0$; **while** $i < d$ **do**
        for *each value* $v_{i.j}$ *of* A_i **do**
          Calculate $P(c_t|A_i = v_{i.j})$;
          if $P(c_t|A_i = v_{i.j}) > p_{max}$ **then**
            $p_{max} = P(c_t|A_i = v_{i.j})$; $w = i$; $b = j$;
          end
        end
        $x++$;
      end
      assign $A_w = v_{w.b}$ to R as a rule term; $AS = AS - \{A_w\}$;
      $d = d - 1$; $\forall e : T'' = T'' - \{e\}$, if $e \in T'$ and e does not comprise $A_w = v_{w.b}$;
    end
    $RS = RS \cup \{R\}$; $T' = T'$ - T'';
  end
  $T' = T$; $t++$;
end

Fig. 3. Prism algorithm [26]

as a rule based classifier; T'' represents the result set of instances covered by rule R in the rule set RS.

We illustrate the procedure of the Prism algorithm by using the Weather

data set[1] – see Table 1.

We replaced the original attribute values and the original class labels with specific symbols for the Weather data set. In particular, the attribute 'Outlook' has three values, namely, 'sunny', 'overcast' and 'rain', which are replaced with the three symbols '1', '2' and '3', respectively. The attribute 'Temperature' has three values, namely, 'hot', 'mild' and 'cool', which are replaced with the three symbols '1', '2' and '3', respectively. The attribute 'Humidity' has two values, namely, 'high' and 'normal', which are replaced with the two symbols '1' and '2', respectively. The attribute 'Windy' has two values, namely, 'true' and 'false',

Table 1
Weather data set

Outlook	Temperature	Humidity	Windy	Play?
1	1	1	0	N
1	1	1	1	N
2	1	1	0	Y
3	2	1	0	Y
3	3	2	0	Y
3	3	2	1	N
2	3	2	1	Y
1	2	1	0	N
1	3	2	0	Y
3	2	2	0	Y
1	2	2	1	Y
2	2	1	1	Y
2	1	2	0	Y
3	2	1	1	N

[1] `http://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/weather.nominal.arff`

which are placed with the two symbols '1' and '0', respectively. The two class labels 'Yes' and 'No' are replaced with the two symbols 'Y' and 'N', respectively.

The 'for' loop shown in line 2 of the Prism algorithm [5] shown in Fig. 3 indicates the need to select a target class at each iteration of producing the final rule set, in order to learn an independent set of rules for the selected target class at each iteration. The Weather data set contains instances which each belongs to one of the two classes ('Y' and 'N'), so two independent sets of rules need to be trained for the two classes, respectively, by using the Prism algorithm. In other words, one rule set consists of rules assigned the class 'Y' as the rule consequent, whereas the other rule set consists of rules assigned the class 'N' as the rule consequent.

In accordance with Table 1, a frequency table needs to be created for each attribute, i.e., there are totally four frequency tables (Tables 2-5) created, respectively, for the four attributes, namely, 'Outlook', 'Temperature', 'Humidity' and 'Windy'.

Table 2
Frequency table for the attribute 'Outlook'

Class label	Outlook= 1	Outlook= 2	Outlook= 3
Y	2	4	3
N	3	0	2
Total	5	4	5

Table 3
Frequency table for the attribute 'Temperature'

Class label	Temperature= 1	Temperature= 2	Temperature= 3
Y	2	4	3
N	2	2	1
Total	4	6	4

On the basis of the above frequency tables, the conditional probability of the target class can be calculated, when a specific attribute-value pair is given

Table 4

Frequency table for the attribute 'Humidity'

Class label	Humidity= 1	Humidity= 2
Y	3	6
N	4	1
Total	7	7

Table 5

Frequency table for the attribute 'Windy'

Class label	Windy= 1	Windy= 0
Y	3	6
N	3	2
Total	6	8

as the condition.

In accordance with Table 2, the conditional probability of each of the two target classes ('Y' and 'N') needs to be derived, when each of the three attribute-value pairs ($Outlook = 1$, $Outlook = 2$ and $Outlook = 3$) is given as the condition.

$P(Class = Y|Outlook = 1) = 0.40$

$P(Class = Y|Outlook = 2) = 1.00$

$P(Class = Y|Outlook = 3) = 0.60$

$P(Class = N|Outlook = 1) = 0.60$

$P(Class = N|Outlook = 2) = 0$

$P(Class = N|Outlook = 3) = 0.40$

In accordance with Table 3, the conditional probability of each of the two target classes ('Y' and 'N') needs to be derived, when each of the three attribute-value pairs ($Temperature = 1$, $Temperature = 2$ and $Temperature = 3$) is given as the condition.

$P(Class = Y|Temperature = 1) = 0.50$

$P(Class = Y|Temperature = 2) = 0.67$

$P(Class = Y|Temperature = 3) = 0.75$

$P(Class = N|Temperature = 1) = 0.50$

$P(Class = N|Temperature = 2) = 0.33$

$P(Class = N|Temperature = 3) = 0.25$

In accordance with Table 4, the conditional probability of each of the two target classes ('Y' and 'N') needs to be derived, when each of the two attribute-value pairs ($Humidity = 1$ and $Humidity = 2$) is given as the condition.

$P(Class = Y|Humidity = 1) = 0.43$

$P(Class = Y|Humidity = 2) = 0.86$

$P(Class = N|Humidity = 1) = 0.57$

$P(Class = N|Humidity = 2) = 0.14$

In accordance with Table 5, the conditional probability of each of the two target classes ('Y' and 'N') needs to be derived, when each of the two attribute-value pairs ($Windy = 1$ and $Windy = 0$) is given as the condition.

$P(Class = Y|Windy = 1) = 0.50$

$P(Class = Y|Windy = 0) = 0.75$

$P(Class = N|Windy = 1) = 0.50$

$P(Class = N|Windy = 0) = 0.25$

If the class 'Y' is selected as the target class, the attribute-value pair '$Outlook = 2$' leads to the maximized conditional probability, i.e., $P(Class = Y|Outlook = 2) = 1$. At this point, the uncertainty has been reduced to 0, since the conditional probability of 1 is obtained. Therefore, the stopping criteria is satisfied, which indicates that the antecedent of the first learned rule has been finalized. The expression of the learned rule is shown as: $Outlook = 2 \rightarrow class = Y$.

