Collaborative Decision Making by Ensemble Rule Based Classification Systems

Han Liu[1] and Alexander Gegov[1]

Abstract Rule based classification is a popular approach for decision making. It is also achievable that multiple rule based classifiers work together for group decision making by using ensemble learning approach. This kind of expert system is referred to as ensemble rule based classification system by means of a system of systems. In machine learning, an ensemble learning approach is usually adopted in order to improve overall predictive accuracy, which means to provide highly trusted decisions. This chapter introduces basic concepts of ensemble learning and reviews Random Prism to analyze its performance. This chapter also introduces an extended framework of ensemble learning, which is referred to as Collaborative and Competitive Random Decision Rules (CCRDR) and includes Information Entropy Based Rule Generation (IEBRG) and original Prism in addition to PrismTCS as base classifiers. This is in order to overcome the identified limitations of Random Prism. Each of the base classifiers mentioned above is also introduced with respects to its essence and applications. An experimental study is undertaken towards comparative validation between the CCRDR and Random Prism. Contributions and Ongoing and future works are also highlighted.

Keywords Data Mining · Machine Learning · Rule Based Classification · Ensemble Learning · Collaborative Decision Making · Random Prism

1 Introduction

Rule based classification is a common approach used for decision making. It is also feasible for multiple rule based classifiers to collaborate for group decision making by adopting ensemble learning approaches. This kind of expert system is referred

[1] Han Liu
University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, United Kingdom Email: Han.Liu@port.ac.uk
Alexander Gegov
University of Portsmouth, School of Computing, Buckingham Building, Lion Terrace, PO1 3HE Portsmouth, United Kingdom Email: Alexander.Gegov@port.ac.uk

to as ensemble rule based classification system by means of a system of systems. In this context, the ensemble rule based classification system is seen as a super system and consists of a number of single rule based classification systems, each of which is seen as a sub-system of the ensemble rule based classification system. In machine learning, an ensemble learning approach is usually adopted in order to improve overall predictive accuracy, which means to provide highly trusted decisions.

Ensemble learning can be done in parallel or sequentially. In the former way, there are no collaborations among different algorithms in training and only their predictions are combined for final decision making [1]. In this context, the final prediction is typically made by means of majority voting as part of the classification tasks. In the latter way of ensemble learning, the first algorithm learns a model from data and then the second algorithm learns to correct the former one etc [1]. In other words, the model built by the first algorithm is further corrected by the following algorithms sequentially. In parallel ensemble learning, a popular approach is to take sampling to a data set in order to get a set of samples. A classification algorithm is then used to train a classifier on each of these samples. The group of classifiers constructed will make predictions on test instances independently and final predictions on the test instances will be made based on majority voting. A commonly used sampling method is Bagging [2]. The Bagging method is useful especially when the base classifier is not stable due to high variance of data sample. This is because the method is robust and does not lead to overfitting as the number of generated hypothesizes is increased [1]. Some unstable classifiers include neural networks, decision trees and some other rule based methods [3].

In this chapter, all of the base classifiers used for ensemble learning tasks are rule based classification methods, namely original Prism [4], PrismTCS [5] and Information Entropy Based Rule Generation (IEBRG) [6]. All of the three methods follow 'separate and conquer' approach [7], which is one of the rule generation approaches. This is because each of the three methods generates if-then rules directly from training instances. The other approach of rule generation is referred to as 'divide and conquer' approach [8], which generates classification rules in the intermediate form of decision trees. As the generation aims to construct decision trees, the above approach is also referred to as Top-Down Induction of Decision Trees (TDIDT). A principal problem that usually arises with rule based classification methods is the overfitting of generated hypothesis to training data [9]. As mentioned earlier, the Bagging method is robust and helps avoid overfitting for rule based classifiers. It thus motivates the use of Bagging as a sampling method for ensemble learning tasks, especially when rule based methods are used as base classifiers.

The rest of this chapter is organized as follows. Section 2 introduces the three rule based classification methods, namely original Prism, PrismTCS and IEBRG. An existing ensemble learning method, called Random Prism [10, 11], is also introduced in the Section 2 in order to comparatively analyze the performance of the method. Section 3 introduces an extended framework of ensemble learning, which is referred to as Collaborative and Competitive Random Decision Rules (CCRDR) and includes the three base classifiers mentioned above. An experimental study is

undertaken towards comparative validation between the CCRDR and Random Prism in Section 4. The contributions and further directions of this research area are also highlighted in Section 5.

2 Related Work

As mentioned in Section 1, this chapter investigates parallel ensemble learning approaches which use Bagging as the sampling method and rule based methods as base classifiers. Therefore, this section introduces three rule based methods, namely original Prism, PrismTCS and IEBRG, the Bagging method and Random Prism.

2.1 Original Prism

The original Prism method was introduced by Cendrowska in [4] and the basic procedure of the underlying Prism algorithm is illustrated in Fig. 1. This algorithm is primarily aimed at avoiding the generation of complex rules with many redundant terms [9] such as the 'replicated subtree problem' [4] that arises with decision trees as illustrated in Fig. 2.

