Granul. Comput. (2016) 1:259–274
DOI 10.1007/s41066-016-0021-6

ORIGINAL PAPER

Rule-based systems: a granular computing perspective

Han Liu[1] · **Alexander Gegov**[1] · **Mihaela Cocea**[1]

Received: 6 November 2015 / Accepted: 3 May 2016 / Published online: 11 May 2016

Abstract A rule-based system is a special type of expert system, which typically consists of a set of if–then rules. Such rules can be used in the real world for both academic and practical purposes. In general, rule-based systems are involved in knowledge discovery tasks for both purposes and predictive modeling tasks for the latter purpose. In the context of granular computing, each of the rules that make up a rule-based system can be seen as a granule. This is due to the fact that granulation in general means decomposition of a whole into several parts. Similarly, each rule consists of a number of rule terms. From this point of view, each rule term can also be seen as a granule. As mentioned above, rule-based systems can be used for the purpose of knowledge discovery, which means to extract information or knowledge discovered from data. Therefore, rules and rule terms that make up a rule-based system are considered as information granules. This paper positions the research of rule-based systems in the granular computing context, which explores ways of achieving advances in the former area through the novel use of theories and techniques in the latter area. In particular, this paper gives a certain perspective on how to use set theory for management of information granules for rules/rule terms and different types of computational logic for reduction of learning bias. The effectiveness is critically analyzed and discussed. Further directions of this research area are recommended towards achieving advances in rule-based systems through the use of granular computing theories and techniques.

✉ Han Liu
Han.Liu@port.ac.uk

[1] School of Computing, University of Portsmouth, Buckingham Building, Lion Terrace, Portsmouth PO1 3HE, UK

Keywords Data mining · Machine learning · Rule-based systems · Granular computing · Deterministic logic · Probabilistic logic · Fuzzy logic

1 Introduction

A rule-based system is a special type of expert system, which is made up of a set of rules, which typically takes the form of if–then rules. In general, such rules can be designed through the use of expert knowledge or through learning from real data. On the basis of this viewpoint, rule-based systems can be designed in two ways: expert-based design and data-based design. The former follows traditional engineering approaches whereas the latter follows machine learning approaches. Due to the vast and rapid increase in data size, machine learning approaches have, thus, become increasingly popular towards the design of rule-based systems. The rest of this section focuses on description of the rule-based systems in the machine learning context.

As introduced in Liu et al. (2014), rules can be used for different tasks, e.g., classification, regression and association, and thus these rules are referred to as classification rules, regression rules and association rules, respectively. In terms of classification rules, a unfied framework for design of rule-based classification systems, which is made up of rule generation, rule simplication and rule representation, was developed in Liu et al. (2015b). In particular, rule generation approaches can be divided into two categories: divide and conquer (Quinlan 1993) and separate and conquer (Fürnkranz 1999). Rule simplification can be done through use of pruning algorithms (Fürnkranz 1999), which can be specialized into the following two types: pre-pruning and post-pruning. Rule representation is aimed at

managing the computational complexity and interpretability for rule-based models (Liu et al. 2015a). A more detailed explanation of the above framework is presented in Sect. 3.

As mentioned in Hu and Shi (2009), Zadeh (1997) proposed that granulation is a basic concept that underlies human cognition and generally involves decomposition of a whole into several parts. In this context, each of the rules or rule bases that make up a rule-based system can be considered as a granule. Similarly, each of the rule terms that make up a single rule is also considered as a granule. As introduced in Liu et al. (2015a), rule-based systems can be used in practice for the purpose of knowledge discovery, which means to extract knowledge or information discovered from data. In this context, rules, rule bases and rule terms are seen as information granules.

As introduced in Fürnkranz (1999), the existing rule generation approaches mentioned above have three main biases, one of which is referred to as 'search bias'. The search bias means the way that the hypothesis space is searched and is essentially originated from the so-called 'greedy search' strategy. For the divide and conquer approaches, such as ID3 (Quinlan 1986), the search bias can result from the fact that attribute selection is based on the average entropy calculated for each single attribute and the attribute with the minimum entropy is selected. In this context, selection of the attribute with the minimum entropy can only manage to minimize the uncertainty remaining in the training set at the current iteration. In other words, the recursive partition of the training set can only lead to local optimization rather than global optimization towards reduction of the uncertainty at each iteration. The same problem may also arise with the separate and conquer approaches when a particular attribute-value pair is selected at each iteration leading to local optimization towards reduction of the uncertainty remaining in the training set at each iteration. On the basis of the above description, this paper pays attention to the use of granular computing techniques towards the search of a globally optimal set of rules in terms of rule quality. Similar work has also been proposed in Yao and Yao (2002), but the approach proposed in this paper involves the use of the concept of granular structures that will be described in Sect. 4.

On the other hand, in machine learning tasks, the presence of missing values in training and test data is a far large issue that needs to be resolved effectively in practice. Two popular resolutions are to replace any missing values with the most frequently occurring values for discrete attributes or the average values for continuous attributes, and to delete any instances with missing values from the data set, respectively (Kononenko and Kukar 2007). However, reasons for missing values could be varied in practice (Pigott 2001). In other words, missing values can happen on a random or artificial basis. For example, people may forget to put details on a particular field when they fill in online forms. This can be considered as random missing of values. In contrast, an online form may appear to have some fields not required, i.e., people do not have to fill in details on these fields. This would be considered as artificial missing of values. In addition, on this basis, it is also possible that some details are still pending and thus not known yet leading to the occurrence of missing values such as student exam results. More reasons for missing values are explained in Pigott (2001) in detail. On the basis of the above description, this paper proposes a technique through use of the rough set theory towards appropriate handling of missing values when their absence is on the artificial basis mentioned above. This proposal is inspired by the concept that a rough set contains a boundary region that indicates insufficient knowledge about the set for judging the membership of an element (Pawlak 1982).

The rest of this paper is organized as follows: Sect. 2 presents theoretical preliminaries relating to rule-based systems and granular computing. Section 3 presents a unified framework for the design of rule-based classification systems. Section 4 provides an overview of granular computing concepts in the context of set theory and information granulation. The relationship between rule-based systems and granular computing is argued in Sect. 4. Section 4 also explores ways of achieving advances in rule-based systems through novel use of granular computing theories and techniques. In particular, a certain perspective is given to show how to use set theory for management of information granules for rules/rule terms. In addition, this section also justifies how different types of computational logic can be used effectively towards reduction of bias from rule learning algorithms. Section 5 critically discusses the effectiveness of granular computing techniques towards advances in rule-based systems. Section 6 summarizes the contributions of this paper and provides recommendations for further directions of research in rule-based systems in the context of granular computing.

2 Theoretical preliminaries

As mentioned in Sect. 1, some fundamental concepts strongly relate to rule-based systems, machine learning and granular computing. In particular, these concepts include if–then rules, computational logics, statistical measures and rough sets. This section describes these concepts in detail.

2.1 If–then rules

As introduced in Sect. 1, a rule-based system is made up of a set of rules. Ross (2004) described that a varied number

of ways have been used for knowledge representation in engineering applications of artificial intelligence, but one of the most popular ways is to take the form of if–then rules denoted by the expression: IF cause (antecedent) THEN effect (consequent).

The above expression typically provides an inference that if the input (cause, antecedent, condition) is given, then the output (effect, consequent, outcome) can be derived (Ross 2004). In this paper, each item that makes up a condition (the left hand side of a rule) is read as a rule term. Gegov (2007) introduced that both the left hand side (antecedent) and the right hand side (consequent) of a rule can contain multiple terms (inputs/outputs). In this context, an antecedent that contains multiple conditions (input terms) linked by 'and' connectives is referred to as a conjunctive antecedent, whereas an antecedent that consists of the input terms that are linked by 'or' connectives is referred to as a disjunctive antecedent. The above concepts related to antecedents also apply to the consequents of a rule. Also, it is presented in Gegov (2007) that rules would be conjunctive, if all of these rules are linked through use of 'and' connectives, or disjunctive, if any of these rules are connected through use of 'or' connectives. A rule may also be inconsistent, which indicates the possibility that a rule has the same antecedent mapped to a number of different consequents. In this case, the rule would appear to have its antecedent conjunctive and its consequent disjunctive. In addition, rules with the same set of input attributes on their left hand side can make up a rule base and more details about the concept of rule bases can be seen in Gegov (2010).

