Heuristic creation of deep rule ensemble through iterative expansion of feature space

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

[a] *School of Computer Science and Informatics, Cardiff University, Cardiff, United Kingdom*

[b] *Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan*

***Corresponding Author.**
E-mail addresses: liuh48@cardiff.ac.uk (H. Liu), smchen@mail.ntust.edu.tw (S.-M. Chen)

Abstract

Rule learning approaches, which essentially aim to gerenate a decision tree or a set of "if-then" rules, have been popularly used in practice for automatically building rule-based models for prediction tasks, e.g., classification and regression. The key strength of rule-based models is their ability to interpret how an output is obtained given an input, in comparison with models trained by other machine learning approaches, e.g., neural networks. Moreover, ensemble learning approaches have been adopted as a popular way for advancing the performance of rule-based prediction through producing multiple rule-based models with diversity. Traditional approaches of ensemble learning are typically designed to train a single ensemble. In recent years, there have been some studies on creation of multiple ensembles towards increasing the diversity among rule-based models and the depth of ensemble learning. In this paper, we propose a feature expansion driven approach for automatic creation of deep rule ensembles, i.e., the dimensionality of the feature space is increased at each iteration by adding features newly created at the previous iteration. The proposed approach is compared with more recent approaches of rule learning and ensemble creation. The experimental results show that the proposed approach achieves improved performance on various data sets.
***Keywords*:** Machine learning; Rule learning; Ensemble learning; Rule ensemble.

---

1. Introduction

In machine learning tasks, interpretability is one of important aspects that people highly expect in practice [3, 19], i.e., it is crucial to interpret how a predictive model built using a machine learning approach makes an output after being given an input. In this context, rule learning approaches are considered to have the strength in the model interpretation [3, 46], comparing with other popular learning approaches, e.g., neural

networks. Therefore, rule learning approaches have been popularly adopted to build rule-based models for knowledge discovery and predictions in some application domains, e.g., medicine [42, 46].

In traditional machine learning tasks, a rule learning approach is usually used to build a single rule-based model for classification or regression, through two typical learning strategies, namely, divide and conquer (DAC) [5] and separate and conquer (SAC) [16]. One of the main differences between the above two strategies is in terms of the model representation. In particular, rule-based models which are generated by taking the DAC strategy are automatically represented in the form of decision trees, whereas models would be represented in the form of a set of "if-then" rules if the SAC strategy is taken. In practice, rule-based models built through taking either one of the two strategies tend to overfit training data [26, 29], leading to the worse performance on unseen data. Therefore, several ways have been undertaken towards avoiding the case of overfitting, where one popular way is to build multiple models in the setting of ensemble learning [3].

Since the main aim of producing multiple models is to let the models collaborate each other, it would be highly important to ensure that there is some diversity among the models [50]. In general, there is not a commonly-accepted formal definition for the term "diversity" [24, 49], and thus people usually design some heuristic ways of diversity creation [50], e.g., diversification of features, samples, heuristic strategies of learning or hyper-parameters of the same learning approaches. Some specific ways of diversity creation that we design in this paper will be presented in Section 3.

Apart from the diversity creation, it is also important to ensure that each member of an ensemble needs to have as high performance as possible [50], which indicates the necessity to make the learning go deeper, i.e., increasing the depth of learning in addition to increasing the width of learning (through diversity creation). In this paper, we propose a deep rule ensemble creation approach that is driven by iterative expansion of the feature space used for learning classifiers. The key contributions of this paper are as follows:

(1) We propose a deep rule ensemble creation approach, which involves multiple iterations of learning and an automatic creation of new features at each iteration to increase the dimensionality of the feature space.

(2) The depth of learning is increased iteratively by adding new features, such that deeper ensembles are produced at the next iteration by learning from a richer set of features (i.e., the original features + the features created at the previous iterations of learning). In other words, while the original features are regarded as the prior knowledge, the features created at each iteration can be viewed as

newly learned knowledge and are added to the prior knowledge for further learning in more depth.

(3) In the creation of new features, multiple ways are taken at each iteration to produce diverse outputs, i.e., fusing the outputs of multiple ensembles created at the current iteration in different ways, such that the new features which are added at the next iteration contribute towards the production of deeper and more diverse ensembles.

(4) The experimental results indicate that the proposed approach performs considerably better than some recent methods [28, 31] of rule learning and ensemble creation on various data sets.

The remainder of this paper is organized as follows. Section 2 provides a review of the related work on the creation of ensembles of rule-based classifiers. In Section 3, we present the procedure of our proposed approach of deep rule ensemble creation in details, where some relevant preliminaries are also included. In Section 4, the details on conducting the experiments are provided and the experimental results are presented with discussions. In Section 5, the contributions of this paper are highlighted and some further directions are suggested.

2. Related work

Since rule learning can be operated in practice through two different strategies, namely, DAC and SAC, there are thus two main families of learning algorithms that aim to generate a decision tree and a set of "if-then" rules, respectively. In particular, the DAC strategy is designed to involve a recursive selection of an attribute A_x out of a set of candidate attributes in order to generate each non-leaf node of a decision tree, whereas the SAC strategy is designed to involve an iterative selection of an attribute-value pair (e.g., $A_x = v_{xb}$, where v_{xb} is the b-th value of attribute A_x) to generate a rule and repeatedly perform the same procedure for generating the next rule, until a complete set of "if-then" rules has been produced.

In general, the DAC strategy can be used to produce either binary trees or multi-way trees, depending on the type of the attribute selected for each node and how to handle a specific type of attributes. In other words, while all candidate attributes are of the binary type, the adoption of the DAC strategy would automatically result in the generation of a binary tree, which guarantees that each non-leaf node has two children. However, when an attribute A_x selected for generating a non-leaf node is multi-valued, the node may have n children (n is the number of possible values for A_x) by using some decision tree learning algorithms, such as ID3 [36] and C4.5 [37]. It is also possible to generate binary trees by learning from multi-valued attributes if the learning algorithm is designed to binarize multi-valued attributes. In particular, when a multi-valued

attribute A_x is selected for a non-leaf node N_{xy}, some algorithms, such as CART [8], are designed to generate two children of node N_{xy}, where one child results from taking the attribute-value pair $A_x = v_{xb}$ and the other child results from taking the opposite attribute-value pair $A_x \neq v_{xb}$.

In terms of generating a set of "if-then" rules, the SAC strategy can also be taken in different manners. In particular, one way is to select a target class *TC* and then iteratively select attribute-value pairs to become antecedents of a rule R_c, whereas another way is to learn a rule R_c by iteratively selecting attribute-value pairs to become antecedents of rule R_c, without the need to pre-select a target class *TC*. In other words, the former way is designed essentially to learn a rule that can effectively identify the instances of the target class *TC*, where a well-known example of such learning algorithms is so-called "Repeated Incremental Pruning to Produce Error Reduction" (RIPPER) [13]. In contrast, the essence of the latter way of taking the SAC strategy is to achieve that a rule R_c is learned to be capable of discriminating one class from the other classes, where a well-known example of such learning algorithms is so-called "CN2" [12].

