Extracting Symbolic Rules from Trained Neural Network Ensembles

Zhi-Hua Zhou*, Yuan Jiang and
Shi-Fu Chen
National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, P.R.China
E-mail: zhouzh@nju.edu.cn, jy@ai.nju.edu.cn, chensf@nju.edu.cn

Neural network ensemble can significantly improve the generalization ability of neural network based systems. However, its comprehensibility is even worse than that of a single neural network because it comprises a collection of individual neural networks. In this paper, an approach named REFNE is proposed to improve the comprehensibility of trained neural network ensembles that perform classification tasks. REFNE utilizes the trained ensembles to generate instances and then extracts symbolic rules from those instances. It gracefully breaks the ties made by individual neural networks in prediction. It also employs specific discretization scheme, rule form, and fidelity evaluation mechanism. Experiments show that with different configurations, REFNE can extract rules with good fidelity that well explain the function of trained neural network ensembles, or rules with strong generalization ability that are even better than the trained neural network ensembles in prediction.

Keywords: neural networks, neural network ensembles, rule extraction, machine learning, comprehensibility

1. Introduction

Neural network ensemble is a learning paradigm where a collection of a finite number of neural networks is trained to solve a problem. It originates from Hansen and Salamon's work [16], which shows that the generalization ability of a neural network based system can be significantly improved through ensembling neural networks, i.e. training several neural networks and then combining their predictions. Since it behaves remarkably well, neural network ensemble is regarded as a promising methodology that can profit not only experts in neural computing but also ordinary engineers in real-world applications. Recently, it has become a very hot topic in both machine learning and neural computing communities [38].

Neural network is regarded as a blackbox technology because of the lack of comprehensibility. The knowledge learned by neural networks is hard to be understood by users because it is concealed in a large amount of connections. Moreover, it is difficult to give an explicit explanation for the reasoning process of trained neural networks. In general, comprehensibility is one of the required characteristics of reliable systems. Baum and Haussler [2] claim that if a trained neural network could generate correct answers to a large number of training instances then it is reliable for dealing with future unseen instances. However, such a claim does not wipe off the desire for the comprehensibility. Therefore many researchers have addressed the issue of improving the comprehensibility of neural networks, where the most attractive solution is to extract symbolic rules from trained neural networks [1].

Since neural network ensembles comprise several or many individual neural networks, their behavior are more difficult to be understood by users than that of single neural networks. In other words, the comprehensibility of a neural network ensemble is worse than that of a single neural network. However, until now few works have addressed the issue of improving the comprehensibility of neural network ensembles.

In this paper, an approach named REFNE (Rule Extraction From Neural network Ensemble) is proposed, which is designed to extract symbolic rules from trained neural network ensembles that perform classification tasks. REFNE utilizes trained

*Corresponding author: Zhi-Hua Zhou, National Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, P.R.China, E-mail: zhouzh@nju.edu.cn, Tel: +86-25-359-3163, Fax: +86-25-330-0710.

ensembles to generate a number of instances and then extracts rules from those instances. It could gracefully breaks the ties made by individual neural networks in prediction. Instead of discretizing all the continuous attributes at the beginning of the extraction of symbolic rules, REFNE adopts a specific discretization scheme so that different continuous attributes can be discretized to different number of intervals and unnecessary discretization can be avoided. The extracted rules are expressed in priority form so that the time-consuming conflict check, which prevents the specific rules from being taken over by the general ones, is omissible when the rules are in use. Moreover, since REFNE can generate very accurate rules, it can also be used as a rule learning approach.

The rest of this paper is organized as follows. In Section 2, we briefly review previous works on neural network ensembles and rule extraction from neural networks. In Section 3, we present REFNE. In Section 4, we report the experimental results. In Section 5, we discuss some issues related to REFNE and rule extraction from neural network ensembles. Finally in Section 6, we conclude and indicate several issues for future works.

2. Previous work review

2.1. Neural network ensemble

In the beginning of the 1990s, Hansen and Salamon [16] have shown that the generalization ability of a neural classifier can be significantly improved through training several neural classifiers and then combining their predictions via voting. Since neural network ensemble has great potential, many researchers run into this area and this technology has already been successfully applied to many domains such as optical character recognition [17, 26], face recognition [14, 18], scientific image analysis [6], medical diagnosis [48], seismic signals classification [39], *etc.*

In general, a neural network ensemble is built in two stages, i.e. training several individual neural networks and then combining their predictions.