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

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

Fig. 1. Basic Prism algorithm [12]

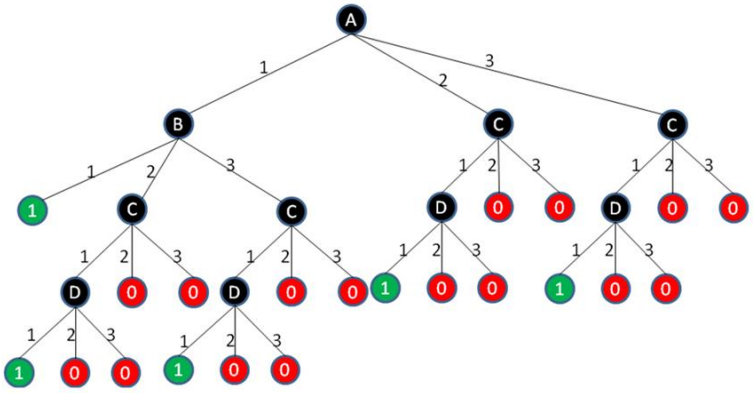

Fig. 2. Cendrowska's replicated sub-tree example [16, 17]

The original Prism algorithm cannot directly handle continuous attributes as it is based on the assumption that all attributes in a training set are discrete. When continuous attributes are actually present in a dataset, these attributes should be discretized by preprocessing the dataset prior to generating classification rules [12, 13, 14]. In addition, Bramer's Inducer Software handles continuous attributes as described in [12, 13, 14].

On the other hand, the original Prism algorithm does not take into account clashes, i.e. a set of instances in a subset of a training set that are identical apart from being assigned to different classes but cannot be separated further [12, 14]. Clashes usually occur in two principal ways:

1) One of the instances has at least one incorrect record for its attribute values or its classification [12].
2) The clash set has both (or all) instances correctly recorded but it is impossible to discriminate between them on the basis of the attributes recorded and thus it may be required to examine further attributes [12].

However, the Inducer software implementation [15] of Prism can handle clashes and the strategy of handling a clash is illustrated in Fig. 3. This way of dealing with clashes would result in underfitting of generated hypothesis to training data. This is because there would be a large number of instances that are not covered by the generated rule set if the rules that cover the instances are discarded. In testing stage, the way of clash handling would also make a large number of unseen instances left unclassified. This is because the algorithm does not generate a default rule that assigns a default classification (usually majority class) [15] to those instances that the generated rule set does not cover.

Another problem that arises with Prism is tie-breaking, i.e. if there are two or more attribute-value pairs which have equal highest probability in a subset (see step 3 in Fig.1). The original Prism algorithm makes an arbitrary choice in step 4 as

illustrated in Fig. 1 whereas the Inducer software makes the choice using the highest total target class frequency [12].

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

Fig. 3. Dealing with clashes in Prism [12, 16, 17]

Also, the original Prism may generate a rule set which may result in a classification conflict in predicting unseen instances. This can be illustrated by the example below:

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

Rule 2: If z=1 then class= b

What should the classification be for an instance with x=1, y=1 and z=1? One rule gives *class a*, the other one gives *class b*. A method is required to choose only one classification to classify the unseen instance [12]. Such a method is known as a conflict resolution strategy. Bramer mentioned in [12] that Prism uses the 'take the first rule that fires' strategy in dealing with the conflict problem and therefore it is required to generate the most important rules first. However, the original Prism cannot actually introduce an order to a rule according to its importance as each of those rules with a different target class is independent from each other. As mentioned in [5, 13, 14], this version of Prism would restore the training set to its original size after the completion of rule generation for class *i* and before the start for class *i+1*. This indicates the rule generation for each class may be done in parallel so the algorithm cannot directly rank the importance among rules with different target classes. Thus the 'take the first rule that fires' strategy may not deal with the classification confliction well.

2.2 PrismTCS

Bramer pointed out that the original Prism algorithm always deletes instances covered by those rules generated so far and then restores the training set to its original size after the completion of rule generation for class *i* and before the start for

class *i+1*. This results in a high number of iterations resulting in high computational cost [5] when the training data is very large. For the purpose of increasing the computational efficiency, a modified version of Prism, called PrismTCS, was developed by Bramer [5]. PrismTCS always chooses the minority class as the target class pre-assigned to a rule being generated as its consequence. Besides this, it does not reset the dataset to its original state and thus introduces an order to each rule according to its importance [5, 13, 14]. Therefore, PrismTCS is not only faster in generating rules compared with the original Prism, but also provides a similar level of classification accuracy [5, 13, 14].

As mentioned in Section 2.1, original Prism has some disadvantages in dealing with continuous attributes, tie-breaking, clashes and classification conflict. For each of these issues, Bramer introduces a corresponding solution in [12]. Each of the solutions is also applied to PrismTCS for each of the corresponding issues. In comparison to original Prism, PrismTCS can deal with conflict of classification better. This is because PrismTCS generates a set of ordered rules as mentioned earlier in this section. However, similar with original Prism, the way of dealing with clashes also results in underfitting to training data. As mentioned earlier, PrismTCS always chooses the minority class in the current training set as the target class of the rule being generated. Since the training set is never restored to its original size as mentioned above, it can be proven that one class could always be selected as target class until all instances of this class have been deleted from the training set because the instances of this minority class covered by the current rule generated should be removed prior to generating the next rule. This case may result in that the majority class in the training set may not be necessarily selected as target class to generate a list of rules until the termination of the whole generation process. In this case, there is not even a single rule having the majority class as its consequence (right hand side of this rule).

Although PrismTCS can generate a rule set which includes a default rule as introduced in [15] and thus leads to the decrease of number of unclassified instances, the default rule is likely to give a wrong classification to those unseen instances that are not covered by the generated rule set. This is because the assumption needs to be guaranteed that the training set covers complete patterns in a domain, which is in order to make the default rule unlikely to give wrong classifications. Otherwise, the rule set could still underfit the training set as the conditions of classifying instances to the other classes are probably not strong enough.