As mentioned in Sect. 1, rules can be applied for different practical tasks, e.g., classification, regression and association. In this paper, rules are mainly used as prediction techniques for classification tasks. Therefore, each of the rules is called a classification rule, which can contain multiple terms on its left hand side, but only a single term on its right hand side. In a classification rule, the consequent expresses the class which unseen instances are assigned and the rule antecedent expresses the adequate condition that any unseen instances need to meet in order for these instances to be assigned this class. A rule set that is used for classification may contain overlapped rules. This indicates the possibility that different rules could cover some common instances. In this context, if these overlapping rules have different consequents on their right hand side, an issue, which is called classification conflict, would arise. In this case, conflict resolution is used towards resolving the issue according to the specific criteria such as weighted voting or fuzzy inference (Ross 2004). The presence of inconsistent rules usually leads to uncertainty in classification. This is mainly due to the case that the prediction towards a class becomes non-deterministic when the conflict of classification issue arises. More details about the resolution of conflicts and handling inconsistent rules can be found in Liu et al. (2015a).

2.2 Computational logic

Ross (2004) stated that logic is a small part of the capability of human reasoning, which can assist people towards making decisions or judgments. A basic type of logic is known as Boolean logic in computer science. As presented in Liu et al. (2015a), in the context of Boolean logic, each variable is binary, which means that the value of such a variable is 0 (false) or 1 (true). This indicates that reasoning and judgment that are made without uncertainty would normally lead to deterministic outcomes. From this viewpoint, Boolean logic can also be called deterministic logic. However, it is quite usual in reality that people can only make practical operations under uncertainty such as decision making, reasoning and judgment. Due to the above case, the other three types of computational logic, namely probabilistic logic, fuzzy logic and rough logic, have thus become more popular approaches. Each of the three types of logic can be seen as an extension of deterministic logic. The three types of computational logic mentioned above are mainly different from deterministic logic in terms of the truth value, which is numerical between 0 and 1 rather than binary (0 or 1). The numerical truth value expresses a probability of getting one of the binary truth values in probabilistic logic and a membership degree of truth in fuzzy logic as well as a possibility of truth in rough logic. The rest of this subsection presents the key features of the four types of computational logic mentioned above and discusses the main difference among them as well as in what ways they are related to the concept of rule-based systems. The differences among the four types of computational logic are also compared in the perspectives of any related statistical heuristics, set theory and corresponding event type in Table 1.

Deterministic logic handles certain events. For example, all elements in a crisp set should fully belong to the set without uncertainty, i.e., each of these elements is certainly assigned a full membership to the above set.

Probabilistic logic handles random events under probabilistic uncertainty. For example, in a probabilistic set, an element may be randomly put into the set with a certain probability. An element must be given a full membership to the set once the element has been put into the above probabilistic set.

Fuzzy logic handles events under non-probabilistic uncertainty. In this context, each set is known as a fuzzy set, which is due to the fact that each of the elements in such a set may only be given a partial membership to the

Table 1 Comparison among deterministic, probabilistic, fuzzy and rough logic

Logic type	Related heuristics	Related set theory	Related event type
Deterministic Logic	Binary truth value	Crisp set	Certain event
Probabilistic Logic	Probability	Probabilistic set	Random event
Fuzzy Logic	Fuzzy truth value	Fuzzy set	Fuzzy event
Rough Logic	Possibility	Rough set	Possible event

set, i.e., each of these elements belongs to the fuzzy set to a certain degree.

Rough logic handles events under uncertainty which results from incomplete information. In the context of set theory, a rough set is a special type of set, which restores information on the basis of different subsets of attributes (Yao 2004). In other words, in a rough set, all instances belong to the set subject to specific conditions by means of employing a boundary region of the set (Pawlak 1982). For example, an instance belongs to a rough set subject to two conditions, which have the weight of 0.7 and 0.3, respectively. In this context, if the first condition is met and the second condition is still pending, then the possibility for the instance to belong to the rough set is 0.7.

In the context of rule-based systems, if deterministic logic is adopted, then each rule as a part of a rule-based system would happen to either fire or not without any uncertainty. If the rule happens to fire, the consequent would be deterministic. If probabilistic logic is adopted, then each rule as a part of a rule-based system would be given a firing probability. The consequent would be probabilistic depending on the posterior probability of the consequent of the rule given the specific antecedent. If fuzzy logic is adopted, then each rule as a part of a rule-based system would be given a firing strength. The consequent would be weighted depending on the fuzzy truth value of the most likely outcome. Also, it is required for fuzzy rule-based systems to handle continuous attributes through mapping these numerical values to a finite number of fuzzy linguistic terms in accordance with those fuzzy membership functions defined. If rough logic is adopted, each rule as a part of a rule-based system would be given a firing possibility, when there are missing values in test instances against some of these attribute-value pairs to be judged about their firing status as part of the antecedents of these rules. In particular, each input term as a part of the antecedent of a rule contributes a possibility towards firing this rule, while this input term is firing. Therefore, the firing possibility of a rule is equal to the sum of all the possibilities to which the input terms of the rule contribute.

Overall, deterministic rules do not accept overlapping of instances, which means that any rule sets covered by such rules should have all instances belong to the same class. In contrast, probabilistic and fuzzy rules accept such overlapping of instances, which indicates that rule sets covered by these two types of rules may have instances belong to different classes. When probabilistic rules are used, test instances can only be assigned a single class with a certain probability. However, fuzzy rules typically assign a test instance multiple classes with different degrees of membership. In other words, the test instance belongs to one class to a certain extent and to another class to another extent. In contrast, rough rules do not accept overlapping of instances as mentioned above and assign a test instance a single class subject to specific conditions. In other words, the test instance belongs to a single class subject to one or more conditions, each of which is assigned a certain weight.

2.3 Statistical measures

In the context of rule learning, statistical measures are usually used as heuristics for generation, simplification and evaluation of rules. This subsection presents several measures, namely entropy, J-measure and confidence.

Entropy is a measure of uncertainty, which is introduced in Shannon (1948). Entropy E can be calculated in the way illustrated in Eq. (1):

$$E = -\sum_{i=0}^{n} p_i \cdot \log_2 p_i \tag{1}$$

where p is read as the probability of the occurrence of event i and i is used as the index of a particular event.

The J-measure is introduced in Smyth and Rodney (1992), which is an information theoretic measure of average information content of a single rule. The J-measure can be calculated through the product of two terms as illustrated in Eq. (2):

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

where the first term $P(x)$ is known as the probability of the occurrence of the left hand side of a rule and used as a measure of simplicity (Smyth and Rodney 1992). Besides, the second term is known as j-measure, which was first introduced in Blachman (1968), but later modified in Smyth and Rodney (1992). This term is used as a measure of goodness-of-fit of a single rule (Smyth and Rodney 1992). The j-measure is calculated in the way as illustrated in Eq. (3):

$$j(Y, X = x) = P(y|x) \cdot \log\left(\frac{P(y|x)}{P(y)}\right) + (1 - P(y|x)) \cdot \log\left(\frac{1 - P(y|x)}{1 - P(y)}\right) \quad (3)$$

where $P(y)$ is known as the prior probability of the occurrence of the consequent of a rule and $P(y|x)$ is read as the posterior probability of the occurrence of the rule consequent with the rule antecedent given as the condition.