For both the DAC and SAC strategies of rule learning, the algorithms are typically designed to induce rules heuristically from training data. The nature of heuristic learning of rules is likely to result in unstable performance due to the greedy search of attributes for generating decision trees or attribute-value pairs for generating "if-then" rules [3]. In other words, the attribute selected at each iteration for generating a non-leaf node of a decision tree is considered to be the locally (but not globally) best option, based on a specifically employed heuristic, e.g., Gini-index [39]. The same issue may also arise with learning algorithms that are designed for generating "if-then" rules [29]. Although there have been some more recent rule learning algorithms that involve heuristic modifications of classic algorithms towards the reduction of the bias on heuristic selection of attributes (or attribute-value pairs) through the greedy search, e.g., the PrismCTC algorithm [31] is a variant of the Prism algorithm [9], it is still unavoidable to have unstable performance on various data sets, i.e., a rule-based model may perform well on some data sets but the model may not generalize well on other data sets [2, 34]. In order to achieve better stability and higher generalization performance, it has become a popular way to train and fuse multiple rule-based models that are reliable and diverse, where the above way is known as ensemble learning [49].

The majority of the popular ensemble approaches, e.g., Bagging [6], Boosting [40] and Random Subspace [22], have been effectively used in the creation of decision forests (i.e., ensembles of decision trees). These ensemble approaches each involve specific ways of creating diversity among members of an ensemble (i.e., decision trees in the case of a decision forest). In particular, the diversity creation through Bagging or

Boosting is achieved essentially by drawing diverse samples of training data for building diverse decision trees. The Bagging approach is designed to draw each training sample S_i through Boostrap Sampling, i.e., each training sample S_i has the same size as the original training data D, where some instances (originally from D) may appear multiple times in S_i but some other instances (known as Out-of-Bag instances) may not appear in S_i at all. Since the production of n training samples $\{S_1, S_{2, ..., } S_n\}$ results in n different sets of Out-of-Bag instances, the n training samples $\{S_1, S_{2, ..., } S_n\}$ drawn from D are thus diverse leading to the production of a decision forest that consists of n diverse decision trees. The Bagging approach also has some variants such as Dagging [45] and Wagging [4].

In contrast to the Bagging approach, the Boosting approach involves assigning a weight to each training instance at each iteration i of drawing a training sample S_i from D, such that different instances have different chances for being selected into the training sample S_i. In particular, at the first iteration (i=1), all the training instances are given equal weights, i.e., they have the same chance for being selected into the training sample S_1. On this basis, the first decision tree DT_1 is trained on S_1, and some instances in D may be misclassified by DT_1, so these misclassified instances are given higher weights than the other instances, which indicates that the misclassified instances are more likely to be selected into S_2 at the next iteration (i=2). The second decision tree DT_2 is thus trained on S_2 and the same procedure for weighting of instances is repeated until the pre-defined number of iterations has been reached. Since n different sets of misclassified instances are normally obtained at n iterations, the n training samples $\{S_1, S_{2, ..., } S_n\}$ drawn from D are thus diverse, which makes it achieveable to produce a decision forest that consists of n diverse decision trees. A popular method of Boosting-driven creation of decision forests is known as Gradient Boosted Tree (GBT) [15].

Random Subspace essentially involves n independent iterations of random sampling of features to draw n feature subsets, which results in the possibility that decision trees trained on different feature subsets are diverse [22]. The Random Subsapce method has also been combined with the Bagging approach for creating decision forests, given the motivation to avoid the case that a forest involves many correlated trees produced from training samples drawn by the Bagging approach, due to the high likelihood of selecting some common features into many of these trees for generating non-leaf nodes, especially when these common features are strongly predictive of the target output (i.e., the class). The above combination of Bagging and Random Subspace has resulted in development of the so-called "Random Forest" method [7]. In creating random forests, the Random Subspace method is adopted at the node level instead of the model (tree) level, i.e., the Random Subspace method can generally be used at the tree level to produce n feature subsets $\{FS_1, FS_2, \ldots, FS_n\}$ on

which n decision trees DT_1, $DT_{2, ...,} DT_n$ are trained as parts of a decision forest, but for building a random forest, the production of a feature subset FS_{ia} is independently undertaken at the node level for generating each non-leaf node N_{ia} of a decision tree DT_i in the forest. The number of features selected in each feature subset FS_{ia} is generally treated as a hyper-parameter of the Random Forest method, but the suggested number of features in practice can be int($\log_2 M$+1) [7] or $\sqrt{M}$ [21], where M is the number of features in the original feature set FS. Overall, the design of the Random Forest method involves two aspects of diversity creation through learning of mutliple decision trees from various samples of training data and different feature subspaces.

There have also been some more recent methods of decision forest creation, such as Rotation Forest [38], Extremely Randomized Trees (ExtraTree) [18], Random Feature Weights for Decision Tree Ensemble Construction (RFW) [33], Forest by Continuously Excluding Root Node (FCERN) [1], Forest by Penalizing Attributes (FPA) [2], which have shown different ways of creating diversity on features used for generating each decision tree. In particular, the Rotation Forest method [38] is designed to employ the Principal Component Analysis (PCA) method [21] to combine features in the original feature set FS for drawing a new feature set FS_i at each iteration i of building a decision tree DT_i. The ExtraTree method [18] essentially involves the creation of randomness in dealing with continuous attributes, i.e., while a numeric value v_{xb} of a continuous attribute A_x needs to be selected as a threshold ($A_x \geq v_{xb}$ or $A_x < v_{xb}$) for spliting a training subset at a non-leaf node of a decision tree, the way designed in the ExtraTree method [18] is to make a fully random selection. The other three methods (i.e., the RFW method, the FCERN method and the FPA method) are all designed essentially to involve specific ways of assigning weights to features at each iteration i of training a decision tree DT_i, such that different features have different chances to be selected for generating non-leaf nodes of decision tree DT_i at each iteration i, i.e., the features which are selected at the n iterations for generating n decision trees are likely to be different.

In recent years, it has been emphasized that the learning needs to go deeper [50], which indicates the necessity to increase the depth of learning base classifiers in addition to the creation of the diversity among the base classifiers in the setting of ensemble learning. In particular, the so-called "Deep Forest" method has been developed in [50], which involves multi-grained scanning for feature representation learning from spatial data (e.g., images) or sequential data (e.g., text and signals) and then learning a deep forest by using a cascade forest architecture that involves L levels (i.e., multiple forests are produced at each level l and the deep forest model is gradually getting deeper by producing further forests at the subsequent levels l+1, l+2, ..., L), where the value of L is self-adaptive, i.e., the value of L is initialized to 0 and will be

continuously increased by 1 until the learning performance measured using validation data is not advanced any further.

Another approach, which is referred to as "Multi-Stage Mixed Rule Learning" (MSMRL), has been proposed in [28] for creating rule ensembles that are gradually getting deeper, through learning from general structured data (i.e., the features in the data do not have spatial or sequential relationships). The MSMRL approach is designed to have a pre-defined number of iterations towards granually increasing the depth of learning rule ensembles, and also to involve multiple ways of diversity creation through diversification of features and heuristics for learning different rule-based classifiers at each iteration, while the C4.5 algorithm [37] and the Mixed Fuzzy Rule Formation algorithm [17] are adopted in a collaboative manner for effectively dealing with data sets that contain both discrete and continuous attributes (features), i.e., the C4.5 algorithm and the Mixed Fuzzy Rule Formation algorithm involve different heuristic ways for dealing with continuous attributes, and the C4.5 algorithm can also effectively deal with discrete attributes that can not be handled directly by the Mixed Fuzzy Rule Formation algorithm. In the next section, we will show how our proposed approach works in a different way for achiving the gradual increase of the depth of learning rule ensembles from the general structured data, while multiple ways of diversity creation are also heuristically designed and incorporated in the proposed approach.