2.3 Information Entropy Based Rule Generation

IEBRG is developed in [6] in order to overcome the limitations of both original Prism and PrismTCS. This method is attribute-value-oriented like Prism but it uses the ‘from cause to effect’ approach. In other words, it does not have a target class pre-assigned to the rule being generated. The main difference with respect to Prism

is that IEBRG focuses mainly on minimizing the uncertainty for each rule being generated no matter what the target class is. A popular technique used to measure the uncertainty is information entropy introduced by Shannon in [18]. The basic idea of IEBRG is illustrated in Fig.4 as below:

1. Calculate the conditional entropy of each attribute-value pair in the current subset
2. Select the attribute-value pair with the smallest entropy to spilt on, i.e. remove all other instances that do not comprise the attribute-value pair.
3. Repeat step 1 and 2 until the current subset contains only instances of one class (the entropy of the resulting subset is zero).
4. Remove all instances covered by this rule.

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

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

Fig. 4. IEBRG algorithm

As mentioned in Section 2.1, all versions of Prism need to have a target class pre-assigned to the rule being generated. In addition, an attribute might be not relevant to each particular classification and sometimes only one value of an attribute is relevant [19]. Therefore, the Prism method chooses to pay more attention to the relationship between attribute-value pair and a particular class. However, the class to which the attribute-value pair is highly relevant is probably unknown, as can be seen from the example in Table 1 below with reference to the lens 24 dataset reconstructed by Bramer in [12]. This dataset shows that P (class=3|tears=1) =1 illustrated by the frequency table for attribute "tears". The best rule generated first would be *if tears=1 then class=3*.

Table 1. Lens 24 dataset example

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

This indicates that the attribute-value "tears=1" is only relevant to class 3. However, this is actually not known before the rule generation. According to PrismTCS strategy, the first rule being generated would select "class =1" as target class as it is the minority class (Frequency=4). Original Prism may select class 1 as well because it is in a smaller index. As described in [12], the first rule generated by Original

Prism is “if astig=2 and tears=2 and age=1 then class=1”. It indicates that the computational efficiency is slightly worse than expected and the resulting rule is more complex. When a large data set is used for training, the Prism method would be even likely to generate an incomplete rule covering a clash set as mentioned in Section 2.2 if the target class assigned is not a good fit to some of those attribute-value pairs in the current training set. Then the whole rule would be discarded resulting in underfitting and redundant computational effort.

In order to find a better strategy for reducing the computational cost, the IEBRG method is developed in [6]. In this method, the first iteration of the rule generation process for the “lens 24” dataset can make the resulting subset’s entropy reach 0. Thus the first rule generation is complete and its rule is represented by “if tears=1 then class=3”.

In dealing with continuous attributes, IEBRG takes the same way as applied to the Prism family, which includes original Prism and PrismTCS in the Inducer software implementation. With regard to tie-breaking, IEBRG deals with this issue in the way similar to that Prism family does, which means that when two or more attribute-value pairs have the same smallest entropy value the one with the highest total frequency is selected as introduced by Bramer in [12]. IEBRG can also deal with conflict of classification well because the method also generates a set of ordered rules like PrismTCS. In dealing with clashes, majority voting, which assigns the most common classification of the instances in the clash set to the current rule [12], is usually used for IEBRG, especially when the objective is to validate this method and to find its potential in improving accuracy and computational efficiency.

In comparison with the Prism family, this algorithm would reduce significantly the computational cost when the training set is large. In addition, in contrast to Prism, the IEBRG method deals with clashes by assigning a majority class in the clash set to the current rule. This would potentially reduce the underfiting of rule set thus reducing the number of unclassified instances although it may increase the number of misclassified instances. As mentioned in [12], Prism prefers to discard a rule rather than to give a wrong classification when a clash occurs and thus is more noise tolerant than TDIDT. However, if the reason that a clash occurs is not due to noise and the training set covers a large amount of data, then it would result in serious underfitting of the rule set by discarding rules as it would leave many unseen instances unclassified at prediction stage. The fact that Prism would decide to discard the rules in some cases is probably because it uses the so-called ‘from effect to cause’ approach. As mentioned in Section 2.1, each rule being generated should be pre-assigned a target class and then the conditions should be searched by adding terms (antecedents) until the adequacy conditions are met. Sometimes, it may not necessarily receive adequacy conditions even after all attributes have been examined. This indicates the current rule covers a clash set that contains instances of more than one class. If the target class is not the majority class, this indicates the search of causes is not successful so the algorithm decides to withdraw the task by discarding the incomplete rule and deleting all those instances that match the target

class in order to avoid the same case to happen all over again [13, 14]. This actually not only increases the irrelevant computation cost but also results in underfitting of the rule set. On the other hand, the IEBRG would also have the potential to avoid occurring clashes better compared with Prism. This is due to the strategy of rule generation from IEBRG as mentioned earlier in this section.

2.4 Bagging

As mentioned in Section 1, Bagging is a popular method of data sampling for ensemble learning tasks due to its robustness in avoiding overfitting. The term Bagging stands for bootstrap aggregating which is a method for sampling of data with replacement [1]. In detail, the Bagging method is to take a sample with a size as same as the original data set and to randomly select an instance from the original data set to be put into the sample set. This means that some instances in the original set may appear more than once in the sample set and some other instances may not even appear once in the sample set. According to the principle of statistics, the bagging method would produce a sample that is expected to contain 63.2% of the original data instances [1, 2, 10, 11]. Therefore, the Bagging method is useful especially when the base classifier is not stable due to high variance of data sample as mentioned in Section 1 and thus helpful to rule based classification methods in avoiding overfitting. For example, the method is successfully applied with PrismTCS into Random Prism for construction of ensemble learners [10, 11], which is further introduced in Section 2.5.

2.5 Random Prism

Random Prism, an existing ensemble learning method [10, 11], follows the parallel ensemble learning approach and uses Bagging for sampling as illustrated in Fig.5. It has been proven in [10, 11] that Random Prism is a noise-tolerant method alternative to Random Forests [20]. However, the Random Prism has two aspects in which can be improved in training and testing stages respectively. The above two aspects are also mentioned with suggestions for further improvements in [10, 11].