The first rule covers four instances, i.e., there are four instances that meet the condition '$Outlook = 2$'. Therefore, the four instances should be deleted from the training set, such that it can be initiated to learn the second rule from the updated training set (excluding the above four instances), while the target

class is still the class 'Y'. Finally, a complete set of rules for the target class 'Y' is obtained when each instance that belongs to the class has been covered by one or more of the trained rules.

If the class 'N' is selected as the target class, the first learned rule is expressed as $Outlook = 1 \wedge Humidity = 1 \rightarrow class = N$. Finally, the Prism algorithm leads to a complete set of rules that cover the instances of the target class 'N'. The final rule set, which is used as a whole rule based classifier, is obtained by merging the two independent sets of rules trained for the two target classes 'Y' and 'N', respectively.

2.4. Measures of rule quality

Popularly used rule quality measures include confidence [1], J-measure [45], lift [4] and leverage [39].

Confidence aims at measuring the degree that the consequent of the rule holds when the rule antecedent is satisfied, i.e., the percentage of training instances that satisfy the conditions reflected from the antecedent of this rule but also belong to the class shown in the rule consequent. The calculation of the rule confidence is shown in Eq. (1)

$$Confidence = \frac{P(x, y)}{P(x)} \tag{1}$$

where the term $P(x, y)$ represents the joint occurrence rate (joint probability) of the rule antecedent and the rule consequent and the term $P(x)$ represents the independent occurrence rate of the rule antecedent.

J-measure aims at measuring the quantity of the average information content covered by a learned rule. The essence of J-measure is the dot product of two terms as defined in Eq. (2).

$$J(Y, X = x) = P(x) \cdot j(Y, X = x) \tag{2}$$

The first term, which aims at measuring the simplicity of a rule [45], represents the independent occurrence rate of the rule antecedent $X = x$, since

a simpler rule generally leads to a higher value of $P(x)$. The second term is referred to as j-measure, which aims at measuring how well a single rule fits the training data and is also known as cross entropy [45]. The calculation of j-measure is shown in Eq. (3).

$$j(Y, X = x) = P(y|x) \cdot \log_2 \frac{P(y|x)}{P(y)} + (1 - P(y|x)) \cdot \log_2 \frac{1 - P(y|x)}{1 - P(y)} \tag{3}$$

where the term $P(y|x)$ represents the posterior probability that the rule consequent y occurs when the rule antecedent $X = x$ is satisfied, and the term $P(y)$ represents the prior probability that the rule antecedent y occurs without any specific preconditions.

Lift is a measure of the degree to which the joint occurrence of the antecedent and the consequent of a rule is higher than expected, given that they are statistically independent [4]. The calculation of the lift of a rule is shown in Eq. (4).

$$Lift = \frac{P(x, y)}{P(x) \cdot P(y)} \tag{4}$$

Leverage aims at measuring the difference between the actual joint probability and the expected joint probability in terms of the event that the rule antecedent and the rule consequent both occur [39]. The calculation of the leverage of a rule is shown in Eq. (5).

$$Leverage = P(x, y) - P(x) \cdot P(y) \tag{5}$$

3. Related work

Decision tree learning approaches have been adopted very popularly in machine learning tasks, because of the fact that this kind of learning approaches usually enable the generation of models with high accuracy and interpretability. In other words, a transparent process of reasoning is shown through using decision tree models, such that people can clearly identify how an output is mapped from an input [29].

On the other hand, decision tree learning has been very competitive to other popular machine learning approaches in terms of the algorithmic strengths, because the ID3 algorithm was developed in [41] demonstrating very high accuracy of classification especially on the chess end games data set [40]. Furthermore, due to the lack of the direct handling of continuous attributes in the ID3 algorithm, the C4.5 algorithm was thus developed as a successor of ID3 for the effective handling of continuous attributes and the replacement of missing values [43, 44].

In order to avoid the case of overfitting of a decision tree on training data, the development of pruning methods [42] has become necessary for simplifying decision trees. An empirical study was made in [16] for comparing different pruning methods. In addition, ensemble learning approaches have been adopted to create decision tree ensembles for advancing the overall performance of classification. Bagging [3] and Boosting [17] are two popular ensemble learning approaches, which have been used respectively for creating Random Forests and Gradient Boosted Trees as decision tree ensembles. On the other hand, some researchers have extended decision tree learning algorithms by modifying the heuristic functions for attribute selection. For example, cost functions could be incorporated for advancing the heuristic attribute selection, such that the risk of getting classification errors can be minimized. Some more recent works on cost functions based decision tree learning can be found in [22, 50]. Also, the heuristic attribute selection can be done in the setting of fuzzy logic. In particular, continuous attributes can be fuzzified for learning fuzzy decision trees through fuzzy information granulation [21].

It is guaranteed that a decision tree learned from a data set can be transformed into a set of non-overlapping rules, i.e., the rules mentioned above cover disjoint sets of training instances. The representation of a decision tree constrains that at least one attribute needs to appear at each branch, i.e., the attribute appears at the root node is the common attribute for all the branches of the decision tree. The above constraint usually results in the trained decision tree being complex and thus difficult to enable people to follow the information