Also, the j-measure has its upper bound called jmax as introduced in Smyth and Rodney (1992) and can be calculated as illustrated in Eq. (4):

$$j(Y,\ X\ =\ x)\ \leq\ \max\left(P(y|x) \cdot \log\left(\frac{1}{P(y)}\right), (1 - P(y|x)) \cdot \log\left(\frac{1}{1 - P(y)}\right)\right) \quad (4)$$

However, it is generally possible that the class to which the rule will eventually be assigned to reflect the rule consequent is unknown. In this case, the j-measure can be calculated through taking into account all of the possible classes as illustrated in Eq. (5):

$$j(Y,\ X = x)\ =\ \sum_{i=0}^{n} P(y_i|x) \cdot \log\left(\frac{P(y_i\,|x)}{P(y_i)}\right) \quad (5)$$

In the above case, the corresponding jmax needs to be calculated through the way illustrated in Eq. (6):

$$j(Y,\ X\ = x) \leq\ \max_i\left(P(y_i\,|x) \cdot \log\left(\frac{1}{P(y_i)}\right)\right) \quad (6)$$

Confidence is introduced in Agrawal et al. (1993), which is used to measure the predictive accuracy of a single rule, i.e., the extent to which the consequent of the rule is reliable when the corresponding antecedent of the rule is satisfactory. The confidence can be calculated in the way illustrated in Eq. (7):

$$\text{Conf} = \frac{P(x,\ y)}{P(x)} \quad (7)$$

where $P(x,\ y)$ is known as the joint probability that the antecedent and consequent of a rule commonly occur and $P(x)$ is known as the prior probability provided with the same essence as given in the J-measure introduced above.

A more detailed overview of the above statistical measures can be found in Tan et al. (2004) and Geng and Hamilton (2006). Section 4 will illustrate in what way these measures are effective for the purpose of rule learning.

3 Design of rule-based classification systems

As introduced in Sect. 1, a unified framework for the design of rule-based classification systems consists of three operations: rule generation, rule simplification and rule representation. This section illustrates the three operations in detail.

As mentioned in Sect. 1, the generation of rules can be achieved following two approaches, namely, divide and conquer and separate and conquer. The aim of the former is to generate rules in the form of decision trees on an inductive basis such as ID3 (Quinlan 1986) and C4.5 (Quinlan 1993), whereas the aim of the latter is to generate if–then rules directly from training instances on an iterative basis such as Prism (Cendrowska 1987) and Information Entropy-Based Rule Generation (IEBRG) (Liu et al. 2014).

The divide and conquer approach is also known as Top-Down Induction of Decision Trees (TDIDT). This is because of the fact that rules generated using this approach are represented in the form of decision trees and that the induction procedure is from general to specific like the top-down approach (Avison and Fitzgerald 2006) in the context of software engineering. The basic procedure of the TDIDT is illustrated in Fig. 1.

The separate and conquer approach is also known as the covering approach. This is because of the fact that this approach typically involves generating if–then rules sequentially. In particular, the aim of this approach is generating a rule which covers the instances belonging to the same class and then iteratively starting the generation of the next rule through the use of the rest of the training instances that should not have been covered by the rules that are generated previously. In other words, all the above instances covered by the previously generated rules need to have been deleted from the current training subset. The separate and conquer approach basically works following the procedures illustrated in Fig. 2.

In terms of rule simplification, as mentioned in Sect. 1, it can be achieved by adopting pruning algorithms towards reduction of both overfitting and complexity of computational models. In particular, pruning algorithms can be specialized into two types, namely, pre-pruning and post-pruning. The former pruning generally means that pruning actions are taken when rules are being generated whereas the latter pruning means that pruning actions are taken after the rule generation is completed. A special type of pruning algorithms, which are based on J-measure, is introduced in Sect. 2.3. The rest of this subsection focuses on the illustration of J-measure-based pruning.

With regard to pruning of decision trees, the aim of pre-pruning is stopping a particular branch in a tree growing further. The stopping criteria are typically determined

Fig. 1 Decision tree learning algorithm (Kononenko and Kukar 2007)

Input: A set of training instances, attribute A_i, where *i* is the index of the attribute A, value V_j, where *j* is the index of the value V
Output: A decision tree.
if the stopping criterion is satisfied **then**
create a leaf that corresponds to all remaining training instances
else
choose the best (according to some heuristics) attribute A_i
label the current node with A_i
for each value V_j of the attribute A_i **do**
label an outgoing edge with value V_j
recursively build a subtree by using a corresponding subset of training instances
end for
end if

Input: A set of training instances
Output: An ordered set of rules
while training set is not empty **do**
generate a single rule from the training set
delete all instances covered by this rule
if the generated rule is not good **then**
generate the majority rule **and** empty the training set
end if
end while

Fig. 2 Rule covering approach (Kononenko and Kukar 2007)

using statistical heuristics such as the J-measure. Due to the nature of decision tree learning, it is not known which class can eventually be labelled on the leaf node of a particular branch when this branch is stopped growing further. Therefore, the value of J-measure would be calculated following the Eqs. (2) and (5) as illustrated in Sect. 2.3. In this case, the corresponding Jmax value, which is used to reflect the upper bound of J-measure, can be calculated following the Eqs. (2) and (6) illustrated in Sect. 2.3. The basic procedure of J-measure-based pre-pruning of decision trees is illustrated in Fig. 3.

As mentioned above, the stopping criteria against further growth of a tree branch can be made according to the values of J-measure and Jmax. In particular, the growth of a branch in a tree could be stopped once the highest value of J-measure observed so far already gets higher than the Jmax value achieved at the current node. Also, the stopping could be made once the value of J-measure has been equal to the Jmax value at the current node.

On the other hand, simplification of decision trees can also be achieved through use of post-pruning algorithms. In this context, the aim of the pruning action is at simplifying any particular branches in a decision tree after the whole tree has been generated. In some cases, it is necessary to covert the tree into a set of single rules prior to the pruning action being taken. When J-measure is used, the procedure of post-pruning is illustrated in Fig. 4. However, post-pruning of decision trees can also be done through replacing a subtree with a leaf node when other statistical heuristics are used. A popular method used in this strategy is referred to as Reduced Error Pruning (REP) (Elomaa and Kääriäinen 2001).

With regard to pruning of if–then rules, it is different from that of decision trees. In particular, the pruning action

Fig. 3 J-measure-based pre-pruning of decision trees (Liu et al. 2016b)

Input: a tree node *n*, J-measure of a node *J(n)*, Jmax of a node *Jmax(n)*, Maximum of J-measure *max(J),* class *C*
Output: a decision tree T
Initialize: *max(J)*= 0;
for each branch **do**
calculate *J(n)* and *Jmax(n)*
if *J(n)*> *max(J)* **then**
max(J)= *J(n)*
end if
if stopping criteria is met **Then**
stop this branch growing, label the current node *n* with the most common class *C*
end if
end for

Fig. 4 J-measure-based post-pruning of decision trees (Liu et al. 2016b)

Input: a set of rules *rule[i]*, where *i* is the index of rule, values of J-measure *J[i][j][k]*, where *j* is the order of a particular child rule and *k* is the index of a child rule in the order of *j*, highest value of J-measure (for *rule[i]*) *max[i]*, order for the optimal rule *m,* index for the optimal rule *n[i]*

Output: a set of simplified rules

```
for each rule rule[i] do
    max[i]=0, m=0, n[i]=0
    for each order j
        for each child rule k
            calculate J[i][j][k]
            if J[i][j][k]> max[i] then
              max[i]= J[i][j][k]
              m=j
              n[i]=k
            end if
            else if J[i][j][k]= max[i]
                take the rule with a higher order
            end else if
        end for
    end for
end for
```

Fig. 5 J-measure-based rule pre-pruning (Liu et al. 2016b)

Input: a set of rules *rule[i]*, where *i* is the index of rule, values of J-measure *J[i][j][k]*, where *j* is the order of a particular child rule and *k* is the index of a child rule in the order of *j*, highest value of J-measure (for *rule[i]*) *max[i]*, order for the optimal rule *m,* index for the optimal rule *n[i]*

Output: a set of simplified rules

```
for each rule rule[i] do
    max[i]=0, m=0, n[i]=0
    for each order j
        for each child rule k
            calculate J[i][j][k]
            if J[i][j][k]> max[i] then
              max[i]= J[i][j][k]
              m=j
              n[i]=k
            end if
            else if J[i][j][k]= max[i]
                take the rule with a higher order
            end else if
        end for
    end for
end for
```

for if–then rules needs to be made per single rule generated. In other words, each single rule is pruned immediately once this rule has been completely generated, which indicates that the pruning action needs to be taken before starting the generation of the next rule rather than after the completion of the generation of a whole rule set.