As for training individual neural networks, the most impressive approaches are Boosting and Bagging. Boosting is proposed by Schapire [33] and improved by Freund *et al.* [9, 10]. It generates a series of neural networks whose training sets are determined by the performance of former ones. Training instances that are wrongly predicted by former networks will play more important roles in the training of later networks. Bagging is proposed by Breiman [5] based on bootstrap sampling [8]. It generates several training sets from the original training set and then trains an individual neural network from each generated training set. There are also many other approaches for training individual neural networks. Examples are as follows. Hampshire and Waibel [15] utilize different objective functions to train different individual neural networks. Cherkauer [6] trains individual networks with different number of hidden units. Maclin and Shavlik [25] initialize individual networks at different points of the weight space. Krogh and Vedelsby [23] employ cross-validation to create individual networks. Opitz and Shavlik [28] exploit genetic algorithm to train diverse knowledge based neural networks. Yao and Liu [46] ensemble all the individuals in an evolved population of neural networks. Zhou *et al.* [49] employs genetic algorithm to select a subset of a population of neural networks.

As for combining the predictions of individual neural networks, the approaches used for classification and regression are quite different because those two kinds of tasks have distinct characteristics, i.e. the outputs of classification are class labels while those of regression are real values. At present, the most prevailing approaches for classification tasks are plurality voting or majority voting [16], and those for regression tasks are simple averaging [27] or weighted averaging [29]. There are also many other approaches for combining individual predictions. Examples are as follows. Wolpert [45] utilizes learning systems to combine individual predictions. Jimenez [20] employs dynamic weights determined by the confidence of the individual networks to combine the predictions. Ueda [43] exploits optimal linear weights to combine individual predictions based on statistical pattern recognition theory.

2.2. Rule extraction from neural networks

Rule extraction from trained neural networks originates from Gallant's work [12] on connectionist expert systems, where he exploits the order of the available attributes based on their inference strengths to find a rule that could explain the conclusion reached by the neural network for a given case. From the style of the exploration of neural networks, present rule extraction approaches could be roughly categorized to two classes, i.e. architecture-analysis-based or function-analysis-based approaches.

Architecture-analysis-based approaches regard rule extraction as a search process that maps the architecture of a trained neural network to a set of rules. Examples are as follows. Fu [11] extracts rules for each output unit through searching for subsets of connections whose summed weights exceed the bias of the unit. Towell and Shavlik [42] cluster similar weights in knowledge based artificial neural networks to construct equivalent classes and then extract MOFN rules. Sestito and Dillon [35] generate conjunctive rules using a multi-layered network to measure the closeness between inputs and using an inhibitory single-layered network to measure the relevance between inputs and outputs. Krishnan [22] orders the incoming weights and forming a combination tree for a unit to generate rules. Setiono [36] extracts rules through clustering the activation values of hidden units and repeatedly splitting the network to sub-networks. Setiono and Leow [37] divide the activation values of relevant hidden units to two subintervals and then find out the set of relevant connections to those relevant units to construct rules. Ishikawa [19] generates a small number of dominant rules at an earlier stage and less dominant rules or exceptions at later stages with the help of structural learning with forgetting.

Function-analysis-based approaches do not disassemble the architecture of the trained neural networks. Instead, they regard the network as an entity and try to extract rules that could explain its function. Examples are as follows. Saito and Nakano [32] select useful rules from a candidate rule set that is generated through examining activation levels of a network while inputs are gradually changed. Craven and Shavlik [7] regard rule extraction as a learning task and extract conjunctive rules with the help of two oracles. Thrun [41] extracts rules through analyzing the input-output behavior of a network with Validity Interval Analysis. Benitez *et al.* [3] establish the equality between a certain class of neural networks and fuzzy rule based systems so that the network could be explained by a set of fuzzy rules. Schmitz *et al.* [34] grow binary decision trees from trained neural networks with the help of an attribute selection criterion based on significance analysis. Tanaka *et al.* [40] extract linguistic rules through feeding rule antecedents to a network and then comparing the fuzzy outputs of the network with linguistic values. Garcez *et al.* [13] define a partial ordering on the set of input vectors and then extract nonmonotonic rules from the network.