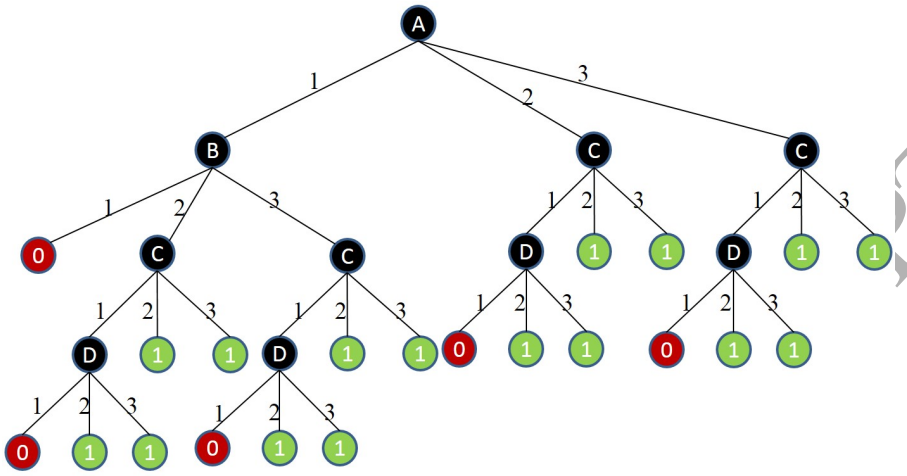

Fig. 4. Cendrowska's example of the replicated sub-tree problem [26, 33]

extracted from the tree [18]. For the purpose of reducing the complexity of decision trees, the use of pruning methods was investigated theoretically and experimentally in [42], such as the reduced error pruning method [15]. Unfortunately, even if pruning methods are used for simplifying decision trees, it is still difficult to avoid the generation of cumbersome, complex and inscrutable decision trees, leading to the difficulty in providing insight into a domain as applied knowledge [18, 43]. Also, for the purpose of the overfitting avoidance, simple rules are usually preferred to complex rules [18]. Moreover, it was argued in [5] that the adoption of decision tree learning algorithms is difficult to avoid the generation of a decision tree with replicated sub-trees shown in Fig. 4, due to the above mentioned constraint on having one or more common attributes appearing at all the branches of a decision tree. It is also argued in [5] that the existence of replicated sub-trees could result in the need of a complete traversal of a decision tree in the worst case for extraction of rules about a single classification, leading to an increased difficulty in the manipulation of expert systems.

In order to fulfill the need of solving the replicated sub-tree problem, it has been proposed to generate if-then rules directly through the SAC strategy. In

particular, the Prism algorithm [5] has been developed leading to more follow-up research on the SAC strategy based rule learning. A comprehensive review of the SAC strategy based rule learning algorithms can be found in [18]. For the Prism algorithm, the key feature is the need to learn an independent set of rules for each selected target class. For example, in a classification task that involves the two classes 'positive' and 'negative', the Prism algorithm would lead to training the first set of rules for identifying instances of the selected target class 'positive' and then training the second set of rules for identifying instances of the selected target class 'negative'. However, the above fixed strategy of the target class selection usually leads to the uncertainty on whether the selected target class provides a good basis for producing a high quality rule based on a given training sample. In fact, in compliance with the SAC strategy, rules are learned sequentially for each target class, i.e., the learning of one rule is started after the learning of the last rule has been completed, while the two rules are learned by selecting the same target class. In this context, if a low quality rule is learned, then it would result in a negative impact on learning all the subsequent rules, leading to low quality rules as the likely outcomes. From the above point of view, if the selected target class does not provide a good basis for getting a high quality rule learned, the Prism-driven rule learning strategy could even lead to an inconsistent rule being produced covering instances of different classes [26, 28]. In this case, the subsequent rules are likely to be poorly trained, due to the unexpected situation that the previously trained rule cannot cover instances of the same class. The above description indicates that the fixed strategy of the target class selection leads to a selection bias affecting the performance of rule learning driven by the Prism algorithm, i.e., a low quality rule is likely to be produced due to the ineffective target class selection for learning this rule, which indicates the need to develop a more effective strategy of the target class selection.

4. Rule learning driven by PrismCTC

In this section, we provide the description of the proposed PrismCTC algorithm on its essence and illustrate its procedure using the weather data set shown in Table 1. We also give a theoretical justification on the suitability of the PrismCTC algorithm for reducing both the risk of overfitting and the model complexity, through analyzing the main difference between Prism and PrismCTC in terms of their strategies of the target class selection.

4.1. Key features

As mentioned in Section 1, PrismCTC is essentially a variant of the Prism algorithm, and the key difference between the two algorithms is in terms of their strategies of the target class selection. In particular, the Prism algorithm involves a fixed strategy of the target class selection, whereas the PrismCTC algorithm involves a trained strategy, i.e., the target class selection through the PrismCTC algorithm is based on the outcome of the quality evaluation of rules trained for different classes. In this context, at each iteration of producing the final rule set, k rules are trained as the candidates for competition towards selecting the best target class from the k classes predefined in the training set, i.e., one of the k rules would become the successful candidate and is thus added into the final rule set. Therefore, the consequent of each rule shows a different target class. The quality of the k candidate rules should be evaluated by using one of the statistical measures defined in Eqs. (1)- (5), such that the rule of the highest quality can be found. Following the above evaluation of the rules quality, the target class shown in the consequent of the highest quality rule (the successful candidate) is judged as the best one. Overall, at each iteration of producing the final rule set, only the successful candidate is added into the final rule set, but the successful candidates (rules) identified at different iterations may show different rule consequent (i.e., different classes are selected as the target classes for these rules), e.g., the class 'Yes' is selected as the best target class leading to the first successful candidate added into the final rule set, but

Input : training set T

Output: rule set RS

Initialize: $T' = T$, $T'' = T'$, $t = 0$, $p_{max} = 0$, $q_{best} = 0$; **while** $T' \neq \phi$ **do**
  for $t = 0; t < k; t++$ **do**
    if $\exists e : e \in T' \wedge e \in c_t$ **then**
      while $\exists e : e \in T' \wedge e \notin c_t$ **do**
        $i = 0$; $j = 0$; $p_{max} = 0$; **while** $i < d$ **do**
          for *each value* $v_{i.j}$ *of* A_i **do**
            Calculate $P(c_t|A_i = v_{i.j})$; **if** $P(c_t|A_i = v_{i.j}) > p_{max}$ **then**
              $p_{max} = P(c_t|A_i = v_{i.j})$; $w = i$; $b = j$;
            end
          end
          $x++$;
        end
        assign $A_w = v_{w.b}$ to R_t as a rule term; AS= AS - $\{A_w\}$;
        $d--$; $\forall e : T'' = T'' - \{e\}$, if $e \in T'$ and e does not comprise $A_w = v_{w.b}$;
      end
      CS= $CS \cup \{R_t\}$; Calculate q_t of R_t; **if** $q_t > q_{best}$ **then**
        $q_{best} = q_t$; $R_{best} = R_t$
      end
    end
  end
  RS= $RS \cup \{R_{best}\}$; CS= ϕ; T'= T' - T'';
end

Fig. 5. The proposed PrismCTC algorithm

the second successful candidate results from selecting the class 'No' as the best target class.