In the context of pruning of if–then rules, the aim of pre-pruning is stopping the specialization of the left hand side of a rule. The aim of post-pruning would be simplifying the left hand side of a rule after the generation of this rule has been completed. When J-measure is used as the heuristics, the procedure of the pre-pruning of if–then rules is illustrated in Fig. 5.

The post-pruning of if–then rules, which is based on J-measure, is similar to the strategy involved in pruning of decision trees. A main difference to tree pruning is that the pruning action is taken immediately after the generation of a single rule is completed and before the start of generating the next rule rather than after the end of generating the whole rule set.

In terms of rule representation, it is argued in Liu et al. (2015a) that appropriate representation of rules is highly important towards improvement of model efficiency and interpretability. In particular, decision tree, linear list and rule-based network are considered to be three main representation techniques, which are illustrated using the set of rules below as an example.

Rule 1: if $x1 = 0$ and $x2 = 0$ then $y = 0$;
Rule 2: if $x1 = 0$ and $x2 = 1$ then $y = 0$;
Rule 3: if $x1 = 1$ and $x2 = 0$ then $y = 0$;
Rule 4: if $x1 = 1$ and $x2 = 1$ then $y = 1$;

The above set of rules is already represented in a linear list which is in the form of if–then rules as defined in Liu et al. (2015a).

With regard to decision tree representation, the above rule set would be represented as illustrated in Fig. 6. In this representation, the root node or each of the internal nodes represents an input attribute. Each of the branches splitting from the root or an internal node represents a condition judgement. Each of the leaf nodes represents a class label.

With regard to rule-based network representation, the above rule set would be represented as illustrated in Fig. 7. In this network topology, the nodes (e.g., x_1 and x_2) in the input layer represent input attributes. Each node in the conjunction layer represents a rule antecedent, and the corresponding consequent of the same rule is a class label which is represented as a node in the output layer. In addition, each of the connections between the nodes in the input and conjunction layers represents the condition judgement, and each of the connections between the nodes in the last two layers (conjunction and output) represents the mapping between a rule antecedent and a rule consequent.

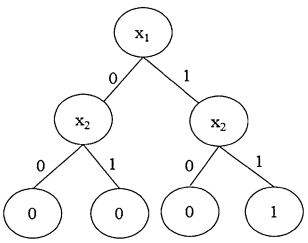

Fig. 6 Decision Tree representation

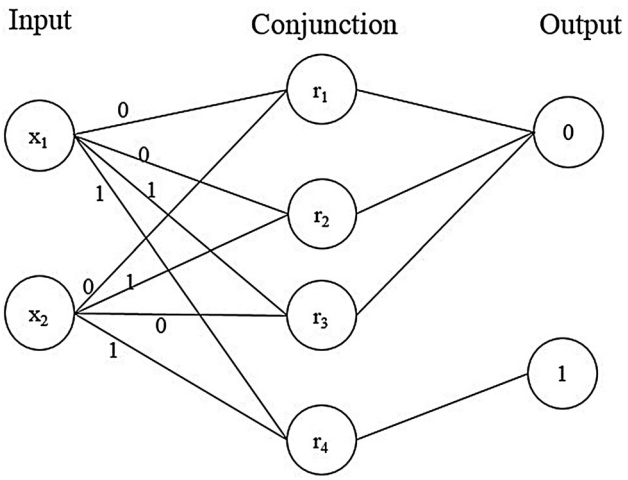

Fig. 7 Rule-based network representation

4 Granular computing-based rule learning

Sections 2 and 3 described theoretical preliminaries of rule-based systems and granular computing as well as a framework for the design of rule-based classification systems. As mentioned in Sect. 1, rules and rule terms can be considered as information granules, which shows the relationship between rule-based systems and granular computing. This section particularly justifies such a relationship, as well as explores in what way granular computing techniques contribute to rule learning towards advances in this research area.

4.1 Overview of granular computing

Granular computing is an emerging approach of information processing. It is applied with two main aims as pointed out in Yao (2005). One aim is at structured thinking at the philosophical level and the other one is at structured problem solving at the practical level. The fundamentals of granular computing generally involve information granulation which includes probabilistic sets, fuzzy sets and rough sets. As also mentioned in Sect. 1, granulation generally means to decompose a whole into several parts. The rest of this subsection focuses on description of granular computing concepts in the context of set theory and granulation.

In the context of probabilistic sets, each set employs a chance space which can be partitioned into a number of subspaces. Each of the subspaces can be viewed as a granule that can be randomly selected towards wining of a chance. In this context, all these granules make up the chance space mentioned above. As introduced in Sect. 2.2, in a probabilistic set, each element is granted a probability to get the full membership to the set. In the context of

granular computing, the probability can be viewed as a percentage of the granules that make up the chance space. For example, if the probability for an element to get the full membership is 70 %, then the element would be assigned 70 % of the granules (70 % chance). In this case, if any one of the granules is selected on a random basis, then the element will be granted the full membership to the set.

In the context of fuzzy sets, as mentioned in Sect. 2.2, each element has a certain degree of membership to the set. The membership can be partitioned into a number of parts, each of which can be viewed as a granule. From this point of view, if an element has a membership degree of 70 %, then it would be considered that an element is assigned 70 % of the granules. This is very similar to the example that an academic society offers different levels of memberships, which grant the members with different levels of access to the resources.

In the context of rough sets, as mentioned in Sect. 2.2, a set employs a boundary region to restore some elements under insufficient information. In other words, all those elements within the boundary region would only be given conditional memberships since these elements have only partially met the conditions towards being members of the set. Once the conditions are met, then full memberships will be granted to the elements. In this context, the condition for an element to become a member of the set can be partitioned into a number of sub-conditions, each of which can be viewed as a granule. As mentioned in Sect. 2.2, possibility is used to measure the extent to which the condition is satisfactory. Therefore, if the possibility that an element belongs to the set is 70 %, then it would be considered that the element has met 70 % of the granules (sub-conditions).

On the basis of the above description, granular computing is very useful to simplify a complex problem through decomposing it into several sub-problems in practice. It can also be used to measure a qualitative attribute in a quantitative way in the context of information granulation. In machine learning research, due to the presence of insufficient data, it is always required to measure uncertainty properly. Therefore, it is necessary to position the research of rule learning in the context of granular computing towards the improvement of model accuracy, and the relationship between rule-based systems and granular computing is argued in Sect. 4.2.

4.2 Relationship between rule-based systems and granular computing

As introduced in Yao (2006), the ideas of granular computing have been involved in the areas such as the theory of hierarchy, computational intelligence, artificial intelligence, divide and conquer and the theory of small groups. In particular, granular computing is given as a basis of data mining, such as rule mining and representation. On the basis of the above description, the relationship between rule-based systems and granular computing can be outlined as follows:

Firstly, a rule-based system is a special type of systems, which consists of a set of rules or rule bases. Each of the rules is also made up of a number of rule terms. Each of the rule bases consists of rules that have the same set of attributes. From this viewpoint, the concept of rule-based systems is linked to granular computing in the context of the theory of hierarchy. In particular, the concept of rule-based systems involves a four level structure, i.e., rule set, rule base, rule and rule term.