3. The proposed approach of deep rule ensemble creation

In this section, we propose an approach for deep rule ensemble creation in a step-by-step manner. The entire procedure of the proposed approach is shown in Fig. 1, Fig. 2 and Fig. 3, where Fig. 1 and Fig. 2 show how a deep rule ensemble is built in the training stage (i.e., the process of producing an ensemble committee *EC* that involves multiple levels and multiple classification models in each level); Fig. 3 shows how each test instance is classified by the deep rule ensemble produced in the training stage. The proposed approach essentially involves four main steps shown as follows:

Step 1: Build classification models at iteration i, where i is initialized to 1.

Step 2: Generate new features which are added into the feature set for possible use at the next iteration $i+1$.

Step 3: Determine whether the learning task continues, i.e., whether it is necessary to go for the next iteration $i+1$ of the learning task. If Yes, then go to **Step 1**. Otherwise, go to **Step 4**.

Step 4: Classify each of the test instances based on the procedure shown in Fig. 3.

Input: Data set *D*;
Output: Ensemble committee *EC*; /**EC* is a committee that consists of multiple levels, where multiple ensembles are involved in each level of the committee.*/

Let L = 0; /* Initialize the model depth L of EC (the number of levels that EC involves) */
Let $i = L+1$; /*Initialize the iteration index i */
Let max_Acc = 0; /* Initialize the maximum training accuracy max_Acc */
Initially select a set $M = \{M_1, M_2, \ldots, M_n\}$ of ensemble methods which are adopted at each iteration i;
Initially select a set $FR = \{FR_1, FR_2, \ldots, FR_q\}$ of fusion rules which are adopted at each iteration i;
/* Acc_i is the training accuracy measured using the data set D at iteration $i = L+1$ */
While L= 0 or $Acc_i > max\_Acc$ **Do**
    If $L > 0$ **Then**
      $max\_Acc = Acc_i$
    End If;
/*$|M|$ is the number of methods adopted for creating ensembles at iteration $i = L+1$ */
    For j = 1 to $|M|$ **Do**
        Create an ensemble E_{ij} on D using method M_j at iteration $i = L+1$; /* M_j is the jth ensemble method that is initially selected into a set $M = \{M_1, M_2, \ldots, M_n\}$ of ensemble methods */
        Add E_{ij} into an ensemble set ES_i built at iteration $i = L+1$;
    End For
      Add ES_i into ensemble committee EC;
      Increase the model depth L of EC by letting $L = L+1$;
/* $|FR|$ is the number of fusion rules adopted for generating class vectors at iteration $i= L+1$ */
    For k =1 to $|FR|$ **Do**
        For each training instance e **Do**
            Generate a class vector by letting Vec_{ik} = **Generate_Class_Vector** (D, e, i, k ,M, FR_k), where the return value of **Generate_Class_Vector** (D, e, i, k, M, FR_k) is obtained based on Fig. 2
            Update the feature set FS of the data set D by adding each dimension of Vec_{ik} as a new feature of e;
        End For
    End For;
    Let Acc_i = **Learning_Performance_Validation** (D, M), where the return value of **Learning_Performance_Validation** (D, M) is obtained through a K-fold cross validation or a hold-out validation;
End While.
Obtain the ensemble committee $EC = \{ES_1, ES_2, \ldots, ES_L\}$ by collecting each ensemble set ES_i produced at a specific iteration i, where i = 1, 2, …, L.

Fig. 1. The training process of the proposed deep rule ensemble creation approach.

The proposed approach of deep rule ensemble creation is described as follows:

(1) At **Step 1** of the proposed approach of deep rule ensemble creation, there are totally n classification models built using n learning methods at each iteration i. In general, a classification model can be in one of multiple forms, i.e., it can be an individual classifier, an ensemble of individual classifiers or an ensemble of ensembles,

depending on the nature of the chosen learning methods. In this paper, we aim at building the second and third forms of classification models at each iteration i. In particular, four models are built at each iteration i. The first model is built by adopting the Random Forest method [7], where multiple random trees are produced as parts of the model. The second model is built by adopting the FPA method [2], which consists of multiple decision trees produced heuristically by the CART algorithm [8]. The third and fourth models are both built by jointly adopting the Bagging approach [6] and the Random Subspace method [22]. In other words, the Bagging approach is adopted to draw g samples from the original training data, and then h feature subsets are drawn randomly on each training sample, which will need to have $g \cdot h$ base classifiers produced in total. The C4.5 algorithm [37] is used for producing the base classifiers which form the third model, whereas the base classifiers that form the fourth model are produced by the RIPPER algorithm [13]. The above-mentioned four models are produced in different settings of ensemble creation in order to achieve the heuristic creation of the diversity. In particular, the first and second models are essentially two ensembles of individual classifiers (i.e., two ensembles of decision trees), whereas the third and fourth models are essentially two ensembles of ensembles (i.e., two ensembles of forests). Moreover, four different algorithms are employed to train individual rule-based classifiers, respectively, for producing the four models. Moreover, the Random Forest method [7] involves adopting the Random Subspace method [22] when generating each non-leaf node of a decision tree, in order to achieve feature diversification (i.e., different features are used for generating different trees). In contrast, the FPA method [2] involves achieving the feature diversification in a different way, i.e., to assign different weights to each specific feature when generating different trees, such that each feature may have a higher chance to be selected for generating some trees but will have a lower chance to be selected for generating other trees. In this way, the two forests produced by the Random Forest method [7] and the FPA method [2], respectively, are likely to be diverse, which means that the two ensembles of trees are produced using diverse subsets of features. For producing the third and fourth models, the way of adopting the Random Subspace method [22] is heuristically different from the way taken for building the first model (i.e., the random forest). In other words, in order to produce the third and fourth models, the production of feature subsets by the Random Subspace method [22] only needs to be undertaken at the beginning through the feature subsampling from the full feature set of a training sample. In this context, once a feature subset is drawn from the full feature set, a rule-based classifier is trained entirely on the drawn feature subset, without the need to repeat the feature subset selection for generating each part of a rule-based classifier (e.g., a non-leaf node of a decision tree). Therefore, involving

different ways of adopting the Random Subspace method [22] also increases the effectiveness of the feature diversification. Besides, the adoption of the Bagging approach [6] is involved in building each of the above-mentioned four models (i.e., each of the four ensembles), which essentially aims to achieve a heuristic creation of the diversity through the diversification of samples in addition to the diversification of features.