In the whole procedure of the proposed PrismCTC algorithm shown in Fig. 5, T represents the original training set and T' is a subset of T; each instance in T is expressed as e; AS represents an attribute set that contains d attributes; A_i represents an attribute in AS, where i is the attribute index; each value of the attribute A_i is expressed as $v_{i.j}$, where j is the attribute-value index; c_t represents the target class selected from the k classes predefined in the original training set T; p_{max} represents the maximum conditional probability of the target class c_t obtained by selecting the best attribute-value pair '$A_w = v_{w.b}$'; R_t represents a candidate rule trained when selecting c_t as the target class, and q_t represents the quality of rule R_t; R_{best} represents the best (successful) candidate rule, and q_{best} represents the quality of the successful candidate R_{best}; RS represents a set of rules that is finally used as a rule based classifier; CS consists of the candidate rules and each one is trained for a specific one of the k classes; T" represents the result set of instances covered by the successful candidate rule R_{best} identified at each iteration.

We illustrate the proposed PrismCTC algorithm by following the illustration of the Prism algorithm on the Weather data set. In particular, as shown in Section 2.3, the first rule trained for the class 'Y' is expressed as: $Outlook = 2 \rightarrow class = 1$, whereas the first rule trained for the 'N' class is expressed as: $Outlook = 1 \wedge Humidity = 1 \rightarrow class = N$. For the PrismCTC algorithm, the quality of the above two rules need to be evaluated and one of them is selected as the successful candidate at the first iteration and is added into the final rule set. For illustrative purpose, we choose to measure the confidence (see Eq. (1)) of the two rules as follows:

The Weather data set contains 14 instances. For the rule of the class 'Yes', there are 4 instances that meet the condition '$Outlook = 2$', so $P(x) = \frac{4}{14} = 0.29$. Also, all of the four instances belong to the 'Yes' class, so $P(x, y) = \frac{4}{14} = 0.29$, which indicates that the confidence equals 1.

For the rule of the class 'N', there are 3 instances that meet the two conditions '$Outlook = 1$' and '$Humidity = 1$', so $P(x) = \frac{3}{14} = 0.21$. Also, all of the three instances belong to the class 'No', so $P(x, y) = \frac{3}{14} = 0.21$, which indicates that

the confidence equals 1 as well.

Since both rules obtain the confidence of 1, we need to measure the support of each rule according to Eq. (6). In this way, we can see that the rule of the class 'Y' obtains a higher value (0.29) of the support, so this rule would be selected as the successful candidate that gets into the final rule set and the best target class identified at the first iteration is thus the class 'Y'. The four instances covered by the rule of the class 'Y' should be deleted from the training set, such that the above procedure of rule learning can be repeated for the second iteration, by learning two rules (one for the class 'Y' and the other one for the class 'N') separately from the other 10 instances.

$$Support = P(x, y) \tag{6}$$

4.2. Justification

The designs of both the Prism and PrismCTC algorithms involve the adoption of granular computing concepts in an information processing paradigm [36, 37, 38]. The role of granular computing is to enable the effective thinking in a structured way at the philosophical level but granular computing can also act as a role leading to intelligent problem solving in a structured way at the practical level [48].

In practice, granular computing techniques generally need to involve granulation and organization as the two main operations [48]. The former is essentially an operation of transforming a whole into multiple parts, i.e., top-down information processing, whereas the latter is an operation of integrating multiple parts into a whole, i.e., bottom-up information processing [24, 25, 47].

What is actually processed through granulation and organization is referred to as an information granule, which can be seen as a collection of smaller particles forming a larger unit. Due to the difference in the sizes of different granules, the concept of information granularity is thus needed, which is aimed for each granule to be located at a suitable level of granularity according to the actual size of the granule.

In the setting of rule learning driven by the Prism algorithm, an independent set of rules needs to be learned for each predefined class. When there are k classes and each one should be selected as a target class, k independent rule sets would be trained to form a rule based classifier (granule at the top level of granularity) and each of the k rule sets is viewed as a sub-classifier (sub-granule) of the rule based classifier. Also, each rule in one of the k rule sets is viewed as a sub-sub-granule, because it is a part of a sub-classifier of the rule based classifier.

In the setting of information granularity, the Prism algorithm is designed to undertake the rule learning task in the sub-classifier level, which could result in the case that the rule set trained for each target class contains some but not all of the rules of high quality. In order to make the quality of each single rule as high as possible, the PrismCTC algorithm is designed to undertake the rule learning task in the rule level (in more depth), which manages to evaluate at each iteration the quality of all the rules trained for the given target classes.

As argued in Section 3, rule learning driven by the Prism algorithm leads to the target class selection bias, since the selection is done in a fixed strategy by selecting a target class from the k predefined classes in the training set and then learning rules continuously for the same class until each of the instances of this class has been covered by one or more of the learned rules. However, at each iteration of producing the final rule set, the best target class that leads to the highest quality rule may not be the same one. Since it is impossible to identify at each iteration which one of the k class is the best target class without having a rule trained for each of the k classes, the PrismCTC algorithm is thus designed to let each class be the target class leading to a candidate rule and evaluating the quality of all the candidate rules, as illustrated in Fig. 5. In the above way, it can ensure that a rule of as high quality as possible is trained and identified at each iteration and is then added into the final rule set that will be used as a rule base classifier.

5. Experimental results

In this section, experiments are conducted for evaluating the proposed Prism-CTC algorithm by comparing its performance with the performance of the C4.5 algorithm [43] and the Prism algorithm [5], in terms of the classification accuracy and the model complexity.

Table 6
Data sets.