Secondly, in the context of computational intelligence, as described in Yao (2006), the notion of information granulation was first time introduced in Zadeh (1979) and the fuzzy set theory was suggested as a technique for applications in this perspective. Later, rough set theory was proposed in Pawlak (1982) to show the significance of the notion of information granulation. On this basis, Yao (2006) defined each granule as a set and a family of sets as the granular structure. In the context of rule learning, a rule set is used to restore a group of rules that are learned from a particular data set by the same algorithm. From this point of view, a rule set is considered as a granule. In addition, a set is used to restore a subset of training instances covered by a single rule or any one of its child rules, which again shows the connections to information granulation. More details on the use of set theory are presented in Sect. 4.3.

Thirdly, in the context of artificial intelligence, Yao (2006) described the fact that the role of the notion of granules is significant in some studies such as representation, searching and reasoning of knowledge. In the context of rule-based systems, rule representation has been defined as an important operation in Liu et al. (2015a) for the purpose of searching and reasoning towards the derivation of an output, which emphasizes the role of the notion of granules in the study of rule-based systems.

Fourthly, as mentioned in Sect. 2.3, divide and conquer is a popular approach for decision trees learning. In addition, divide and conquer is popularly used as a search strategy in the subject of data structures as well as prediction of unseen instances (Liu et al. 2015a). On the basis of the above description, the concept of rule-based systems is linked to the granular computing in terms of knowledge learning and searching through the divide and conquer strategy.

Finally, in the context of the theory of small groups, as mentioned in Yao (2006), a small group can be seen as a granule. As introduced in Arrow et al. (2000), groups can be studied as complex systems which are adaptive and dynamic. Such groups need to have the following

characteristics: interaction among group members, interaction between different groups and the contexts of groups (Yao 2006).

In this context, as introduced in Sect. 3, the unified framework for design of rule-based systems consists of three components: rule generation, rule simplification and rule representation. If each of the three components is viewed as a group, then ID3 and C4.5 can be seen as the members of the group. Similarly, decision trees, linear lists and rule-based networks can also be viewed as members of the group for rule representation. As argued in Liu et al. (2015b), the three groups have interactions with each other. For example, rule generation methods interact with rule simplification methods in the process of rule learning. In addition, when the divide and conquer approach is adopted for generation of rules, the rule-based model is automatically represented in the form of a decision tree while the rules are being generated. Also, when rules are generated using the separate and conquer approach, the rule-based model is automatically represented in the form of if–then rules in the meantime. The above description indicates that the above framework for design of rule-based systems demonstrates the characteristic of interaction among group members.

With regard to interaction between different groups, a special type of rule-based systems, which is referred to as ensemble rule-based systems (Liu and Gegov 2015), can demonstrate this characteristic. For example, a random forest is a group of decision trees. As each decision tree is viewed as a group of rules, the random forest can thus be seen as an ensemble of groups. In fact, these decision trees that make up a random forest have collaborations with each other in the testing stage for the final prediction of unseen instances. The above fact indicates that ensemble rule-based systems demonstrate the characteristic of interaction between different groups.

4.3 Applications of granular computing techniques for rule learning

Section 4.2 argued the relationship between rule-based systems and granular computing. This subsection explores in detail how granular computing techniques can be applied towards advances in rule learning. In particular, set theory is used for management of information granules for rules or rule terms. Probabilistic, fuzzy and rough logic are used as techniques for handling uncertainty in rule-based classification tasks. A rule-based network topology is used for representation of rules.

As mentioned in Sect. 3, each rule covers a set of instances that are used in the training stage. In this context, each set of instances covered by a single rule is viewed as a granule. In accordance with the justification in Sect. 4.2, such a granule could also be seen as a set. Therefore, such a set of instances that are covered by a rule is seen as a set. In addition, a rule would have a finite number of child rules, which is very similar to a finite set having a finite number of subsets. From this point of view, each child rule also covers a set of instances, which could be viewed as a set. The set S_i that contains the instances covered by a child rule R_i is actually the superset of the set S that contains the instances covered by the corresponding parent rule R. In other words, S is equal to the intersection of all the sets S_i where $i = 0, 1, 2 \ldots k$. The mathematical notation is shown in Eq. (9):

$$S = \bigcap_{i=0}^{k} S_i \tag{9}$$

On the basis of the above description, a parent rule and each of its child rules are viewed as granules, each of which covers instances that belong to a set. The family that consists of these sets is viewed as a granular structure in accordance with the justification in Sect. 4.2. The granular structure can be built following the procedure illustrated by the following example:

Suppose there is a three order rule: if $x = 1$ and $y = 1$ and $z = 1$, then class $= 1$ that covers instances belonging to set d. It would have 6 child rules, each of which covers instances belonging to one of the numbered sets as follows: a, b, c, e, f, and g. The corresponding six child rules are listed, respectively, as below:

if $x = 1$, then class $= 1$;
if $x = 1$ and $y = 1$, then class $= 1$;
if $y = 1$, then class $= 1$;
if $x = 1$ and $z = 1$, then class $= 1$;
if $y = 1$ and $z = 1$, then class $= 1$;
if $z = 1$, then class $= 1$;

The full family of sets is represented in the following:

Order 3: {d}
Order 2: {b, e, f}
Order 1: {a, c, g}

The above granular structure was created through the following procedure:

Iteration 1: when the first rule term $x = 1$ is appended, the set a is created accordingly and the parent rule is in the form: if $x = 1$ then class $= 1$;

Iteration 2: when the second rule term $y = 1$ is appended, the set b is created accordingly and the parent rule is specialized in the form: if $x = 1$ and $y = 1$ then class $= 1$; then one child rule is generated in the form: if $y = 1$ then class $= 1$; and the set c is created accordingly.

Iteration 3: when the third rule term $z = 1$ is appended, the set d is created and the parent rule is specialized in the form: if $x = 1$ and $y = 1$ and $z = 1$ then class $= 1$; in

addition, the three child rules are generated subsequently in the forms, respectively: if $x = 1$ and $z = 1$ then class $= 1$; if $y = 1$ and $z = 1$ then class $= 1$; and if $z = 1$ then class $= 1$; also, the three sets e, f and g are created accordingly. The creation of the granular structure illustrated above is complete in this iteration.

The importance of constructing the granular structure illustrated above is justified in Sect. 5.

In terms of probabilistic logic, it can be used for handling uncertainty when rules are used to make predictions. As introduced in Sect. 2.2, probabilistic logic generally means to solve problems under probabilistic uncertainty. In the context of rule learning, all generated rules are supposed to be inconsistent, which means that each of such rules can be mapped to different consequents with different levels of confidence. For example, if $x = 1$ and $y = 0$ then $z = 0$ (70 % chance) or $z = 1$(30 % chance); the above example indicates that if the condition is met, then the output variable has 70 % chance to equal to 0 and 30 % chance to equal to 1.

In terms of fuzzy logic, it can be also used for handling uncertainty when rules are used to make predictions. As introduced in Sect. 2.2, fuzzy logic generally means to solve problems under non-probabilistic uncertainty. In the context of rule learning, all generated rules are supposed to be inconsistent, which means that each of such rules can be mapped to different consequents with different degrees of membership. For example, if $x = 1$ and $y = 0$ then $z = 0$ (70 % membership) or $z = 1$(30 % membership); the above example indicates that if the condition is met, then the output variable equals to 0 with the membership degree of 70 % and to 1 with the membership degree of 30 %.