Input: Data set D, instance e, iteration i, a set of ensemble methods $M = \{M_1, M_2, \ldots, M_n\}$, fusion rule FR_k, and index k of FR_k;
Output: Class vector Vec_{ik}; /* Vec_{ik} is a p-dimensional vector, which represents p probability values for p classes involved in the data set D*/
Initialize the number K of folds for K-fold cross validation (i.e., K is a hyper-parameter);
Initialize Vec_{ik} to a zero vector (e.g., Let $Vec_{ik} = (0, 0, 0)$ if the data set D involves three classes);
Initially create an empty vector set VS for storing a class vector Vec'_{ikj} generated at each fold f; where j is the index of the ensemble method used for generating the class vector Vec'_{ikj};
For each fold $f = 1$ to K **Do**
  Get the training set TS = **Get_Training_Set** (D, f), where **Get_Training_Set** (D, f) is obtained by taking K-1 folds of the data set D, i.e., excluding fold f from the data set D;
  For $j = 1$ to $|M|$ **Do**
    Build an ensemble E'_{ij} on TS;
    If $e \in TS$ **Then**
      Classify training instance e using E'_{ij} to generate a class vector Vec'_{ikj};
    End If;
    Add Vec'_{ikj} into a vector set VS
  End For;
  If $e \in TS$ **Then**
    Generate a class vector Vec'_{ik} by combining the class vectors in VS using fusion rule FR_k;
    Let $Vec_{ik} = Vec_{ik} + Vec'_{ik}$
  End If
End For;
Let $Vec_{ik} = \frac{1}{K-1} \cdot Vec_{ik}$.

Fig. 2. Procedure of class vector generation for each training instance.

(2) At **Step 2** of the proposed approach of deep rule ensemble creation, the aim is to update the feature set FS by adding new features created at iteration i. In particular, the new features are essentially represented by numeric values of class probability. In other words, for each training instance e, a class vector can be generated after the training instance has passed through a classification model. In this case, the

generated class vector has p dimensions representing the posterior-probability values for p possible classes. When there are n classification models deployed for classifying the training instance e, n class vectors will be generated. In order to obtain the finalized class vectors contributing to the new features added into the feature set FS, a further operation needs to be undertaken, i.e., to combine the above-mentioned n class vectors by using a fusion rule FR_k. As a result, while q fusion rules are adopted for combining the above-mentioned n class vectors, there will be q class vectors generated in total and the number of new features added into FS is $p \cdot q$. In practice, as illustrated in Fig. 2, the generation of a class vector Vec_{ik} which represents p new features of a training instance e can be operated in the following way: a K-fold cross validation is undertaken on the training data set, such that each training instance e will be used K-1 times for building K-1 models and one time for validation of another model. In this context, there will be K-1 class vectors generated for each training instance and the averaging of the K-1 class vectors will produce the finalized class vector Vec_{ik} for the instance. The above way of class vector generation for each training instance is recommended in [50] to avoid overfitting in the training stage, i.e., it may result in the risk of overfitting if the entire training data set is used for building a classification model to classify training instance e to generate a class vector Vec_{ik}. In general, the dimensionality of the feature space (i.e., the number of features in a feature set FS_i) is increased iteratively with the increase of the learning depth L. The general relationship between the dimensionality $|FS_i|$ of the feature space updated at the end of each iteration i and the learning depth L can be formulated as follows:

$$|FS_i| = |FS| + L \cdot p \cdot q, \tag{1}$$

where $|FS|$ denotes the number of features in the original feature set FS used at the iteration i=1. Moreover, because the generation of a class vector aims to obtain a numeric feature representation by probability values of classes, the chosen fusion rules need to work in an algebraic manner using some popular rules of algebraic fusion, including the "mean" rule (Eq. (2)), the "median" rule (Eq. (3)), the "minimum" rule (Eq. (4)), the "maximum" rule (Eq. (5)) and the "product" rule (Eq. (6)), shown as follows:

$$P^{FR_k}(class = C_t|FV_i^e) = \frac{1}{n}\sum_{j=1}^{n} P^{E_{ij}}(class = C_t|FV_i^e), \tag{2}$$

$$P^{FR_k}(class = C_t|FV_i^e) = \underset{1<j<n}{\text{med}}\{P^{E_{ij}}(class = C_t|FV_i^e)\}, \tag{3}$$

$$P^{FR_k}(class = C_t|FV_i^e) = \min_{1<j<n}\{P^{E_{ij}}(class = C_t|FV_i^e)\}, \tag{4}$$

$$P^{FR_k}(class = C_t|FV_i^e) = \max_{1<j<n}\{P^{E_{ij}}(class = C_t|FV_i^e)\}, \quad (5)$$

$$P^{FR_k}(class = C_t|FV_i^e) = \prod_{j=1}^{n} P^{E_{ij}}(class = C_t|FV_i^e). \quad (6)$$

More details of these fusion rules can be found in [23, 49]. In Eqs. (2)-(6), "$P^{E_{ij}}(class = C_t|FV_i^e)$" denotes the posterior probability of *class* = C_t given the feature vector FV_i^e which represents training instance *e* at iteration *i*, which is estimated by adopting ensemble E_{ij}. "$P^{FR_k}(class = C_t|FV_i^e)$" denotes the posterior probability of *class* = C_t given the same feature representation FV_i^e of training instance *e* at iteration *i*, which is obtained by combining *n* posterior probability values estimated by *n* ensembles $\{E_{i1}, E_{i2}, \ldots, E_{in}\}$. For the above rules of algebraic fusion, the "minimum" rule is equivalent to the "maximum" rule for two-class classification tasks [23]. Moreover, the "product" rule may result in the veto mechanism problem [44] in practice, i.e., it is possible to occur that the probability values of all the classes are zero. Therefore, only the three fusion rules, namely, the "maximum" rule, the "mean" rule and the "median" rule, are adopted for generating class vectors in the setting of the proposed approach, in order to avoid generating zero-vectors by the "product" rule or generating identical vectors by the "minimum" rule and the "maximum" rule. The aim of adopting multiple fusion rules to generate multiple class vectors is to create more diverse features. In other words, those different class vectors can be viewed as features from different views. In particular, different fusion rules work in different ways to combine the class probability values estimated by different members of an ensemble [23], which is likely to lead to different impacts on the difference between the combined probability value and the true probability value for each class. Moreover, each fusion rule can have different impacts on different training instances [44], i.e., the use of the same fusion rule may result in an increase of the chance of correctly classifying some training instances but the chance of correctly classifying other training instances may be decreased using the same fusion rule. From this point of view, the feature representation of some training instances may be better improved by adding class vectors generated using one fusion rule (say FR_1), in comparison with another fusion rule (say FR_2), whereas the use of FR_2 may lead to a better improvement of the feature representation of other training instances, in comparison with the use of FR_1. Because rule learning methods [8, 37] essentially involve self-selection of features for generating rule-based classifiers, the increase of the feature space dimensionality is likely to provide more chance of achieving a better selection of features during the learning process. While the Bagging approach [6] is used to draw diverse training samples and multiple ways of feature diversification are

involved, adding multiple class vectors which are produced by using different fusion rules is thus considered an effective strategy of achieving a further creation of diversity in addition to the increase of the learning depth.