Data sets	Attribute types	No. of attributes	No. of instances	No. of classes
Anneal	discrete, continuous	38	898	6
Balance-scale	discrete	4	625	3
Breast-cancer	discrete	9	286	2
Breast-w	continuous	9	699	2
Credit-a	discrete, continuous	15	690	2
Credit-g	discrete, continuous	20	1000	2
Cylinder-bands	discrete, continuous	39	540	2
Dermatology	discrete, continuous	34	366	6
Diabetes	discrete, continuous	8	768	2
Hepatitis	discrete, continuous	19	155	2
Ionosphere	continuous	34	351	2
Iris	continuous	4	150	3
kr-vs-kp	discrete	36	3196	2
Labor	discrete, continuous	16	57	2
Lymph	discrete, continuous	18	148	4
Sponge	discrete	45	76	3
Tae	discrete, continuous	5	151	3
Vote	discrete	16	435	2
Wine	continuous	13	178	3
Zoo	discrete, continuous	17	101	7

In order to measure the classification accuracy obtained by using each learning algorithm, each data set is partitioned into two subsets (i.e., 70% for training and 30% for testing), for the conduct of the experiments. The data partitioning is repeated 100 times on each data set and the average accuracy obtained through the 100 runs of the experiments is taken for comparison. The model complexity is measured by counting the rules and the terms in a rule set. In particular, a rule based classifier is trained on each whole data set towards counting the rules and the terms that make up the classifier. In terms of rule quality evaluation, confidence, J-measure, lift and leverage are adopted, respectively, for the effective identification of the best quality rule resulting from the best target class, at each iteration of the rule set learning.

There are 20 UCI data sets [23] used by following the above procedures, for the performance evaluation against the above three learning algorithms. The descriptions of the 20 data sets are given in Table 6 to show their characteristics, such as the number of attributes.

5.1. Accuracy comparison

Table 7 shows the results on the classification accuracy. The four headers of Table 7 'PrismCTC1', 'PrismCTC2', 'PrismCTC3' and 'PrismCTC4' represent that the four statistical measures (confidence, J-measure, lift and leverage) shown in Eqs (1)-(5) are used, respectively, for the heuristic evaluation of the quality of each single rule, immediately after the rule has been learned by using the proposed PrismCTC algorithm. In other words, the PrismCTC algorithm is considered as parametric, where each of the above four statistical measures can be used as a parameter for rule quality evaluation, e.g., PrismCTC1 represents the case of choosing the statistical measure 'confidence' as the parameter for evaluating each rule learned by using the PrismCTC algorithm.

In comparison with C4.5, PrismCTC1 leads to better classification accuracy in 11 out of 20 cases. In the other 9 cases, PrismCTC1 performs the same as or very close to C4.5, with exceptions on the 'Credit-a' and 'Zoo' data sets. PrismCTC2 leads to better classification accuracy in 10 out of 20 cases. In the

Table 7

Classification accuracy.

Data sets	C4.5	Prism	PrismCTC1	PrismCTC2	PrismCTC3	PrismCTC4
Anneal	0.98	0.98	**0.99**	**0.99**	**0.99**	0.98
Balance-scale	0.78	0.83	**0.85**	**0.85**	0.84	**0.85**
Breast-cancer	**0.67**	**0.67**	0.66	0.65	0.64	**0.67**
Breast-w	0.94	0.93	**0.95**	**0.95**	**0.95**	**0.95**
Credit-a	**0.83**	0.80	0.77	0.77	0.78	0.81
Credit-g	0.68	**0.74**	0.70	0.68	0.68	0.70
Cylinder-bands	0.58	0.69	0.70	0.70	0.69	**0.72**
Dermatology	**0.94**	0.84	0.90	0.91	0.88	0.85
Diabetes	0.72	0.70	0.70	0.69	0.70	**0.73**
Hepatitis	0.76	0.76	0.82	0.81	0.78	**0.83**
Ionosphere	0.89	0.90	**0.92**	**0.92**	**0.92**	**0.92**
Iris	**0.94**	0.88	**0.94**	**0.94**	0.93	0.92
Kr-vs-kp	**0.99**	0.98	0.98	**0.99**	**0.99**	0.98
Labor	0.80	**0.88**	0.81	0.85	0.87	0.84
Lymph	0.76	0.78	**0.79**	0.77	0.78	0.76
Sponge	**0.93**	0.91	0.90	**0.93**	**0.93**	0.92
Tae	0.53	0.49	**0.59**	0.57	0.58	0.45
Vote	**0.95**	0.93	0.94	0.94	0.94	0.90
Wine	0.91	0.84	0.93	0.93	0.90	**0.94**
Zoo	**0.92**	0.61	0.80	0.86	0.63	0.86

other 10 cases, PrismCTC2 performs the same as or very close to C4.5, with exceptions on the 'Credit-a' and 'Zoo' data sets. PrismCTC3 leads to better classification accuracy in 9 out of 20 cases. In the other 11 cases, PrismCTC3 performs the same as or very close to C4.5, with exceptions on the 'Credit-a', 'Dermatology' and 'Zoo' data sets. PrismCTC4 leads to better classification accuracy in 9 out of 20 cases. In the other 11 cases, PrismCTC4 performs the

same as or very close to C4.5, with exceptions on the 'dermatology', 'Tae', 'Vote' and 'Zoo' data sets.

In comparison with Prism, PrismCTC1 leads to better classification accuracy in 13 out of 20 cases. In the other 7 cases, PrismCTC1 performs the same as or very close to Prism, with an exception on the 'Labor' data set. PrismCTC2 leads to better classification accuracy in 14 out of 20 cases. In the other 6 cases, PrismCTC2 performs the same as or very close to Prism, with an exception on the 'Credit-g' data set. PrismCTC3 leads to better classification accuracy in 13 out of 20 cases. In the other 7 cases, PrismCTC3 performs the same as or very close to Prism, with an exception on the 'Credit-g' data set. PrismCTC4 leads to better classification accuracy in 12 out of 20 cases. In the other 8 cases, PrismCTC4 performs the same as or very close to Prism.

Overall, the results shown in Table 7 indicate that in 10 out of 20 cases at least one of the four rule quality measures leads to the PrismCTC algorithm outperforming both C4.5 and Prism in terms of the classification accuracy. In the other 10 cases, the best performance of the PrismCTC algorithm achieved using one of the four rule quality measures is equal to or slightly worse than the performance of the best performing one of the other two algorithms (C4.5 and Prism), with an exception on the 'Zoo' data set.