In terms of rough logic, it can be also used for handling uncertainty when rules are used to make predictions. As introduced in Sect. 2.2, rough logic generally means to solve problems under insufficient knowledge. In the context of rule learning, it is possible that testing data contain missing values against some attribute-value pairs to be judged about their firing status as part of the left hand side of a rule. For example, if $x = 1$(70 % weight) and $y = 0$ (30 % weight), then $z = 0$; the above example indicates that if the value of x is missing in a test instance but the value of y is 0 then the possibility that z equals to 0 is 30 %. This means that the value of 0 is assigned subject to the condition that x equals to 1 with the weight of 0.7. In other words, the condition for the variable z to be assigned the value of 0 is only 30 % satisfied due to the case that the value of x is still pending.

On the basis of the above description, it also indicates the needs to represent probabilistic, fuzzy or rough rules to improve the interpretability of rule-based models. A rule-based network topology is developed through modifications made to the topology illustrated in Fig. 7, and the modified version is illustrated in Fig. 8.

This network topology can be applied to any types of computational logic such as deterministic logic,

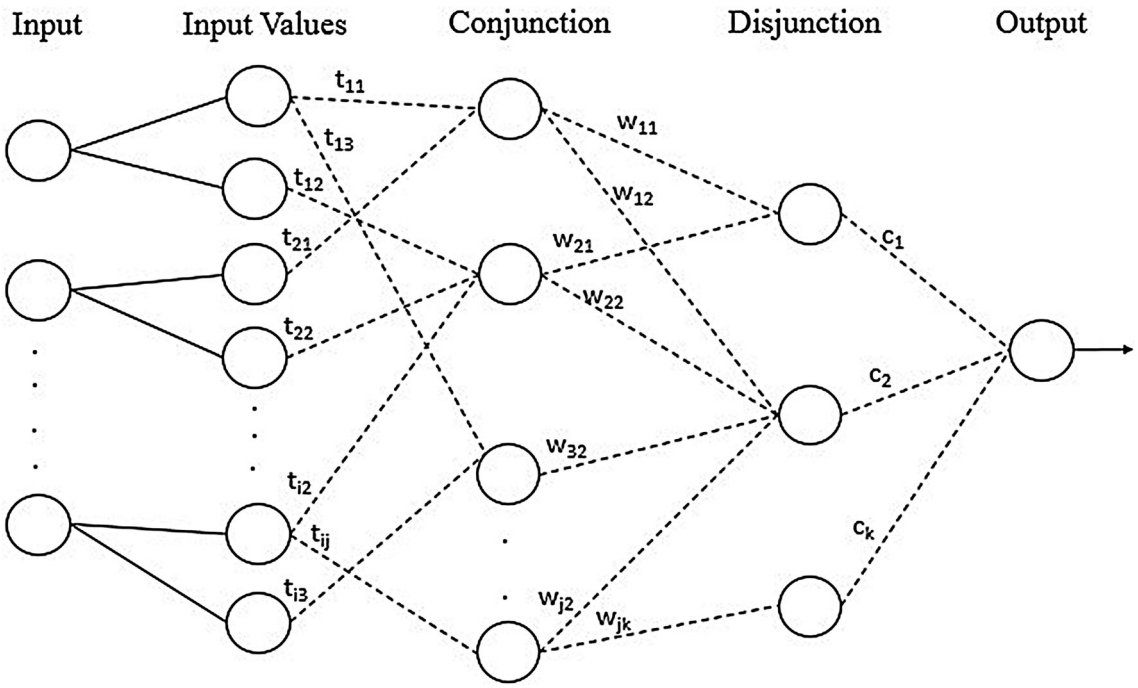

Fig. 8 Unified rule-based network (Liu et al. 2016a)

For each single rule
Calculate the J-measure for the rule which is referred to as the parent rule.
Compare the current J-measure with the highest one recorded so far
Update the highest value of the J-measure
For each of its child rules
Repeat the process recursively for calculating the J-measure
Choose the child rule with the highest J-measure. Special cases:
a) If two rules have the same J-measure, the one with the lower order is selected.
b) If two rules have the same J-measure as well as the order, randomly choose one.

Fig. 9 J-measure-based post-pruning inspired by ITRULE (Liu et al. 2016b)

probabilistic logic, fuzzy logic and rough logic. In the input layer, same as that illustrated in Fig. 7, each node represents an input attribute. In the input values layer, each node represents the value of a particular attribute. None of such input attributes can have its different values commonly appear in the same rule as rule terms. Therefore, none of the nodes in the input values layer can be commonly connected to the same node in the conjunction layer if the above nodes are connected to the same node in the input layer. In the disjunction layer, each node represents a class label used as a rule consequent. It is allowed in this topology to represent inconsistent rules, which indicates that it is acceptable for the same rule antecedent to be mapped to different consequents, each of which reflects a possible class assigned to unseen instances. For example, the first node in the conjunction layer has connections to the first two nodes in the disjunction layer as can be seen in Fig. 8. Each of these weights, which is assigned to the connection between two nodes, expresses a truth value while the computation is made by adopting deterministic or fuzzy logic. The type of a truth value is binary (0 or 1) if using deterministic logic whereas the type needs to be numerical (between 0 and 1) if using fuzzy logic. In addition, while the computation is made by adopting probabilistic logic, the weight assigned to the connection between two nodes would represent the probability. The final output from the only node in the output layer illustrated in Fig. 8 is probabilistic on the basis of the class probabilities originating from the nodes in the disjunction layer. Otherwise, the weight assigned to the connection between two nodes would represent a fuzzy truth value (fuzzy membership degree) for fuzzy logic-based computation, as mentioned above, or the possibility for rough logic-based computation. The final output for the computation based on fuzzy or rough logic is voted on the basis of the weights originating from the nodes in the disjunction layer. The importance of the rule-based network topology illustrated above is justified in Sect. 5.

5 Discussion

Section 4 argued the relationship between rule-based systems and granular computing and explores some important applications of granular computing concepts for advancing the development of rule-based systems. This section justifies critically why the applications of granular computing techniques illustrated in Sect. 4.3 are significant.

In terms of set theory, Sect. 4.3 presented that a family that consists of a parent rule and all its child rules can be managed through creation of a granular structure. Such management is important when a rule needs to be simplified using pruning algorithms. A specific pruning algorithm, which is based on J-measure and inspired by the ITRULE approach (Smyth and Rodney 1992), is introduced by Liu et al. (2016b) and illustrated in Fig. 9.

As illustrated in Fig. 9, in order to find the rule with the highest J-measure, it is required to check both the parent rule and all its child rules. In particular, it needs to calculate the J-measure for each of the above rules through checking the corresponding set that contains instances covered by the rule. In traditional ways, these rules, which are viewed as granules, are simply stored in a set. The search for the rule with the highest J-measure can only be done in a linear way, i.e., linear search. This is because all elements are stored in a set in an unordered way. In this case, the time complexity is O (2^n) where n is the number of rule terms (the order) for the parent rule. The detailed analysis of the time complexity is illustrated as follows: suppose that a parent rule has n rule terms. Then, the parent rule would have 2^n-2 child rules. This is similar to that a set that contains n elements would have 2^n-2 non-empty true subsets. On the basis of the above description, the computation time is proportional to 2^n-2 and thus exponential.

However, when set theory is adopted to construct a granular structure for management of the rules (including the parent rule and all its child rules) as well as their corresponding sets, the computational complexity will

decrease through parallelisation. For example, the granular structure has n levels, where n is the number of rule terms for the parent rule. In this context, the search of the rule with the highest J-measure can be done in the following procedure: firstly, to search through each level in parallel; then, to find a child rule with the highest J-measure for each level; finally, to find the target rule through comparing those child rules, each of which is found from a particular level in the granular structure. The application of set theory for reduction of the computational complexity is inspired by the theory of hierarchy as mentioned in Sect. 4.2. The theory of hierarchy is also useful when some particular child rules need to be retrieved for knowledge searching and reasoning. This is because a parent rule and all its child rules are already distributed in different levels of the granular structure according to their orders. The same also contributes to rule generation. For example, when searching for the best rule antecedents on the basis of their entropy values (Yao and Yao 2002) or other heuristics, the granular structure that is used to organize the full set of antecedents would usually lead to a more efficient search.