(3) At **Step 3** of the proposed approach of deep rule ensemble creation, it is essential to determine whether it is necessary to continue the learning task by going for the next iteration i+1, i.e., whether or not the depth of learning is increased towards producing further models on the basis of the updated feature set obtained at the end of iteration i. In general, the above judgement depends on whether the learning performance measured using the validation data can be advanced further as inspired from [50], i.e., the learning task will be terminated automatically if the increase of the learning depth L does not gain any further advances in the learning performance. In practice, the measure of learning performance can be achieved by conducting a K-fold cross validation [21] or a hold-out validation [21]. However, a K-fold cross validation would be generally recommended, especially when the sample size of the training data is not sufficiently large for taking a part of the training data as an independent validation set. In the validation stage, each validation instance is classified through a two-level fusion operation that we design heuristically as part of the proposed approach. In particular, at the first level, each validation instance would first be classified by each ensemble E_{ij} (i.e., a part of ensemble set ES_i) produced at iteration i, so there would be n vectors of continuous-valued outputs obtained by using the n ensembles in ES_i. Each of the n vectors of continuous-valued outputs essentially involves p probability values for p possible classes. In this case, for each class C_t, the n probability values which are produced by n ensembles are combined by using a fusion rule FR_k. As a result, while q fusion rules are adopted for combining the probability values, there would be q new vectors of continuous-valued outputs obtained after the fusion operations. Furthermore, at the second level, the q new vectors of continuous-valued outputs are fused further using the "mean" rule, which is the most commonly used one in practice [25], i.e., for each class C_t, the q values of the posterior probability are averaged to obtain the finalized posterior probability value, where the p finalized probability values for the p possible classes are used together for finally classifying a validation instance, i.e., the validation instance is classified to the class that obtains the highest probability value.

(4) At **Step 4** of the proposed approach of deep rule ensemble creation, each test instance is classified through a level-by-level processing manner. As illustrated in Fig. 3, a test instance u is classified to a class label C_{tl} at level l of an ensemble committee EC by using an ensemble set ES_l (i.e., an ensemble of ensembles in EC) produced at iteration i of the training stage, where $i = l$. In the meantime, a weight w_l is assigned to class label C_{tl}, which indicates the confidence of ES_l outputting C_{tl}

as the class label. The value of the weight w_l is obtained in our setting by measuring the classification accuracy of the ensemble set ES_l on the validation data. Before moving onto the next level l+1 of *EC*, the feature representation of the test instance u needs to be updated by adding q generated class vectors into the feature set FS_{i+1}. In particular, the q class vectors are essentially obtained following a two-stage operation in our setting of the proposed approach. At first, the test instance u passes through the n ensembles in ES_l, which results in n class vectors being generated. Then, the n class vectors are combined using q rules of algebraic fusion, respectively, leading to q new class vectors being obtained. The q new class vectors represent $p \cdot q$ values of class probability obtained for the test instance u (i.e., p is the number of classes), which are added into the feature set FS_{i+1} as new features used at the next iteration i+1. The above procedure is repeatedly performed until the ensemble set ES_L at the last level (i.e., $l = L$) has been used to classify the test instance. At this point, the test instance will have received L class outputs from the L ensemble sets, where each of the L ensemble sets is involved in a specific one of the L levels of the ensemble committee *EC* (i.e., the entire model built on the training data). In this context, a final classification needs to be made as the output of *EC*, which is operated by the weighted voting on the basis of the above-mentioned L class outputs, shown as follows [34]:

$$\mathrm{Vote}(C_t) = \sum_{l=1, l \neq z}^{L} w_l, \tag{7}$$

where $\exists z \in [1, L]$: $h_{ES_z}(u) \neq C_t$, i.e., for some values of z between 1 and L, ES_z will classify a test instance u to another class rather than class C_t, "$\mathrm{Vote}(C_t)$" represents the total (weighted) vote obtained for class C_t, w_l is the weight of ensemble set ES_l involved at level l of Ensemble Committee *EC*, and u denotes a test instance that needs to be classified by using *EC*,

$$C_o = \arg\max_{C_t} \mathrm{Vote}(C_t), \tag{8}$$

where C_o is the final output of Ensemble Committee *EC*, which is made by choosing the class C_t which obtains the highest total weighted vote "$\mathrm{Vote}(C_t)$" (Note: if there are multiple classes that obtain the highest total weighted vote, the class with the smallest index value will be chosen as the final output C_o, e.g., while the two classes C_1 and C_2 both obtain the highest total weighted vote, class C_1 will be chosen due to its smaller index value).

Input: Ensemble committee EC, test instance u; /* EC involves L levels, where an ensemble set ES_l is involved in each level l */

Output: Class label C_o; /* C_o is the class to which test instance u is classified by using EC*/

Let l =1; /* *Initialize* the level l of ensemble committee EC to 1 */

/* L is the number of levels involved in ensemble committee EC, where the value of L is determined automatically in the training stage for the creation of EC */

For l =1 to L **Do**

Classify test instance u to a class label C_{tl} by using ensemble set ES_l;

/* w_l is the accuracy of ES_l measured in the training stage using the validation data */

Assign a weight w_l to C_{tl};

/* $|FR|$ is the number of fusion rules adopted for generating class vectors at level l */

For k=1 to $|FR|$ **Do**

Generate a class vector Vec_{lk} by using fusion rule FR_k to combine the n class vectors (Vec_{lk1}, Vec_{lk2}, …, Vec_{lkn}) generated by n ensembles in ensemble set ES_l;

Update the feature representation of test instance u by adding each dimension of Vec_{lk} as a new feature of u;

End For

End For;

Determine the final output C_o of the ensemble committee by choosing the class C_t which obtains the highest total weighted vote, i.e., the selection of the class C_t is achieved by the weighted voting on the basis of the L class outputs $\{C_{t1}, C_{t2}, \ldots, C_{tL}\}$ predicted by L ensemble sets in EC.

Fig. 3. Procedure for classifying each test instance in the testing stage.

4. Experimental results

In this section, 20 data sets adopted from the UCI repository [27] are used for conducting the experiments. The details of the 20 data sets are described in Table 1. The characteristics of the data sets are diverse, e.g., some data sets contain both discrete and continuous attributes, where the other data sets contain only one type of attributes (i.e., either discrete attributes or continuous attributes). Moreover, some data sets aim for binary classification tasks, whereas the other data sets aim for multi-class classification tasks.

Table 1
Data sets used for experiments.

Data sets	Number of discrete/continuous attributes	Number of instances	Number of classes

Anneal	32/6	898	6
Balance-scale	0/4	625	3
Breast-cancer	9/0	286	2
Breast-w	0/9	699	2
Credit-a	9/6	690	2
Credit-g	13/7	1000	2
Cylinder-bands	21/18	540	2
Dermatology	33/1	366	6
Diabetes	0/8	768	2
Hepatitis	13/6	155	2
Ionosphere	0/34	351	2
Iris	0/4	150	3
Kr-vs-kp	36/0	3196	2
Labor	8/8	57	2
Lymph	15/3	148	4
Sponge	45/0	76	3
Tae	2/3	151	3
Vote	16/0	435	2
Wine	0/13	178	3
Zoo	16/1	101	7

In the experimental setting of the proposed approach, at each iteration i of building the ensemble committee EC (i.e., a deep rule ensemble architecture that consists of multiple levels), there are four classification models built as parts of ensemble committee EC by setting four ensemble creation methods. The first classification model is built using the Random Forest method [7], where the built forest consists of 100 random trees. Each tree is trained by randomly selecting int($\log_2 M$+1) features as candidates for evaluation towards generating each non-leaf node of the tree, where M is the number of features in the full feature set obtained at iteration i. The second model is built using the FPA method [2], where the built forest consists of 100 trees. Each tree is trained heuristically by using the CART algorithm [8] alongisde the cost-complexity pruning (CCP) method [8], where the number of pruning folds is set to 3, i.e., a 3-fold cross validation is conducted on the training data set to obtain the pruned tree. The third classification model is built by adopting jointly the Bagging approach [6] and the Random Subspace method [22]. In particular, 10 data samples are drawn randomly from the original training data set using the Bagging approach [6]. Then, 10 feature subsets are drawn from the full feature set of each training data sample by adopting the Random Subspace method [22], where the size of each subspace is set to 0.6. Finally,