The first aspect is that there is only a single base classifier, PrismTCS, involved in training stage for Random Prism, which cannot always generate strong hypothesis (robust models). In fact, it is highly possible that a single algorithm performs well on some samples but poorly on the others. From this point of view, it is motivated to extend the ensemble learning framework by including multiple base classifiers involved in training stage. This is in order to achieve that on each data sample the learner created is much stronger.

On the other hand, Random Prism uses weighted majority voting to determine the final prediction on test instances. In other words, each model is assigned a weight, which is equal to the overall accuracy checked by validation data from the sample. In prediction stage, each model is used to predict unseen instances and give an individual classification. The ensemble learning system then makes the final classification based on weighted majority voting instead of traditional majority voting. For example, there are three base classifiers: A, B and C. A predicts the classification X with the weight 0.8 and both B and C predicts classification Y with the weights 0.55 and 0.2 respectively so the final classification is X if using weighted majority voting (weight for X: 0.8> 0.55+0.2=0.75) but is Y if using traditional majority voting (frequency for Y: 2>1). However, for the weighted majority voting mentioned above, the strategy in determining the weight is not reliable enough especially for unbalanced data sets. This is because it is highly possible that a classifier performs better on predicting positive instances but worse on negative instances if it is a two class classification task. The similar case can also happen in multi-class classification tasks. Therefore, it is more reasonable to use the individual accuracy for a single classification (e.g. true positive rate) as the weight.

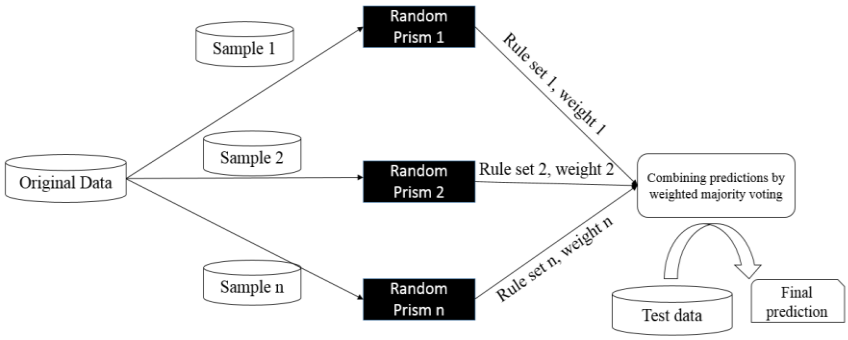

Fig. 5. Random Prism Framework with Bagging [10, 11].

Therefore, an extended framework of ensemble learning, referred to as Collaborative and Competitive Random Decision Rules (CCRDR), is developed in order to overcome the limitations and further introduced in Section 3.

3 Collaborative and Competitive Random Decision Rules

As mentioned in Section 2.5, Random Prism is a noise tolerant ensemble learning algorithm alternative to Random Forests [20]. However, it has two weak points in training and testing stages respectively and thus has space for improvement. This section introduces an advanced ensemble learning framework extended from Random Prism with the aim to overcome the two weak points which are mentioned above and described in Section 2.5. This section introduces a new framework that addresses the two weak points.

The framework developed in the authors' recent research is referred to as Collaborative and Competitive Random Decision Rules (CCRDR) and illustrated in Fig.6, which indicates that the ensemble learning framework includes both cooperative learning and competitive learning involved.

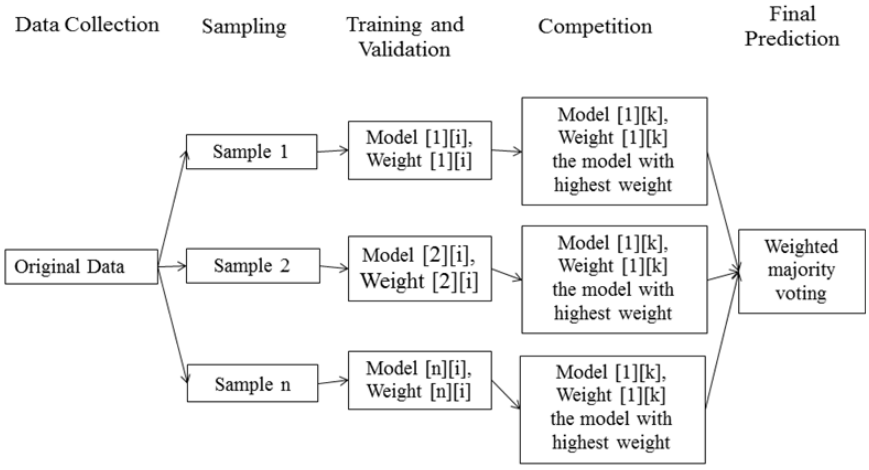

Fig. 6. Procedures of Proposed Ensemble Learning

The first weak point of Random Prism is that there is only a single base classifier involved in training stage, which cannot always generate robust models as mentioned in Section 2.5. In order to overcome the limitation, the ensemble learning framework is modified in the way that the framework can include multiple base classifiers for training. Due to this modification, there is competition involved among the classifiers constructed on a same sample of training data. In other words, there are multiple learning algorithms applied to each sample of training data, which implies that multiple classifiers are constructed on each sample. In this context, it becomes achievable to find better classifiers to be involved in testing stage and worse classifiers to be absent through competition among the classifiers. The competition is based upon the weight (confidence) of each of the classifiers by means

of overall accuracy measured by validation data. In the extended framework, only the classifier with the highest weight (confidence) is eligible to be involved in testing stage. The modification with regard to the first weak point is also reflected from the second part of the name of the method namely 'Competitive Random Decision Rules'. The name of the method indicates that any rule based classification methods are eligible for being involved in training stage as base classifiers. This modification theoretically contributes to that on each sample of data the learners constructed become much stronger.