In addition, the construction of a granular structure mentioned above can also achieve global search of high quality rules leading to generation of a globally optimal set of rules that make up a rule-based system, in comparison with partition-based approaches of rule learning, which usually leads to generation of a locally optimal set of rules. In other words, the construction of a granular structure would provide the capacity towards deep learning of rule-based systems by means of search through rules that comprise any possible conjunctions of rule terms.

In terms of probabilistic logic, as mentioned in Sect. 4.3, it is supposed to achieve increase of accuracy for predictions by rules. In the context of deterministic logic, due to the nature of rule learning algorithms, all rules used are required to be consistent, which means that each of these rules cannot show mapping from the same input to different outputs. In machine learning research environments, the data used usually cover incomplete patterns. In other words, the data cannot represent a population but just a sample in the context of statistics. From this point of view, rules that are forced to be consistent are less confident. In addition, rule learning algorithms are typically based on statistical heuristics which may generally result in inductive bias and thus rules that make deterministic predictions are less confident. On the basis of the above description, when probabilistic logic is incorporated into rule learning algorithms, the bias would usually be reduced through probabilistic predictions, although this may cause some variances. In particular, each rule can be assigned different consequents with different levels of confidence. When a rule fires, the predicted output may be any one of the consequents. However, the consequent with a higher confidence would have a higher probability of being selected as the predicted output.

Similar advantages can also be achieved through incorporation of fuzzy or rough logic. In particular, when fuzzy logic is used, each rule can be assigned different consequents with different degrees of membership. In testing stage, each rule has a firing strength, which provides each consequent with a particular weight, and the maximum weight provided by a particular rule for each consequent is taken for weighted voting for classification tasks or weighted averaging for regression tasks towards the final prediction. When rough logic is used, each rule can only be assigned a single consequent. In testing stage, when the unseen instance contains no missing values, the rule is considered deterministic since there is no uncertainty for making the final prediction. If there are any missing values present in an unseen instance, the rule is considered rough towards making the final prediction since not all conditions can be judged satisfactory, i.e., some of the conditions are still pending. However, on the basis of the known values of the unseen instance, it is likely to have more than one rule partially firing, which means that some of the conditions are met as part of the left hand sides of these rules. In this case, each rough rule assigns the unseen instance a single consequent with a certain possibility, and the maximum possibility provided by a particular rule for each consequent is taken towards the final prediction, which is similar to the way used for fuzzy logic-based prediction.

In terms of the rule-based network topology illustrated in Sect. 4.3, the more important aspects are in model interpretability and computational complexity. As introduced in Liu et al. (2015a), rule-based systems can be used for two purposes: knowledge discovery and predictive modeling. For the former purpose, the model is required to be interpretable for people to read and understand. For the latter purpose, the model needs to demonstrate a high level of computational efficiency in making predictions on unseen instances. The rule-based network topology shows its significance with respect to both model interpretability and computational complexity. The significance analysis can be illustrated using Fig. 10. As this example is based on deterministic logic, in comparison with Fig. 8, the disjunction layer is turned into the output layer and the output layer illustrated in Fig. 8 is simply removed.

The above example of rule-based network is corresponding to the rule set as follows:

Rule 1: if $x1 = 0$ and $x2 = 0$ then $y = 0$;
Rule 2: if $x1 = 0$ and $x2 = 1$ then $y = 0$;
Rule 3: if $x1 = 1$ and $x2 = 0$ then $y = 0$;
Rule 4: if $x1 = 1$ and $x2 = 1$ then $y = 1$;

For the above example, when $x1 = 1$ and $x2 = 1$, the two connections between the node x_1 and the node v_{12} and between the node x_2 and the node $v_{22,}$ respectively, become

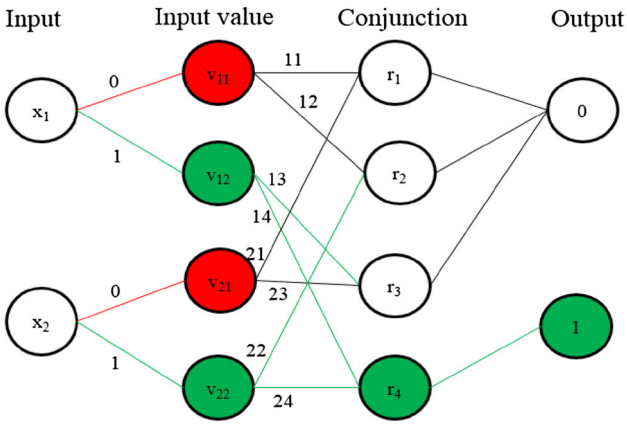

Fig. 10 Deterministic rule-based network example

green, which means that these two paths can be passed through. Then, there are four connections (13, 14, 22, and 24) between the nodes in the second and third layers becoming green as shown in Fig. 10. In the meantime, due to the interactions between the nodes in these two layers, the node r_4 is activated, which means that the corresponding rule (Rule 4) fires, and the output 1 is derived. In other words, the node r_4 can be viewed as an action listener, and will become green once it receives the signal that the two connections (14 and 24) have both become green. On the basis of the above description, the rule-based network illustrated in Fig. 10 demonstrates a divide and conquer search for the firing rules. Therefore, the computational complexity is O[log (n)], where n is the total number of rule terms in a rule set. As reported in Liu et al. (2015a), two other techniques of rule representation, i.e., decision tree and linear list, both demonstrate a search less efficient than the rule-based network. In particular, the computational complexity by linear list is O(n), where the n is the same as used in the rule-based network. In addition, the computational complexity by decision tree is O[log(n)], but the value of n is likely to be higher than the value corresponding to the other two representations as introduced in Liu et al. (2015a). This is due to the replicated subtree problem (Cendrowska 1987) by means of the presence of redundant rule terms.

With regard to model interpretability, as reported in Liu et al. (2015a), the rule-based network topology is capable of interpreting explicitly the following: correlation between attributes and classes, relationship between attributes and rules, ranking of attributes, ranking of rules and attribute relevance. As mentioned in Sect. 4.3, the rule-based network topology illustrated in Fig. 8 is a modified version of the one illustrated in Fig. 7. This version can also explicitly interpret the correlation between attribute values and classes as well as the relationship between attribute values and rules due to adding the layer of attribute values. In addition, ranking of attribute values can also be interpreted explicitly through looking at the newly added layer mentioned above. The comparisons with decision tree and linear list are illustrated in Table 2.

Overall, in comparison with traditional ways of learning rule-based systems, the applications of granular computing techniques described in this section bring the following new perspectives:

Firstly, construction of a granular structure through use of set theory can achieve deep learning of rule-based systems, through global search of high quality rules, and lead to the generation of a globally optimal set of rules that make up a rule-based system, in comparison with partition-based rule learning approaches, such as the divide and conquer and the separate and conquer approaches, which generally work through greedy search of rules based on statistical heuristics.

Secondly, when a rule needs to be simplified by means of finding one of its child rules with the optimal value of a statistical heuristic, it is necessary to achieve a more efficient search towards finding such a child rule. In this context, the construction of a granular structure mentioned above would lead to a more efficient search than traditional ways. In particular, as analyzed in this section, the traditional way of search can only be done linearly leading to an

Table 2 Comparison in interpretability among DT, LL, and RBN

Criteria	DT	LL	RBN
Correlation between attributes and classes	Poor	Implicit	Explicit
Relationship between attributes and rules	Implicit	Implicit	Explicit
Ranking of attributes	Poor	Poor	Explicit
Ranking of rules	Poor	Explicit	Explicit
Attribute relevance	Poor	Poor	Explicit
Correlation between attribute values and classes	Poor	Explicit	Explicit
Relationship between attributes values and rules	Explicit	Explicit	Explicit
Ranking of attribute values	Poor	Poor	Explicit
overall	Low	Medium	High

DT decision tree, *LL* linear list, *RBN* rule-based network

exponential time (2^n-2), where n is the number of rule terms for the parent rule. In contrast, the construction of a granular structure would provide the capacity for parallelized search through each set of the child rules with the same number of rule terms.