3. REFNE

3.1. Motivation

REFNE is a function-analysis-based approach. Suppose that there is a trained neural network ensemble $\mathcal{E}$. If an input vector $A_k = (a_1, a_2, ..., a_n)$ is fed to $\mathcal{E}$, an output vector $C_k = (c_1, c_2, ..., c_q)$ will be generated by the ensemble. Through combining A_k and C_k, an instance (A_k, C_k) appears. Such an instance is generated by $\mathcal{E}$ and could represent the function of $\mathcal{E}$ at the point A_k in the instance space. We believe that if we have a large instance set $\mathcal{S}$ that comprises many instances generated by $\mathcal{E}$, and the instances cover the whole instance space as more as possible, then the function of $\mathcal{E}$ is encoded in $\mathcal{S}$. Thus, if a comprehensible rule set $\mathcal{R}$ could be extracted from $\mathcal{S}$, the function computed by $\mathcal{E}$ will be approximated by the function computed by $\mathcal{R}$ while the size of $\mathcal{S}$ approaching infinity, that is,

$$\lim_{|\mathcal{S}| \to \infty} func(\mathcal{R}) = func(\mathcal{E}) \qquad (1)$$

It may seem strange that here learning is performed twice to solve a problem, that is, a neural network ensemble is trained and then rules are learned. The reason is that the goal of REFNE is to improve the comprehensibility of trained neural network ensembles, which means that the ensembles have already been trained and the "real" cost of REFNE is the second phase learning. Moreover, as shown in later sections, we believe that the cost of twice learning is worthwhile even without the consideration of the goal because REFNE can generate very accurate rules, which makes it a competitive alternative to present rule learning approaches.

3.2. Data generation

The instance set $\mathcal{S}$ could be generated through slowly sliding A_k in the instance space, that is, varying each a_i $(i = 1, 2, ..., n)$ across its value range so that diverse input patterns could appear as frequently as possible. For a categorical attribute, it seems easy to make all the possible values appear in turn. For a continuous attribute, it seems necessary to sample it across its value range with a high frequency. For example, 100 patterns may be generated for an attribute whose value range is $[0, 1]$, i.e. the attribute is sampled with the interval 0.01.

However, such a process is infeasible when dealing with real-world tasks because the number of generated instances will increase exponentially as the number of attributes increases. Suppose the number of attributes is N and the number of possible values of each attribute is M, then the number of instances generated by the process is M^N. Even when facing a toy task where $N = 10$ and $M = 5$, the process will generate about $1,000,000$ instances, which is a heavy burden to both the storage and the computational resources.

In order to relax such a heavy burden, REFNE generates a random set of input vectors in the value ranges of the input attributes. Those input vectors are fed to the trained neural network ensemble, and then the outputs from the ensemble are combined with their corresponding inputs to make up instances that constitute $\mathcal{S}$. The size of $\mathcal{S}$ may impact the extracted rules in such a way that the bigger the size of $\mathcal{S}$ is, the more accurate the extracted rules are. Experiments show that when the size of $\mathcal{S}$ is only twice of that of the training set of the neural network ensemble, very accurate rules can be extracted by REFNE. Note that if the training set of the ensemble is available, it is good to feed the input vectors in the training set to the ensemble, get the outputs, combine the inputs and outputs to form instances, and include those instances in $\mathcal{S}$. In this way, instances that are important to determine the function of the ensemble will not be ignored when rules are extracted.

3.3. Ties broken

At present, plurality voting is very popular in combining the predictions of individual neural networks in an ensemble, which regards the class receiving the most number of votes as the output. When a trained ensemble employing plurality voting is used to classify some instances, ties may appear among individual neural networks, e.g. individual networks each vote for a different class. Most ensembles tackle this problem with *ad hoc* mechanisms, such as regarding the first class as the output, randomly selecting a class as the output, designating the class determined by the first individual network as the output, *etc.* If the instances whose output vectors are determined by those *ad hoc* mechanisms are included in $\mathcal{S}$, then some undesired impacts may be brought to the extraction of rules.

For example, suppose that on input vector I a tie occurs between class A and class B, and the ensemble randomly selects A as the output. If a rule were extracted from I then A would be the rule consequent, which would hamper the extraction of a rule with same antecedents but with B as the consequent. We call such a rule as a *suspectable rule* because it comes from a suspicious reasoning process of the ensemble, i.e. the ensemble outputs A in this time but it may outputs B in the next time under the same input condition.

As described in later sections, the rules extracted by REFNE are expressed in priority form where the earlier extracted rules implicitly act as antecedents of the later extracted rules. Therefore such a suspectable rule will improperly bias its successive rules. For example, as described above, if a rule with A as the consequent is generated, then rules that implicitly regard B as antecedent cannot be generated. In order to get rid of such bias, REFNE excludes instances that may lead to suspectable rules in the generation of $\mathcal{S}$. In other words, if ties appear in determining the output vector of an input vector, then the input and output vectors will not be combined to form an instance of $\mathcal{S}$.