The second weak point is regarding the way of determining the weight of a classifier for weighted majority voting as mentioned in Section 2.5. In order to overcome the limitation, confusion matrix, which reflects the individual accuracy for each single class such as true positive rate and true negative rate, is recommended in [10, 11]. However, the individual accuracy for a single classification reflected from confusion matrix is not effective in some special cases. In contrast, precision for a particular classification would be more reliable in determining the weight of a classifier. For example, there are 5 positive instances out of 20 in a test set and a classifier correctly predicts the 5 instances as positive but incorrectly predicts other 5 instances as positive as well. In this case, the recall/true positive rate is 100% as all of the five positive instances are correctly classified. However, the precision on positive class is only 50%. This is because the classifier predicts 10 instances as positive and only five of them are correct. This case indicates the possibility that high recall could result from coincidence due to low frequency of a particular classification. Therefore, precision is sometimes more reliable in determining the weight of a classifier on a particular prediction from this point of view. Overall, both precision and recall would usually be more reliable than overall accuracy in determining weight of a classifier especially for unbalanced data sets but it is important to determine which one of the two metrics to be used in resolving special issues.

The modifications to Random Prism with regard to its two weak points generally aim to improve the robustness of models built in training stage and to more accurately measure the confidence of each single model in making a particular prediction. In this chapter, original Prism, PrismTCS and IEBRG are used as base classifiers in the CCRDR framework due to the better noise tolerance of Prism family in comparison to TDIDT as well as the advantages of IEBRG listed in Section 2.3 in comparison to Prism family. However, in general, this framework could incorporate any type of rule based classification methods or even other type of machine learning methods such as Neural Networks [36] and Support Vector Machine [37]. With regard to the way to choose machine learning methods that are incorporated into the framework, it is typically based on theoretical analysis on the suitability of a particular method to a particular dataset. For example, some methods cannot directly deal with continuous attributes such as some rule based methods. In this case, it is required to discretize continuous attributes by preprocessing the dataset prior to training stage. One of popular approaches is ChiMerge [38]. There are also some methods that cannot effectively discrete attributes such as Neural Networks and Support

Vector Machine. In this case, it needs to split the discrete attributes into n binary attributes, while n is the number of values for the attribute, and each of the n binary attributes corresponds to a value of the original attribute. For example, gender is a discrete attribute with two values (male and female) and can be divided into two binary attributes named male and female respectively. Each of the binary attributes is to be judged either yes or no. If a dataset contains a large of discrete attributes and each of them has a large number of possible values, it would significantly increase the number of attributes for the dataset resulting in the curse of dimensionality [39]. On the basis of above description, one way to decide which methods are chosen for training could be based on the type of attributes as part of data characteristic. On the other hand, as mentioned in Section 2.4, the training instances are randomly selected from original dataset and different methods may demonstrate different level of robustness with respect to the change of sample. Therefore, the decision on choosing methods could also be based on the robustness of a particular method validated in experimental studies. Appropriate selection of algorithms would obviously help increase the overall performance of using the CCRDR framework with respects to both predictive accuracy and computational efficiency. The empirical validation of CCRDR framework against Random Prism is further introduced in Section 4.

The authors also define a novel way of understanding ensemble learning in the context of system theory by referring an ensemble classifier to as an ensemble rule based classification system. This is because an ensemble classifier actually consists of a number of single base classifiers as mentioned in Section 1. Therefore, in the context of system theory, an ensemble rule based classification system consists of a group of single rule based classification systems as mentioned in Section 1, each of which is a subsystem of the ensemble system. In other words, it is a system of systems like a set of sets in set theory. In addition, an ensemble rule based classification system can also be a subsystem of another ensemble system in theory. In other words, a super ensemble rule based classification system contains a number of clusters, each of which represents a subsystem that consists of a group of single rule based systems.

4 Comparative Validation

The validation of CCRDR framework against Random Prism is in terms of classification accuracy. The experimental study is undertaken by splitting a data set into a training set and a test set in the ratio of 80:20. For each data set, the experiment is repeated five times and the average of the corresponding accuracies is used for comparative validation. The reason is that ensemble learning is usually computationally more expensive because the size of data set dealt with by ensemble learning is as same as *n* times the size of the original data set when using Bagging. In other words, a data set should be pre-processed to get *n* samples, each of which has the same size

of original data set. In addition, the proposed ensemble learning method includes two or more base classifiers in general (three base classifiers in this experiment) used for each of the *n* samples. Therefore, in comparison to single learning such as use of IEBRG or Prism, the computational efforts would be the same as 3*n times that conducted by a single learning task. In this situation, the experimental environment would be computationally quite constrained on a single computer if cross validation is used to measure the accuracy. On the other hand, instances in each sample are randomly selected with replacement from the original data set. Thus the classification results are not deterministic and the experiment is setup in the way mentioned above to make the results more convincing. Besides, the accuracy performed by random guess is also calculated and compared with that performed by each chosen algorithm. This is in order to check whether a chosen algorithm really works on a particular data set as mentioned earlier. The validation of the proposed ensemble learning method does not include this measure of efficiency. This is because, on the basis of above descriptions, the computation conducted using the proposed method is theoretically much more complex if it is done on a single computer. However, the efficiency can be easily improved in practice by adopting parallel data processing techniques and is thus not a critical issue.

In addition, the comparison is also against the random classifier, which predicts classification by random guess. The corresponding accuracy depends on number of classifications and distribution of these classifications. For example, if the objective function is a two class classification problem and the distribution is 50:50, then the accuracy performed by random guess would be 50%. Otherwise, the accuracy must be higher than 50% in all other cases. This setup of experimental study is in order to indicate the lower bound of accuracy to judge if an algorithm really works on a particular data set.