100 decision trees are produced in total by using the C4.5 algorithm [37] alongside the error-based pruning method [20]. In terms of pruning, the confidence factor is set to 0.25, alongside the consideration of the subtree raising operation. Moreover, the minimum description length (MDL) correction [37] is used, in the case of selecting continuous attributes for generating non-leaf nodes of a decision tree. Similar to the way of building the third classification model, the fourth classification model is also built by adopting jointly the Bagging approach [6] and the Random Subspace method [22], but the only difference is that the RIPEER algorithm [13] is used instead of the C4.5 algorithm [37] to produce 100 sets of "if-then" rules rather than 100 decision trees. The RIPEER algorithm [13] is set to involve 2 runs of rule optimization and use 1/3 training data for rule pruning. At the end of each iteration i of building ensemble committee EC, a 3-fold cross validation, which is based on the procedure shown in Fig. 2, is undertaken to generate class vectors for adding new features into the feature set for each instance. After the 3-fold cross validation, it is also determined automatically whether the learning task continues by going for the next iteration i+1, i.e., the learning task would normally continue unless the learning performance (i.e., the classification accuracy measured using the 3-fold cross validation on the training set) is not advanced any further. The proposed approach is compared with a very recent approach MSMRL [28] as well as all the other methods (i.e., the C4.5 method [37], the Prism method [9] and the PrismCTC method [31]) that have been compared with the MSMRL method in [28]. The settings of the hyper-parameters for these existing methods (i.e., the C4.5 method, the Prism method, the PrismCTC method and the MSMRL method) are kept the same as the ones described in [28]. The experiments on the 20 data sets are conducted through random splitting of data into training and test sets. In particular, 70% of a data set is selected for training and the rest of the data set is taken for testing. For each data set, the random data splitting is repeated 100 times and the average accuracy obtained over the 100 runs is used for performance comparison among different approaches. The results on classification accuracy for different methods are presented in Table 2. In particular, the proposed approach shows the top performance among all these existing approaches [9, 28, 31, 37] in 17 out of the 20 cases, i.e., the proposed approach either outperforms all the other methods or performs the same as the best performing one(s) among the other methods. In columns 4-7 of Table 2, the four headers "PrismCTC1", "PrismCTC2", "PrismCTC3" and "PrismCTC4" represent that the PrismCTC algorithm is adopted with four different settings of the hyper-parameter named as "rule quality measure", where the four selected measures of rule quality are referred to as "confidence", "J-measure", "lift" and "leverage", respectively, which are explained in [31] in details. In comparison with the C4.5 algorithm [37], the proposed approach performs better in 18 out of the 20 cases. In the remaining 2 cases, the

proposed approach performs the same as the C4.5 algorithm. In comparison with the Prism algorithm [9], the proposed approach performs better in 18 out of the 20 cases. In the remaining 2 cases, the proposed approach performs worse than the Prism algorithm. In comparison with the PrismCTC1 algorithm [31], the proposed approach performs better in 16 out of the 20 cases. In the remaining 4 cases, the proposed approach performs the same as the PrismCTC1 algorithm in 1 case and performs worse than the PrismCTC1 algorithm in 3 cases. In comparison with the PrismCTC2 algorithm [31], the proposed approach performs better in 14 out of the 20 cases. In the remaining 6 cases, the proposed approach performs the same as the PrismCTC2 algorithm in 4 cases and performs worse than the PrismCTC2 algorithm in 2 cases. In comparison with the PrismCTC3 algorithm [31], the proposed approach performs better in 14 out of the 20 cases. In the remaining 6 cases, the proposed approach performs the same as the PrismCTC3 algorithm in 3 cases and performs worse than the PrismCTC3 algorithm in 3 cases. In comparison with the PrismCTC4 algorithm [31], the proposed approach performs better in 18 out of the 20 cases. In the remaining 2 cases, the proposed approach performs worse than the PrismCTC4 algorithm. In comparison with the MSMRL algorithm [28], the proposed approach performs better in 11 out of the 20 cases. In the remaining 9 cases, the proposed approach performs the same as the MSMRL algorithm in 7 cases and performs worse than the MSMRL algorithm in 2 cases.

Table 2
Classification accuracy.

Data sets	C4.5 [37]	Prism [9]	PrismCTC1 [31]	PrismCTC2 [31]	PrismCTC3 [31]	PrismCTC4 [31]	MSMRL [28]	The proposed method
Anneal	0.98	0.98	**0.99**	**0.99**	**0.99**	0.98	0.97	**0.99**
Balance-scale	0.78	0.83	**0.85**	**0.85**	0.84	**0.85**	0.80	0.82
Breast-cancer	0.67	0.67	0.66	0.65	0.64	0.67	0.69	**0.70**
Breast-w	0.94	0.93	0.95	0.95	0.95	0.95	**0.96**	**0.96**
Credit-a	0.83	0.80	0.77	0.77	0.78	0.81	0.84	**0.87**
Credit-g	0.68	0.74	0.70	0.68	0.68	0.70	0.70	**0.76**
Cylinder-bands	0.58	0.69	0.70	0.70	0.69	**0.72**	0.69	0.65
Dermatology	0.94	0.84	0.90	0.91	0.88	0.85	0.94	**0.97**
Diabetes	0.72	0.70	0.70	0.69	0.70	0.73	**0.76**	**0.76**
Hepatitis	0.76	0.76	0.82	0.81	0.78	0.83	0.81	**0.84**
Ionosphere	0.89	0.90	0.92	0.92	0.92	0.92	**0.93**	**0.93**
Iris	0.94	0.88	0.94	0.94	0.93	0.92	**0.96**	**0.96**
Kr-vs-kp	**0.99**	0.98	0.98	**0.99**	**0.99**	0.98	**0.99**	**0.99**
Labor	0.80	0.88	0.81	0.85	0.87	0.84	0.84	**0.90**
Lymph	0.76	0.78	0.79	0.77	0.78	0.76	0.76	**0.82**
Sponge	**0.93**	0.91	0.90	**0.93**	**0.93**	0.92	**0.93**	**0.93**
Tae	0.53	0.49	0.59	0.57	0.58	0.45	**0.61**	0.57
Vote	0.95	0.93	0.94	0.94	0.94	0.90	**0.96**	**0.96**
Wine	0.91	0.84	0.93	0.93	0.90	0.94	0.96	**0.97**
Zoo	0.92	0.61	0.80	0.86	0.63	0.86	0.92	**0.93**

In order to identify whether the degree to which the proposed approach outperforms each of the other methods is statistically significant, we conduct statistical analysis by taking the Wilcoxon signed-rank tests [14]. The results obtained through the statistical analysis are shown in Table 3, which indicate that the proposed approach performs significantly better than each of the other methods, given that the *p*-value obtained for each pairwise comparison (e.g., C4.5 vs the proposed method) is less than 0.05.

Table 3
Statistical analysis using Wilcoxon signed-rank tests.