To further evaluate the performance of the PrismCTC algorithm, the Wilcoxon sign rank test is applied to identify wheather the average accuracy obtained by using PrismCTC is higher than the average ones obtained by using C4.5 and Prism, respectively. As shown in [14], it would be more appropriate to use the Wilcoxon sign rank test than to use the paired t-test.

Table 8 displays the Wilcoxon's test details when comparing Prism with the different versions of PrismCTC; PrismCTCBest takes the best performance from the four versions of PrismCTC. The results of the comparison between C4.5 and the different versions of PrismCTC are displayed in Table 9.

N denotes the number of cases (data sets), which are divided into three possibilities: negative ranks, positive ranks and ties. The negative ranks indicate the number of cases in which the second algorithm outperforms the first one,

Table 8

Wilcoxon sign rank tests for average accuracy – Prism vs PrismCTC.

Compared classifiers	Ranks	N	Mean ranks	Sum of ranks	z-score	p-value
Prism vs PrismCTC1	Negative ranks	13	9.77	127.00	-1.79	p=3.70%
	Positive ranks	5	8.80	44.00		
	Ties	2				
	Total	20				
Prism vs PrismCTC2	Negative ranks	14	10.86	152.00	-1.76	p=4.00%
	Positive ranks	6	9.67	58.00		
	Ties	0				
	Total	20				
Prism vs PrismCTC3	Negative ranks	13	8.73	113.50	-1.73	p=4.20%
	Positive ranks	4	9.88	39.50		
	Ties	3				
	Total	20				
Prism vs PrismCTC4	Negative ranks	12	8.42	101.00	-1.14	p=12.80%
	Positive ranks	5	10.40	52.00		
	Ties	3				
	Total	20				
Prism vs PrismCTCBest	Negative ranks	17	10.18	173.00	-3.14	p=0.10%
	Positive ranks	2	8.50	17.00		
	Ties	1				
	Total	20				

while the positive ranks indicate the opposite; ties are cases in which the algorithms have the same accuracy. The sum of ranks (i.e., mean ranks × N) allows the comparison of the two classifiers, i.e., similar sums for the negative and positive ranks indicate a negligible difference, i.e., the algorithms have similar performance; when one of the sums is considerably higher than the other one, it indicates a significant difference between the two algorithms. The z-test is used

Table 9

Wilcoxon sign rank tests for average accuracy – C4.5 vs PrismCTC.

Compared classifiers	Ranks	N	Mean ranks	Sum of ranks	z-score	p-value
C4.5 vs PrismCTC1	Negative ranks	11	10.32	113.50	-0.75	p=22.80%
	Positive ranks	8	9.56	76.50		
	Ties	1				
	Total	20				
C4.5 vs PrismCTC2	Negative ranks	10	8.55	85.50	-0.91	p=18.30%
	Positive ranks	6	8.42	50.50		
	Ties	4				
	Total	20				
C4.5 vs PrismCTC3	Negative ranks	9	9.61	86.50	-0.45	p=32.60%
	Positive ranks	8	8.31	66.50		
	Ties	3				
	Total	20				
C4.5 vs PrismCTC4	Negative ranks	9	9.11	82.00	-0.24	p=40.60%
	Positive ranks	8	8.88	71.00		
	Ties	3				
	Total	20				
C4.5 vs PrismCTCBest	Negative ranks	12	9.04	108.50	-2.09	p=1.80%
	Positive ranks	4	6.88	27.50		
	Ties	4				
	Total	20				

to calculate the level of significance of the difference between the algorithms – in general, a p-value lower than 0.05 indicates significance.

For example, when comparing Prism with PrismCTC1, the sum of negative ranks, i.e., 127.00, (indicating PrismCTC1 outperforms Prism), is much higher than the sum of positive ranks, i.e., 44.00, (indicating that Prism outperforms PrismCTC1). The z-score (-1.79) and it's corresponding p-value (3.70%) indi-

cate that PrismCTC significantly outperforms Prism. The mean ranks, sum of ranks, z-scores and p-values are rounded to 2 decimal places.

The Wilcoxon rank test results indicate that, with the exception of PrismCTC4, all versions of PrismCTC significantly outperform Prism. The comparison between C4.5 and the different versions of PrismCTC indicate a similar performance, with the exception of PrismCTCBest. This indicates the proposed PrismCTC algorithm performs as well as C4.5, and that there is potential for this algorithm (through joint use of the different rule quality measures) to outperform established algorithms such as C4.5.

5.2. Model complexity comparison

In terms of model complexity, the results on the number of rules are shown in Table 10, whereas the results on the number of terms are shown in Table 11.

In comparison with C4.5, the PrismCTC algorithm leads to a simpler rule based classifier in 19 out of 20 cases, when using any one of the four rule quality measures, according to the results on the number of terms shown in Table 11. In other words, although the PrismCTC algorithm leads to more rules being produced than C4.5 in most cases as shown in Table 10, the number of terms impacts the model complexity more significantly, as argued in [26].

In comparison with Prism, the PrismCTC algorithm leads to a smaller number of rules in 17 out of 20 cases, when using any one of the four rule quality measures. In the other 3 cases (on the 'Credit-a', 'Tae' and 'Vote' data sets), the PrismCTC algorithm just leads to the same or a marginally higher number of rules, when using some of the four rule quality measures. On the other hand, the PrismCTC algorithm leads to fewer terms being generated in all of the 20 cases, when using any one of the four rule quality measures.

The above description indicates that the PrismCTC algorithm can effectively leads to reduction of the model complexity, in comparison with C4.5 and Prism. In practice, the reduction of the model complexity could not only reduce the risk of overfitting, but also leads to a faster process of rule based classification,

Table 10

A comparison of the number of rules generated by different algorithms for different data sets.