All of the data sets used in this evaluation are retrieved from UCI repository [21], some of which contain missing values in input attributes or class attributes. This is usually a far large issue that needs to be dealt with effectively as it would result in infinite loops for rule based methods in training stage. In machine learning tasks, there are typically two ways of dealing with missing values [12]:

1) Replace all missing values by the most frequent occurring value for each attribute.
2) Discard all instances with missing values.

In this experimental study, the first way is adopted because all of the chosen data sets are relatively small. It indicates that if the second way is adopted both training and test sets would be too small to be representative samples. Under this kind of situation, the model generated is likely to introduce biased patterns with low confidence especially if the model overfits the training data. However, this way of dealing with missing values also potentially introduces noise to the data set. Thus such an experimental setup would also provide the validation with respect to the noise tolerance of an algorithm in the meantime. On the other hand, if missing values are

in the class attribute, then the best approach would be by adopting the second way mentioned above. This is because the first way mentioned above is likely to introduce noises to the data sets and thus incorrect patterns and predictive accuracies would be introduced. It is also mentioned in [12] that the first way is unlikely to prove successful in most cases and thus the second way would be the best approach in these cases. In practice, the two ways of dealing with missing values can easily be achieved by using the implementations in some popular machine learning software such as Weka [22, 23].

The validation is divided into two parts of comparison. The first part is to prove empirically that combination of multiple learning algorithms would usually outperforms a single algorithm as a base classier for ensemble learning with respect to accuracy. The second part is to prove that the use of precision instead of overall accuracy or recall as the weight of a classifier would be more reliable in making final prediction. In Table 2, CCRDR I represents that the weight of a classifier is determined by the overall accuracy of the classifier. In addition, the CCRDR II and III represent the weight determined using precision for the former and using recall for the latter.

Table 2. Ensemble learning results

Dataset	Random Prism	CCRDR I	CCRDR II	CCRDR III	Random classifier
anneal	71%	78%	79%	**80%**	60%
balance-scale	44%	56%	**68%**	64%	43%
diabetes	66%	68%	**73%**	68%	54%
heart-statlog	68%	71%	**74%**	63%	50%
ionosphere	65%	68%	**69%**	65%	54%
lympth	68%	60%	**89%**	65%	47%
car	69%	68%	**71%**	70%	33%
breast-cancer	70%	72%	**74%**	73%	58%
tic-tac-toe	63%	65%	66%	**67%**	55%
breast-w	**85%**	75%	81%	75%	55%
hepatitis	81%	84%	**87%**	82%	66%
heart-c	70%	74%	**83%**	65%	50%
lung-cancer	75%	79%	**88%**	75%	56%
vote	67%	82%	**95%**	80%	52%
page-blocks	**90%**	**90%**	**90%**	89%	80%

The results in Table 2 show that all of the chosen methods outperform the random classifier in classification accuracy. This indicates that all of the methods really work on the chosen data sets. In the comparison between Random Prism and CCRDR I, the results show that the latter method outperforms the former method in 12 out of 15 cases. This indicates empirically that combination of multiple learn-

ing algorithms usually helps generate a stronger hypothesis in making classifications. This is because the combination of multiple algorithms could achieve both collaboration and competition. The competition among these classifiers, each of which is built by one of the chosen algorithms, would make it achievable that for each sample of training data the learner constructed is much stronger. All of the stronger learners then effectively collaborate on making classifications so that the predictions would be more accurate.

As mentioned earlier, the second part of comparison is to validate that precision would usually be a more reliable measure than overall accuracy and recall for the weight of a classifier. The results in Table 2 indicate that in 12 out of 15 cases CCRDR II outperforms CCRDR I and III. This is because in prediction stage each individual classifier would first make classifications independently and their predictions are then combined in making a final classification. For the final prediction, each individual classifier's prediction would be assigned a weight to server for final weighted majority voting. The weight is actually used to reflect how reliable the individual classification is. The heuristic answer would be based on the historical record on how many times the classifier has recommended this classification and how correct it is. This could be effectively measured by precision. The weakness of overall accuracy is that this measure can only reflect the reliability of a classifier in average rather than in making a particular classification as mentioned in Section 2.5. Thus overall accuracy cannot satisfy this goal mentioned above. In addition, although recall can effectively reflect the reliability of a classifier in making a particular classification, the reliability is affected by the frequency of a particular classification and thus cheats the final decision maker, especially when the frequency of the classification is quite low as mentioned in Section 3. Therefore, the results prove empirically that precision would be more reliable in determining the weight of a classifier for weighted majority voting.

The basis of above description with regard to CCRDR validates that combination of multiple learning algorithms would be more effective in improving the overall accuracy of classification and that precision would be a more reliable measure in determining the weight of a classifier to successfully serve for weighted majority voting, especially on unbalanced data sets.

5 Conclusion and Future Work

This chapter reviews an existing ensemble learning method called Random Prism and three rule based classification methods, namely original Prism, PrismTCS and IEBRG. An extended framework of ensemble learning is developed, which is referred to as CCRDR and includes the three methods mentioned above as the base classifiers. The experimental study reports that CCRDR outperforms Random Prism in term of classification accuracy while the overall accuracy measured by validation data is used as the weight of a particular base classifier. In addition, the study also

reports that precision would usually be more reliable than recall and overall accuracy in measuring the confidence of a classification made by a classifier. However, all of the data sets used in the validation introduced in Section 4 are noise free and include well representative samples. Each method may have a particular level of noise tolerance and stability with regard to change of sample. Therefore, the authors will further check the tendency with respect to the change of level of predictive accuracy as the change of noise level. The authors will also further check the variance of the accuracy when the sample of training and test data is changed. These are in order to validate the noise tolerance and stability of the CCRDR against Random Prism.