Compared methods	Number of positive cases	Number of negative cases	Number of ties	*p*-value	Comments
C4.5 vs the proposed method	18	0	2	0%	Significantly better than C4.5
Prism vs the proposed method	18	2	0	0%	Significantly better than Prism
PrismCTC1 vs the proposed method	16	3	1	0.30%	Significantly better than PrismCTC1
PrismCTC2 vs the proposed method	14	2	4	0.30%	Significantly better than PrismCTC2
PrismCTC3 vs the proposed method	14	3	3	0.20%	Significantly better than PrismCTC3
PrismCTC4 vs the proposed method	18	2	0	0.20%	Significantly better than PrismCTC4
MSMRL vs the proposed method	11	2	7	3.40%	Significantly better than MSMRL

The results shown in Table 2 and Table 3 generally indicate that the adoption of the proposed deep rule ensemble creation approach can achieve an improvement of the performance of rule-based classification through iteratively increasing the learning depth and involving multiple ways of the heuristic creation of the diversity. In particular, Table 2 shows that a considerable improvement of the classification performance is achieved by using the proposed approach for some data sets, such as “Credit-a”, “Credit-g”, “Dermatology”, “Labor” and “Lympth”, in comparison with most or even all of the other methods. Table 2 also shows some cases that the proposed approach

performs the same as some of the other methods, while all the other methods achieve generally good performance (i.e., 90% or higher accuracy of classification) on the data sets, such as “Anneal”, “Breast-w”, “Kr-vs-kp” and “Vote”. In addition, Table 2 shows that the proposed approach outperforms the C4.5 algorithm [37] but performs worse than some of the other methods [9, 28, 31] on the “Balance-scale”, “Cylinder-bands” and “Tae” data sets. In the first two cases (i.e., on the “Balance-scale” and “Cylinder-bands” data sets), the results show that the Prism algorithm [9] and all its variants [31] (i.e., “PrismCTC1”, “PrismCTC2”, “PrismCTC3” and “PrismCTC4”) generally perform better than the C4.5 algorithm [37], which indicate that the nature of decision tree learning algorithms [7, 8, 37] may generally not suit well the characteristics of the two data sets. As a result, the majority of the base classifiers that form ensemble committee *EC* cannot perform sufficiently well on the two data sets, given that 75% of the base classifiers in ensemble committee *EC* are produced by various algorithms [7, 8, 37] of decision tree learning. In comparison with the MSMRL method [28], the proposed approach shows better performance on the “Balance-scale” data set, but the performance obtained by the proposed approach is worse on the “Cylinder-bands” data set. Moreover, we can see from Table 2 that the adoption of the MSMRL method [28] leads to better performance than the use of the C4.5 algorithm [37], but the performance improvement achieved on the “Cylinder-bands” data set is much larger than the improvement achieved on the “Balance-scale” data set, i.e., an 11% increase of the classification accuracy is achieved on the “Cylinder-bands” data set, whereas the accuracy is increased by 2% on the “Balance-scale” data set. The above phenomenon indicates that both the proposed approach and the MSMRL method [28] have the ability to outperform the C4.5 algorithm [37] on the two data sets, but the MSMRL method [28] is considered more effective for achieving a performance improvement on the “Cylinder-bands” than for achieving an improvement on the “Balance-scale” data set, where the effectiveness of the MSMRL method [28] is even better than the one of the proposed approach in achieving a performance improvement on the “Cylinder-bands” data set. Regarding the third case on the “Tae” data set that the proposed approach performs worse than some of the other methods [9, 28, 31], the results shown in Table 2 indicate that the data set is generally not suitable for any of the methods to produce rule-based classification models, given that all of the methods consistently get low performance (i.e., no greater than 61%). In this case, the chance of a performance improvement would be much limited, since it is generally necessary to avoid having an individual base classifier of low performance, in order to achieve good classification performance through deploying an ensemble committee produced by the proposed approach.

In order to show how deep the ensemble model (i.e., ensemble committee *EC*) produced on various data sets can be, we provide the statistics on the model depth (i.e., how many levels ensemble committee *EC* involves), as shown in Table 4. In particular, while the experiment on each data set involves 100 runs, columns 2 and 3 of Table 4 indicate the minimum model length and the maximum model length, respectively, among the model depth values obtained over 100 runs. Moreover, the last column of Table 4 shows the average model depth obtained over 100 runs on each data set.

Table 4
Statistics on model depth (number of levels that a model involves).

Data sets	Minimum model depth	Maximum model depth	Average model depth
Anneal	1	4	2.97
Balance-scale	2	4	2.14
Breast-cancer	2	5	3.19
Breast-w	2	5	3.28
Credit-a	3	5	3.33
Credit-g	2	3	2.47
Cylinder-bands	2	2	2.00
Dermatology	2	4	2.41
Diabetes	3	5	3.49
Hepatitis	2	4	2.53
Ionosphere	2	5	2.48
Iris	1	4	2.54
Kr-vs-kp	2	5	3.20
Labor	2	4	2.17
Lymph	2	5	2.63
Sponge	1	3	1.06
Tae	3	5	3.49
Vote	2	6	3.25
Wine	1	3	1.98
Zoo	1	4	2.36

The statistics shown in Table 4 indicate that the design of the proposed approach generally leads to the generated ensemble model involving multiple levels. Since the depth of the generated ensemble model *EC* is automatically determined during the learning process, i.e., it is not a pre-defined hyper-parameter, the statistics on the model depth indicate that updating the feature set FS_i by importing newly created features at the end of each learning iteration i generally results in the advances of the learning

performance achieved at the next iteration $i+1$. As a result, the learning depth is increased automatically as expected leading to the production of a deep model that consists of multiple levels, where multiple ensembles are involved in each level. According to Table 4, the average model depth obtained on the vast majority of the 20 data sets is greater than 2. Moreover, the maximum model depth obtained on the data sets is mostly greater than 3. The above statistics indicate that the ensemble model *EC* produced by the proposed approach can naturally get deeper during the learning process. However, we can see from column 2 of Table 4 that the minimum model depth obtained on five data sets is 1, which indicates the possibility that the increase of the learning depth may not be necessary in some specific cases, i.e., depending on the data characteristics, the performance may already reach the optimal status at the first learning iteration. Therefore, it is a more reasonable strategy to make the learning depth self-adaptive to the data characteristics than to pre-define the learning depth as a hyper-parameter of the proposed approach.

5. Conclusions

In this paper, we have proposed a feature expansion driven approach for deep rule ensemble creation, which essentially involves multiple iterations of learning and an automatic creation of new features at each iteration to increase the dimensionality of the feature space used for building deeper rule ensembles at any subsequent iterations. In the above setting, multiple methods of ensemble creation are adopted to produce diverse ensembles at each iteration, and the number of iterations is increased towards increasing the depth of learning for advancing the learning performance, i.e., the increase of the number of iterations normally continues until the learning performance measured using the validation data is not advanced any further. The proposed approach has been compared with some existing methods [9, 28, 31, 37] of rule learning and ensemble creation using various data sets. The experimental results show that our proposed approach performs considerably better than the other approaches [9, 28, 31, 37] in the majority of the cases. Moreover, the statistical analysis also shows that the extent to which the proposed approach outperforms each of the other methods [9, 28, 31, 37] is statistically significant. In the future, we will investigate the use of the fuzzy set theory [47] to produce multiple fuzzy rule ensembles at each iteration of learning and explore the effectiveness of creating new features at each iteration through the fuzzification of the features obtained at the previous iterations. It is also worth to conduct further studies on adopting multiple ways of constructing fuzzy membership functions [32, 43] to enable the multi-channel creation of deep rule ensembles. In other words, the adoption of each way of constructing fuzzy membership functions leads to

a specific channel for producing deep rule ensembles through multiple iterations of learning, in order to create the further diversity among the deep rule ensembles produced at different channels, while the depth of learning at each channel is increased independently through involving multiple iterations. In addition, we will investigate how granular computing techniques [10, 11, 30, 35, 41, 48] can be incorporated effectively into the proposed approach of deep rule ensemble creation towards further increasing the depth of learning.