Thirdly, adoption of probabilistic, fuzzy and rough logic leads to more effective handling of uncertainty towards reduction of bias originated from algorithms. In particular, the probabilistic logic-based approach described earlier in this section is inspired by nature and biology. In other words, the final prediction towards a class assigned to a test instance is done through natural selection of one of all possible classes on the basis of their posterior probability given rule antecedents. Unlike Bayesian learning methods such as Naïve Bayes, the probabilistic approach is provided with the perspective that the class with the highest probability just has the best chance of being selected towards classifying an unseen instance. Similar advances towards reduction of bias can also be achieved through use of fuzzy and rough logic. For example, rough logic can be used to deal with missing values in a way different from traditional ways such as replacement with the most frequently occurring value for a discrete attribute or with average for a continuous attribute.

Finally, the use of the rule-based networks topologies, illustrated in Figs. 8 and 10, can lead to advances in knowledge discovery and predictive modeling in terms of model interpretability and computational efficiency, respectively, in comparison with existing rule representation techniques such as decision trees and linear lists as reported in Liu et al. (2015a). In this section, the rule-based network topologies are also provided with interpretation of characteristics in depth through use of granular computing concepts. These characteristics are listed in Table 2. As can be seen in Fig. 8, a rule-based network involves different types of objects, such as attributes, attribute values, rule antecedents, rule consequents and outputs. Each type of the objects is put in a particular layer with a number of nodes representing the specific type of objects. On the basis of the above descriptions, the rule-based network topologies represent characteristics relating to rules according to each specific type of objects and its relationships to other types.

6 Conclusions

This paper introduced the theoretical preliminaries of rule-based systems and granular computing as well as argued the relationships between the two areas in several contexts: the theory of hierarchy, computational intelligence, artificial intelligence, divide and conquer and the theory of small groups. This paper also explored in what way granular computing techniques can be effectively used for the design of rule-based systems. In particular, set theory can be effectively used for management of granules, each of which represents a parent rule or one of its child rules. These rules and their corresponding sets can be managed through constructing the granular structure mentioned in Sect. 4.1. In addition, probabilistic, fuzzy and rough logic can be used for handling uncertainty effectively through decrease of consistency but increase of accuracy for any generated rules. A rule-based network topology is also introduced towards effective management of model complexity and interpretability. The significance of using the above-mentioned granular computing techniques is also critically justified. In the future, probabilistic, fuzzy and rough logic will be investigated in depth for its incorporation into rule learning algorithms that follow the divide and conquer or the separate and conquer approaches. In this way, uncertainty would be effectively handled through reduction of bias originating from algorithms for rule generation and simplification. It is also strongly recommended to adopt set theory for the creation of a granular structure. This is in order to achieve effective management of granules that represent rules and child rules of these rules towards increase of effectiveness and reduction of computational complexity in the search of high quality rules.

References

Agrawal R, Imilielinski T, Swami A (1993) Mining association rules between sets of items in large databases. In: Proceedings of ACM SIGMOD International Conference on Management of Data, Washington D.C, p 207–216

Arrow H, McGrath JE, Berdahl JL (2000) Small groups as complex systems: formation, coordination, development. Sage Publications, Thousand Oaks

Avison D, Fitzgerald G (2006) Information systems development: methodologies, techniques and tools, 4th edn. McGraw-Hill Higher Education, London

Blachman NM (1968) The amount of information that y gives about X. IEEE Trans Inf Theory 14(1):27–31

Cendrowska J (1987) PRISM: an algorithm for inducing modular rules. Int J Man Mach Stud 27:349–370

Elomaa T, Kääriäinen M (2001) An analysis of reduced error pruning. J Artif Intell Res 15:163–187

Fürnkranz J (1999) Separate-and-conquer rule learning. Artif Intell Rev 13:3–54

Gegov A (2007) Complexity management in fuzzy systems, vol 211. Springer, Berlin

Gegov A (2010) Fuzzy networks for complex systems: a modular rule base approach. Springer, Berlin

Geng L, Hamilton HJ (2006) Interestingness measures for data mining: a survey. ACM Comput Surv 38(3):9

Hu H, Shi Z (2009) Machine Learning as Granular Computing. In: IEEE International Conference on Granular Computing, Nanchang, IEEE, China, p 229–234

Kononenko I, Kukar M (2007) Machine learning and data mining: introduction to principles and algorithms. Horwood Publishing Limited, Chichester

Liu H, Gegov A (2015) Collaborative Decision Making by Ensemble Rule Based Classification Systems. In: Pedrycz W, Chen S-M (eds), Granular computing and decision making: interactive and iterative approaches, Vol. 10. Springer, Switzerland, p 245–264

Liu H, Gegov A, Stahl F (2014) Categorization and Construction of Rule Based Systems. In: 15th Inter-national Conference on Engineering Applications of Neural Networks, Springer, Sofia, p 183–194

Liu H, Gegov A, Cocea M (2015) Network Based Rule Representation for Knowledge Discovery and Predictive Modelling. IEEE International Conference on Fuzzy Systems, IEEE, Istanbul, p 1–8

Liu H, Gegov A, Stahl F (2015) Unified Framework for Construction of Rule Based Classification Systems. In: Pedrycz W, Chen S-M (eds.) Inforamtion Granularity, Big Data and Computational Intelligence, Vol. 8. Springer, Berlin, p 209–230

Liu H, Cocea M, Gegov A (2016a) Interpretability of Computational Models for Sentiment Analysis. In: Pedrycz W, Chen S-M (eds) Sentiment analysis and ontology engineering: an environment of computational intelligence. Springer, Switzerland, pp 199–220

Liu H, Gegov A, Cocea M (2016b) Rule based systems for big data: a machine learning approach, vol 13. Springer, Switzerland

Pawlak Z (1982) Rough sets. Int J Comput Inform Sci 11(5):341–356

Pigott TD (2001) A review of methods for missing data. Edu Res Eval 7(4):353–383

Quinlan R (1986) Induction of decision trees. Mach Learn 1:81–106

Quinlan R (1993) C4.5: programs for machine learning. Morgan Kaufman

Ross TJ (2004) Fuzzy logic with engineering applications, 2nd edn. Wiley, West Sussex

Shannon C (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423

Smyth P, Rodney GM (1992) An information theoretic approach to rule induction from databases. IEEE Trans Knowl Data Eng 4(4):301–316

Tan P-N, Kumar V, Srivastava J (2004) Selecting the right objective measure for association analysis. Inf Syst 29(4):293–313

Yao Y (2004) Granular computing. In: Proceedings of the 4th chinese national conference on rough sets and soft computing, p 1–5

Yao Y (2005) Perspectives of granular computing. In: Proceedings of 2005 IEEE International Conference on Granular Computing. 1, IEEE, Beijing, p 85–90

Yao, Y. (2006). Granular Computing for Data Mining. In: Proceedings of SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, Kissimmee, p 1–12

Yao J, Yao Y (2002) Induction of Classification Rules by Granular Computing. In: Alpigini JF, Peters JF, Skowron A, Zhong N (eds) Rough sets and current trends in computing, Vol. 2475. Springer, Berlin, p 331–338

Zadeh LA (1979) Fuzzy sets and information granulation. In: Gupta MM, Ragade RK, Yager RR (eds) Advances in Fuzzy Set Theory and Applications, North-Holland Publishing Company, Amsterdam, p 3–18

Zadeh LA (1997) Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets Syst 90(2):111–127