On the other hand, as mentioned in Section 4, ensemble learning methods are usually computationally more expensive than single learning methods such as IEBRG and Prism family. This is because the size of data set dealt with by ensemble learning is as same as n times the size of the original data set. In the CCRDR framework, the size would be $m \times n$ times the size of the original data set, where m is the number of learning algorithms involved in training stage. However, as mentioned in Section 1, this type of ensemble learning tasks belongs to parallel ensemble learning, which indicates the tasks can be parallelized to improve the computational efficiency in both training and testing stages. In practice, each company or organization may have branches in different cities or countries so the databases for the companies or organizations are actually distributed over the world. As the existence of high performance cloud and mobile computing technologies, the ensemble learning framework can also be easily transplant into distributed or mobile computing environments such as multi-agent systems [29].

However, the theoretical framework introduced in this chapter still has space for extension. The ensemble learning concepts introduced in the chapter focus on parallel learning, which means that the building of each classifier is totally parallel to the others without collaborations in training stage and only their predictions in testing stage are combined for final decision making. However, the ensemble learning could also be done in sequential ways with collaborations in training stage. For example, there are two learning algorithms involved; the first one learns a model and the second one learns to correct the former as mentioned in Section 1. This is a direction to extend the theoretical framework further.

So far, ensemble learning concepts introduced in the machine learning literature lie in single learning tasks. In other words, all algorithms involved in ensemble learning need to achieve the same learning outcomes in different strategies. This is defined as local learning by the authors in the chapter. In this context, the further direction would be to extend the ensemble learning framework to achieve global learning by means of different learning outcomes. The different learning outcomes are actually not independent of each other but have interconnections. For example, the first learning outcome is a prerequisite for achieving the second learning outcome. This direction of extension is towards evolving machine learning approach in a universal vision. To fulfil this objective, the networked rule bases as illustrated

in Fig.7 can actually provide this kind of environment for discovering and resolving problems in a global way.

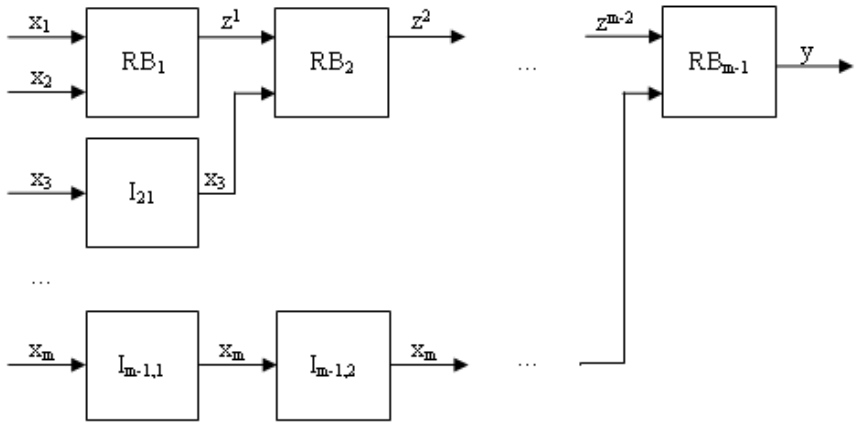

Fig. 7. Rule Based Network (modular rule bases) from [24, 25, 26, 27, 28]

In this network, each node represents a single rule base. The nodes can be connected sequentially or in parallel. In detail, each variable labelled x_{m-1}, while m represents the number of layer in which the node locates, represents an input and y represents the output. In addition, each of these labels labelled z^{m-2} represents an intermediate variable, which means this kind of variable is used as output for a former rule base and then again as inputs for a latter rule base as illustrated in Fig.7. On the other hand, there are two kinds of nodes representing rule bases as illustrated in Fig.7, one of which is a type of standard rule bases and labelled RB_{m-1}. This kind of nodes is used to transform the input(s) to output(s). The other type of nodes, in addition to the standard type, represents identities. It can be seen from the Fig.7 that this type of nodes does not make changes between inputs and outputs. This indicates the functionality of an identity is just like an email transmission, which means the inputs are exactly the same as outputs.

In practice, a complex problem could be subdivided into a number of smaller sub-problems. The sub-problems may need to be solved sequentially in some cases. They can also be solved in parallel in other cases. In connection to machine learning context, each sub-problem could be solved by using a machine learning approach. In other words, the solver to each particular sub-problem could be a single machine learner or an ensemble learner of which a single rule base can consist.

In military process modelling and simulation, each networked rule base can be seen as a chain of command (chained rule bases [24]) with radio transmissions (identities). In a large scale raid, there may be more than one chain of command.

From this point of view, the networked topology should have more than one networked rule bases parallel each other. All these networked rule bases should finally connect to a single rule base which represents the Centre of command.

The basis of above descriptions highlights the further directions of this research area. The extensions with respects to both sequential ensemble learning and networked rule bases would improve the intractability among different algorithms or models during the process of collaborative decision making. In addition, this also improves towards reduction of complexity in problem solving by dividing a complex problem into a set of simple problems. Therefore, the contributions would also be to complexity management [30], systems engineering [31, 32, 33, 34] and Big Data processing [35] in addition to machine learning.