Data sets	C4.5	Prism	PrismCTC1	PrismCTC2	PrismCTC3	PrismCTC4
Anneal	53	48	31	20	19	16
Balance-scale	60	109	106	101	121	83
Breast-cancer	152	110	107	103	96	44
Breast-w	23	36	30	27	28	15
Credit-a	101	148	148	142	110	77
Credit-g	359	310	214	280	248	181
Cylinder-bands	430	310	160	177	160	168
Dermatology	33	48	35	37	33	24
Diabetes	22	223	173	203	172	103
Hepatitis	16	30	23	24	19	15
Ionosphere	18	39	21	20	19	17
Iris	5	16	6	6	6	6
Kr-vs-kp	43	101	77	67	52	51
Labor	13	12	6	5	5	5
Lymph	23	32	23	26	28	15
Sponge	14	10	9	5	5	5
Tae	35	44	57	61	65	21
Vote	19	27	28	27	27	16
Wine	5	16	8	6	10	6
Zoo	9	35	15	10	30	10

since it is essential to check rule by rule in an outer loop and to check term by term in an inner loop, towards finding a rule that fires for classifying an unseen instance. Moreover, a simpler model is easier to interpret – one of the reasons for the popularity of decision trees is the ability to inspect the model and understand the reasoning based on which the classification is made. Thus, PrismCTC has the potential to lead to models that are easier to interpret than

Table 11

A comparison of the number of terms generated by different algorithms for different data sets.

Data sets	C4.5	Prism	PrismCTC1	PrismCTC2	PrismCTC3	PrismCTC4
Anneal	400	81	40	29	27	24
Balance-scale	419	359	308	292	322	258
Breast-cancer	645	329	248	220	210	115
Breast-w	124	87	42	40	39	26
Credit-a	542	392	197	188	187	180
Credit-g	2261	882	457	355	385	413
Cylinder-bands	432	312	160	177	160	168
Dermatology	141	95	40	42	45	25
Diabetes	120	680	305	261	243	216
Hepatitis	81	51	31	27	33	18
Ionosphere	121	92	23	22	21	19
Iris	14	35	8	8	8	7
Kr-vs-kp	380	387	195	134	136	130
Labor	44	15	7	5	6	5
Lymph	93	74	36	42	52	27
Sponge	40	10	9	5	5	5
Tae	242	115	105	95	113	51
Vote	97	88	74	80	83	40
Wine	12	21	9	6	11	6
Zoo	38	35	15	10	30	10

decision trees.

6. Conclusions

In this paper, we have proposed a new version of the Prism algorithm referred to as PrismCTC, and the experimental results show that the proposed PrismCTC algorithm outperforms the C4.5 and Prism algorithms, in terms of

classification accuracy and model complexity.

We have proved through the experiments that the incorporation of the heuristic target class selection into the PrismCTC algorithm through a trained strategy leads to effective reduction of the model complexity, by means of producing a smaller number of simpler rules, comparing with the original Prism algorithm that involves the target class selection in a fixed strategy. Also, the trained strategy of the target class selection involved in the PrismCTC algorithm can also lead to reduction of overfitting, in comparison with the fixed strategy involved in the original Prism algorithm. In particular, confidence, J-measure, lift and leverage are incorporated into the PrismCTC algorithm, respectively, for evaluating the quality of each single rule, such that it is achievable to identify at each iteration which class is the best option for being the target class leading to the best quality rule. In this way, each rule, which is trained at a specific iteration and is finally added into the rule set, would be of as high quality as possible. In contrast, the original Prism algorithm simply chooses each of the predefined classes as the target class for training an independent set of rules until each of the instances of the target class has been covered by one or more of the already trained rules. In this way, it is usually difficult to achieve that the trained rule set contains all rules of high quality, i.e., the quality of some rules in the rule set are low.

In future, we will investigate in more depth how to measure the rule quality more effectively. For example, multiple measures of rule quality can be used jointly towards reduction of the bias resulting from a single measure and improvements of the effectiveness of the rule quality evaluation [30]. It is also worth to investigate granular computing techniques [2, 12, 27, 31] towards multi-granularity rule learning [25]. In addition, the fuzzy set theory [49] has been applied to deal with many applications [7, 11, 13, 20, 35, 46]. The adoption of the fuzzy set theory will be explored for learning of fuzzy rules [6, 8, 9, 10, 34] to deal with data with high fuzziness, e.g., textual data.

Acknowledgements

This work is supported by the University of Portsmouth, UK, under the Research Development Fund, and is supported by the Ministry of Science and Technology, Republic of China, under Grant MOST 107-2221-E-011-122 -MY2.

References

References

[1] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: ACM SIGMOD international conference on Management of data, vol. 13, Washington, D.C., USA, 207–216, 1993.

[2] M. Antonelli, P. Ducange, B. Lazzerini, F. Marcelloni, Multi-objective evolutionary design of granular rule-based classifiers, Granular Computing 1 (1) (2016) 37–58.

[3] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.

[4] S. Brin, R. Motwani, J. D. Ullman, S. Tsur, Dynamic itemset counting and implication rules for market basket data, in: ACM SIGMOD international conference on Management of data, Tucson, Arizona, USA, 255–264, 1997.

[5] J. Cendrowska, Prism: An algorithm for inducing modular rules, International Journal of Man-Machine Studies 27 (4) (1987) 349–370.

[6] S. M. Chen, A fuzzy reasoning approach for rule-based systems based on fuzzy logics, IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics 26 (5) (1996) 769–778.

[7] S. M. Chen, Aggregating fuzzy opinions in the group decision-making environment, Cybernetics and Systems 29 (4) (1998) 363–376.

[8] S. M. Chen, Y. C. Chang, Weighted fuzzy rule interpolation based on GA-based weight-learning techniques, IEEE Transactions on Fuzzy Systems 19 (4) (2011) 729–744.

[9] S. M. Chen, Y. C. Chang, J. S. Pan, Fuzzy rules interpolation for sparse fuzzy rule-based systems based on interval type-2 Gaussian fuzzy sets and genetic algorithms, IEEE Transactions on Fuzzy Systems 21 (3) (2013) 412–425.

[10] S. M. Chen, S. H. Lee, C. H. Lee, A new method for generating fuzzy rules from numerical data for handling classification problems, Applied Artificial Intelligence 15 (7) (2001) 645–664.

[11] S. M. Chen, A. Munif, G. S. Chen, H. C. Liu, B. C. Kuo, Fuzzy risk analysis based on ranking generalized fuzzy numbers with different left heights and right heights, Expert Systems with Applications 39 (7) (2012) 6320–6334.