Acknowledgements

The author would like to acknowledge the support from the School of Computer Science and Informatics at the Cardiff University, United Kingdom.

References

[1] M.N. Adnan, M.Z. Islam, Forest CERN: A new decision forest building technique, in: Proceedings of the 20th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Auckland, New Zealand, pp. 304-315, 2016.

[2] M.N. Adnan, M.Z. Islam, Forest PA: Constructing a decision forest by penalizing attributes used in previous trees, Expert Systems with Applications 89 (2017) 389-403.

[3] M. Azmi, G.C. Runger, A. Berrado, Interpretable regularized class association rules algorithm for classification in a categorical data space, Information Sciences 483 (2019) 313-331.

[4] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: Bagging, boosting, and variants, Machine Learning 36 (1-2) (1999) 105-139.

[5] H. Boström, L. Asker, Combining divide-and-conquer and separate-and-conquer for efficient and effective rule induction, in: Proceedings of the 9th International Workshop on Inductive Logic Programming, Bled, Slovenia, pp. 33-43, 1999.

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

[7] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5-32.

[8] L. Breiman, J.H. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, Chapman and Hall/CRC, Monterey, California, U. S. A., 1984.

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

[10] S.M. Chen, H.P. Chu, T.W. Sheu, TAIEX forecasting using fuzzy time series and automatically generated weights of multiple factors, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 42 (6) (2012) 1485-1495.

[11] S.M. Chen, G.M.T. Manalu, J.S. Pan, H.C. Liu, Fuzzy forecasting based on two-factors second-order fuzzy-trend logical relationship groups and particle swarm optimization techniques, IEEE Transactions on Cybernetics 43 (3) (2013) 1102-1117.

[12] P. Clark, T. Niblett, The CN2 induction algorithm, Machine Learning 3 (4) (1989) 261-283.

[13] W.W. Cohen, Fast effective rule induction, in: Proceedings of the 12th International Conference on Machine Learning, Tahoe City, California, U. S. A., pp. 115-123, 1995.

[14] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1-30.

[15] J.H. Friedman, Greedy function approximation: A gradient boosting machine, The Annals of Statistics 29 (5) (2001) 1189-1232.

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

[17] T.R. Gabriel, M.R. Berthold, Influence of fuzzy norms and other heuristics on mixed fuzzy rule formation, International Journal of Approximate Reasoning 35 (2) (2004) 195-202.

[18] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees. Machine Learning 63 (1) (2006) 3-42.

[19] L.H. Gilpin, D. Bau, B.Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explainations: An overview of interpretability of machine learning, in: Proceedings of 5th IEEE International Conference on Data Science and Advanced Analytics, Turin, Italy, pp. 80-89, 2018.

[20] L.O. Hall, R. Collins, K.W. Bowyer, R. Banfield, Error-based pruning of decision trees grown on very large data sets can work!, in: Proceedings of 14th IEEE International Conference on Tools with Artificial Intelligence, Washington, DC, U. S. A., 2002.

[21] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer, New York, U. S. A., 2009.

[22] T.K. Ho, The random subspace method for constructing decision forests, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (8) (1998) 832-844.

[23] L.I. Kuncheva, A theoretical study on six classifier fusion strategies, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (2) (2002) 281-286.

[24] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2) (2003) 181-207.

[25] L.I. Kuncheva, J.C. Bezdek, R.P.W. Duin, Decision templates for multiple classifier fusion: an experimental comparison, Pattern Recognition 34 (2) (2001) 299-314.

[26] R.G. Leiva, A.F. Anta, V. Mancuso, P. Casari, A novel hyperparameter-free approach to decision tree construction that avoids overfitting by design, IEEE Access 7 (2019) 99978-99987.

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

[28] H. Liu, S.M. Chen, Multi-stage mixed rule learning approach for advancing performance of rule-based classification, Information Sciences 495 (2019) 65-77.

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

[30] H. Liu, M. Cocea, Nature-inspired framework of ensemble learning for collaborative classification in granular computing context, Granular Computing 4 (4) (2019) 715-724.

[31] H. Liu, S.M. Chen, M. Cocea, Heuristic target class selection for advancing performance of coverage-based rule learning, Information Sciences 479 (2019) 164-179.

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

[33] J. Maudes, J.J. Rodriguez, C. Garcia-Osorio, N. Garcia-Pedrajas, Random feature weights for decision tree ensemble construction, Information Fusion 13 (1) (2012) 20-30.

[34] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and Systems Magazine 6 (3) (2006) 21-45.

[35] J. Qi, L. Wei, Q. Wan, Multi-level granularity in formal concept analysis, Granular Computing 4 (3) (2019) 351-362.

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

[37] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers, San Francisco, California, U. S. A.,1993.

[38] J.J. Rodriguez, L.I. Kuncheva, C.J. Alonso, Rotation Forest: A new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (10) (2006) 1619-1630.

[39] L. Rokach, O. Maimon, Top-down induction of decision trees classifiers - a survey. IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews 35 (4) (2005) 476-487.

[40] R.E. Schapire, Y. Freund, P. Bartlett, W.S. Lee, Boosting the margin: A new explanation for the effectiveness of voting methods, The Annals of Statistics 26 (5) (1998) 1651-1686.

[41] Y. Shi, Y. Huang, C. Wang, Q. He, Attribute reduction based on the Boolean matrix. Granular Computing 4 (3) (2019) 313-322.

[42] M. Sikora, Ł. Wróbel, A. Gudyś, GuideR: A guided separate-and-conquer rule learning in classification, regression, and survival settings. Knowledge-Based Systems 173 (2019) 1-14.

[43] P.K. Singh, Concept lattice visualization of data with *m*-polar fuzzy attribute, Granular Computing 3 (2) (2018) 123-137.

[44] D.M.J. Tax, R.P.W. Duin, M.V. Breukelen, Comparison between product and mean classifier combination rules, in: Proceedings of the 1st International Workshop on Statistical Techniques in Pattern Recognition, Prague, Czech Republic, pp. 165-170, 1997.

[45] K.M. Ting, I.H. Witten, Stacking bagged and dagged models, in: Proceedings of the Fourteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc, San Francisco, California, U. S. A., pp. 367-375, 1997.

[46] Ł. Wróbel, A. Gudy´s, M. Sikora, Learning rule sets from survival data, BMC Bioinformatics 18 (2017) 1-13.

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

[48] W. Zhang, X. Wang, X. Yang, X. Chen, P. Wang, Neighborhood attribute reduction for imbalanced data, Granular Computing 4 (3) (2019) 301-311.

[49] Z.H. Zhou, Ensemble Methods: Foundations and Algorithms, CRC, Boca Raton, Florida, U. S. A., 2012.

[50] Z.H. Zhou, J. Feng, Deep Forest: Towards an alternative to deep neural networks, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, pp. 3553-3559, 2017.