References

1. Knoneko, I., Kukar, M.: Machine Learning and Data Mining: Introduction to Principles and Algorithms. Chichester: Horwood Publishing, 2007.
2. Breiman, L.: Bagging Predictors. Machine Learning, 2 (24), pp.123-140, 1996.
3. Tan, P.N., Steinbach, M., V. Kumar.: Introduction to Data Mining. New Jersey: Pearson Education, Inc, 2006.
4. Cendrowska, J.: PRISM: an algorithm for inducing modular rules, International Journal of Man-Machine Studies 27 (1987) 349–370.
5. Bramer, M.A.: Automatic induction of classification rules from examples using N-Prism, Research and Development in Intelligent Systems, vol. XVI, Springer-Verlag, Cambridge, 2000, pp. 99–121.
6. Liu, H., Gegov, A.: Induction of modular classification rules by Information Entropy Based Rule Generation. V. Sgurev, R. Yager, J. Kacprzyk (Eds.), 'Innovative Issues in Intelligent Systems', Springer. (In press)
7. Michalski, R.S.: On the Quasi-Minimal solution of the general covering problem, in: Proceedings of the Fifth International Symposium on Information Processing, Bled, Yugoslavia, pp. 125–128, 1969.
8. Quinlan, J.R.: C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.
9. Bramer, M.A.: Using J-Pruning to Reduce Overfitting of Classification Rules in Noisy Domains. Proceedings of 13th International Conference on Database and Expert Systems Applications— DEXA 2002, Aix-en-Provence, France, September 2–6, 2002.
10. Stahl, F., Bramer, M.A.: Random Prism: a noise-tolerant alternative to Random Forests. Expert Systems, Wiley Online Library, 2013.
11. Stahl, F., Bramer, M.A.: Random Prism: an alternative to Random Forests. *Research and Development in Intelligent Systems XXVIII*, 2011, pp 5-18, Springer.
12. Bramer, M.A.: Principles of Data Mining. London: Springer, 2007.
13. Stahl, F., Bramer, M.A.: Jmax-pruning: A facility for the information theoretic pruning of modular classification rules. Knowledge-Based Systems 29 (2012) 12-19.
14. Stahl, F., Bramer, M.A.: Induction of modular classification rules: using Jmax-pruning. In: In Thirtieth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, 14-16 December 2011, Cambridge.
15. Bramer, M.A.: Inducer: a public domain workbench for data mining. International Journal of Systems Science, 36(14):909–919, 2005.
16. Liu, H., Gegov, A., Stahl, F.: J-measure based hybrid pruning for complexity reduction in classification rules. WSEAS Transaction on Systems: 12(9), pp.433-446, 2013.

17. Liu, H., Gegov, A., Stahl, F.: Unified Framework for Construction of Rule Based Classification Systems. In: Pedrycz. W, Chen. S. M (Eds), Information Granularity, Big Data and Computational Intelligence, Studies in Big Data, Springer, pp.209-230, 2015
18. Shannon, C.: A mathematical theory of communication, Bell System Technical Journal, vol.27, no.3, (1948) 379-423. Fonn.
19. Deng, X.: A Covering-based Algorithm for Classification: PRISM. CS831: Knowledge Discover in Databases, 2012.
20. Breiman, L.: Random Forests. Machine Learning 45 (1): 5-32, 2001.
21. Bache, K., Lichman, M.: UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2013.
22. Hall, M., Frank, E., Holmes, Pfahringer, G., Reutemann, B. P., Witten, I. H.: The WEKA Data Mining Software: An Update; SIGKDD Explorations, 11(1), 2009.
23. Machine Learning Open Source Software. http://jmlr.org/mloss/. Accessed on 15th June 2014.
24. Gegov, A.: Fuzzy Networks for Complex Systems: A Modular Rule Base Approach. Berlin: Springer, 2010.
25. Gegov, A., Petrov, N., Vatchova, B.: Advanced Modelling of Complex Processes by Rule Based Networks, 5th IEEE International Conference on Intelligent Systems, 7-9 July 2010, London, pp. 197-202.
26. Gegov, A., Petrov, N., Vatchova, B., Sanders, D.: Advanced Modelling of Complex Processes by Fuzzy Networks, In: *WSEAS Transactions on Circuits and Systems* 10 (10), pp. 319-330, 2011.
27. Gegov, A.: Rule Based Networks: Theory and Applications, Plenary Lecture in IEEE International Conference on Intelligent Systems, London, UK, 2010.
28. Gegov, A.: Rule Based Network Models for Complex Systems, Tutorial in ENNS International Conference on Neural Networks, Sofia, Bulgaria, 2013.
29. Wooldridge, M.: *An Introduction to Multi-Agent Systems*. John Wiley & Sons, 2002.
30. Gegov, A.: *Complexity Management in Fuzzy Systems*. Berlin: Springer, 2007.
31. Schlager, J.: Systems engineering: key to modern development. *IRE Transactions* EM–3 (3): 64–66, 1956.
32. Sage, A. P.: *Systems Engineering*. Wiley IEEE, 1992.
33. Hall, A. D.: *A Methodology for Systems Engineering*. Van Nostrand Reinhold, 1962.
34. Goode, H. H., Machol, R. E.: *System Engineering: An Introduction to the Design of Large-scale Systems*. McGraw-Hill, 1957.
35. Gaber, M.M., Stahl, F., Gomes, J.B.: Pocket Data Mining, Big Data on Small Devices, Studies in Big Data, 2, Springer, 2014.
36. Stahl, F., Jordanov, I.: An Overview of Use of Neural Networks for Data Mining Tasks, WIREs: Data Mining and Knowledge Discovery, Wiley, 3 (2), pp. 193-208, 2012.
37. Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P.: "Section 16.5. Support Vector Machines". *Numerical Recipes: The Art of Scientific Computing* (3rd ed.). New York: Cambridge University Press, 2007.
38. Kerber, R.: ChiMerge: Discretization of Numeric Attributes. In Proceedings of the 10th National Conference on Artificial Intelligence. AAAI Press, pp. 123-128, 1992.
39. Houle, M. E., Kriegel, H. P., Kröger, P., Schubert, E., Zimek, A.: "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?". "Scientific and Statistical Database Management". Lecture Notes in Computer Science **6187**. p. 482, 2010.