[12] S. M. Chen, K. Tanuwijaya, Fuzzy forecasting based on high-order fuzzy logical relationships and automatic clustering techniques, Expert Systems with Applications 38 (12) (2011) 15425–15437.

[13] H. S. Chiang, M. Y. Chen, Z. W. Wu, Applying fuzzy petri nets for evaluating the impact of bedtime behaviors on sleep quality, Granular Computing 3 (4) (2018) 321–332.

[14] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (2006) 1–30.

[15] T. Elomaa, M. Kaariainen, An analysis of reduced error pruning, Journal of Artificial Intelligence Research 15 (1) (2001) 163–187.

[16] F. Esposito, D. Malerba, G. Semeraro, Simplifying decision trees by pruning and grafting: New results, in: European Conference on Machine Learning, vol. 912, Crete, Greece, 287–290, 1995.

[17] Y. Freund, R. E. Schapire, Experiments with a new boosting algorithm, in: International Conference on Machine Learning, Bari, Italy, 148–156, 1996.

[18] J. Furnkranz, Separate-and-conquer rule learning, Artificial Intelligence Review 13 (1) (1999) 3–54.

[19] I. Kononenko, M. Kukar, Machine Learning and Data Mining: Introduction to Principles and Algorithms, Horwood Publishing Limited, Chichester, West Sussex, 2007.

[20] Y. F. Lai, M. Y. Chen, H. S. Chiang, Constructing the lie detection system with fuzzy reasoning approach, Granular Computing 3 (2) (2018) 169–176.

[21] Y. Lertworaprachaya, Y. Yang, R. John, Interval-valued fuzzy decision trees with optimal neighbourhood perimeter, Applied Soft Computing 24 (2014) 851–866.

[22] X. Li, H. Zhao, W. Zhu, A cost sensitive decision tree algorithm with two adaptive mechanisms, Knowledge-Based Systems 88 (2015) 24–33.

[23] M. Lichman, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml, 2013.

[24] H. Liu, M. Cocea, Fuzzy information granulation towards interpretable sentiment analysis, Granular Computing 2 (4) (2017) 289–302.

[25] H. Liu, M. Cocea, Granular Computing Based Machine Learning: A Big Data Processing Approach, Springer, Berlin, 2018.

[26] H. Liu, M. Cocea, Induction of classification rules by Gini-Index based rule generation, Information Sciences 436-437 (2018) 227–246.

[27] H. Liu, M. Cocea, Granular computing based approach of rule learning for binary classification, Granular Computing 4 (2).

[28] H. Liu, A. Gegov, Induction of modular classification rules by information entropy based rule generation, in: V. Sgurev, R. R. Yager, J. Kacprzyk, V. Jotsov (Eds.), Innovative Issues in Intelligent Systems, vol. 623, 217–230, 2016.

[29] H. Liu, A. Gegov, M. Cocea, Network based rule representation for knowledge discovery and predictive modelling, in: IEEE International Conference on Fuzzy Systems, Istanbul, Turkey, 1–8, 2015.

[30] H. Liu, A. Gegov, M. Cocea, Collaborative rule generation: An ensemble learning approach, Journal of Intelligent and Fuzzy Systems 30 (4) (2016) 2277–2287.

[31] H. Liu, A. Gegov, M. Cocea, Rule based systems: A granular computing perspective, Granular Computing 1 (4) (2016) 259–274.

[32] H. Liu, A. Gegov, M. Cocea, Rule Based Systems for Big Data: A Machine Learning Approach, Springer, Switzerland, 2016.

[33] H. Liu, A. Gegov, F. Stahl, J-measure based hybrid pruning for complexity reduction in classification rules, WSEAS Transactions on Systems 12 (9) (2013) 433–446.

[34] H. Liu, L. Zhang, Fuzzy rule-based systems for recognition intensive classification in granular computing context, Granular Computing 3 (4) (2018) 355–365.

[35] S. Liu, Z. Xu, J. Gao, A fuzzy compromise programming model based on the modified S-curve membership functions for supplier selection, Granular Computing 3 (4) (2018) 275–283.

[36] W. Pedrycz, S. M. Chen, Granular Computing and Intelligent Systems: Design with Information Granules of Higher Order and Higher Type, Springer, Heidelberg, 2011.

[37] W. Pedrycz, S. M. Chen, Granular Computing and Decision-Making: Interactive and Iterative Approaches, Springer, Heidelberg, 2015.

[38] W. Pedrycz, S. M. Chen, Information Granularity, Big Data, and Computational Intelligence, Springer, Heidelberg, 2015.

[39] G. Piatetsky-Shapiro, Discovery, analysis, and presentation of strong rules, in: G. Piatetsky-Shapiro, W. J. Frawley (Eds.), Knowledge Discovery in Databases, AAAI/MIT Press, Cambridge, 229–248, 1991.

[40] J. R. Quinlan, Learning efficient classification procedures and their application to chess end games, in: R. S. Michalski, J. G. Carbonell, T. M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, 463–482, 1983.

[41] J. R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106.

[42] J. R. Quinlan, Simplifying decision trees, International Journal of Man-Machine Studies 27 (3) (1987) 221–234.

[43] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Francisco, 1993.

[44] J. R. Quinlan, Improved use of continuous attributes in C4.5, Journal of Artificial Intelligence Research 4 (1) (1996) 77–90.

[45] P. Symth, R. M. Goodman, An information theoretic approach to rule induction from databases, IEEE Transactions on Knowledge and Data Engineering 4 (4) (1992) 301–316.

[46] H. Y. Wang, S. M. Chen, Evaluating students' answerscripts using fuzzy numbers associated with degrees of confidence, IEEE Transactions on Fuzzy Systems 16 (2) (2008) 403–415.

[47] J. Yao, Information granulation and granular relationships, in: IEEE International Conference on Granular Computing, Beijing, China, 326–329, 2005.

[48] Y. Yao, Perspectives of granular computing, in: IEEE International Conference on Granular Computing, Beijing, China, 85–90, 2005.

[49] L. Zadeh, Fuzzy sets, Information and Control 8 (3) (1965) 338–353.

[50] H. Zhao, X. Li, A cost sensitive decision tree algorithm based on weighted class distribution with batch deleting attribute mechanism, Information Sciences 378 (2017) 303–316.