3.4. Discretization

Once the instance set $\mathcal{S}$ is created, a rule set $\mathcal{R}$ could be extracted via REFNE. Note that REFNE always extracts rules from categorical attributes. Only when new rules can not be extracted out, REFNE resorts to the continuous attribute that has the best clustering effect, i.e. the number of discretized clusters are the smallest. The discretized continuous attribute will be regarded as a new categorical attribute and used along with previous categorical attributes for successive rule extraction.

The discretization is performed by a modified version of ChiMerge [21]. The instances are sorted according to the value of the attribute being discretized. Initial clusters are formed by making each unique value as a cluster. The χ^2 value of adjacent clusters is computed and the pairs of adjacent clusters with the lowest χ^2 value are merged. In Kerber's original algorithm [21], merging continues until the χ^2 values of all pairs of clusters exceed a user-defined parameter χ^2-threshold. Instead of setting a fixed threshold as the stopping criterion for merging, REFNE continues the merging as long as there is no instances that belong to different classes assigned identical discretized values. Such a process has been adopted by Liu and Setiono [24] before.

The discretization process of REFNE has some advantages. Firstly, different continuous attributes may be distributed differently in their value range. If they are discretized to equal number of intervals, some helpful information may be obliterated. In REFNE, since each time only one continuous attribute is processed, different attributes could be discretized to different number of intervals. Secondly, in most cases, especially when there are many attributes, some continuous attributes will not appear in the extracted rules. Therefore, the effort in discretizing those attributes are wasted. In REFNE, since the extracted rule set $\mathcal{R}$ may have already fully covered $\mathcal{S}$ before all continuous attributes are discretized, unnecessary discretization could be avoided. Thirdly, since the continuous attributes are discretized one by one, the computational cost for discretization gradually descends due to REFNE's specific priority rule formation that will be described later.

On the other hand, such a discretization process also has its disadvantages. Since continuous attributes are processed one by one, the interactions among continuous attributes are ignored. However, such interactions are difficult to be utilized in nature because it is difficult to identify the continuous attributes having strong interactions, difficult to weight those attributes against those having weak interactions, and difficult to exploit the interactions in discretizing those continuous attributes. Therefore REFNE is still competitive in dealing with tasks where there are strong interactions among attributes.

3.5. Rule creation

The rule creation process of REFNE is quite similar to the one used by Fu [11], where a subset $\mathcal{H}$ of input attributes is identified and a rule is created based on it. At first, $\mathcal{H}$ comprises only an attribute a_i that is randomly picked from present categorical attributes. All the possible values of a_i are examined. If there is a value u satisfying the condition that all the instances possessing such a value at a_i in $\mathcal{S}$ fall into class $\mathcal{C}$, then a rule r is created via regarding $a_i = u$ as the antecedent and $\mathcal{C}$ as the consequent. Otherwise a_i is replaced by another categorical attribute a_j and a similar process occurs.

If no rule has been created after examining all the single attributes, then another categorical attribute is appended to $\mathcal{H}$. In other words, $\mathcal{H}$ now comprises two attributes. Suppose those two attributes are a_i and a_j. The resulting rule will have two conjunctive antecedents, i.e. $a_i = u$ AND $a_j = v$. Thus, the number of rule antecedents increases as the number of attributes in $\mathcal{H}$ is increasing. Since we believe that rules with more than three antecedents are incomprehensible to human beings, the maximum number of attributes in $\mathcal{H}$ is limited to three. It is obvious that such a constraint may result in large number of extracted rules. However, we believe that it is worthwhile because such a constraint may also increase the comprehensibility of the extracted rules.

If no rule has been created after all the subsets of categorical attributes are examined, then a continuous attribute is discretized as described in Section 3.4 and regarded as a new categorical attributes. If no more rules could be created after all the continuous attributes are discretized, or $\mathcal{S}$ has already been fully covered by $\mathcal{R}$, then the rule creation process terminates.

Note that it is not very expensive to get an adequate subset $\mathcal{H}$, because the "experience" of previous search could help reduce the computational cost of successive search. For example, suppose new rules cannot be extracted from a set of categorical attributes, i.e. $\{a_i\}$, and a continuous attribute b is discretized. Then only the subsets containing b should be examined because all the other subsets of $\{a_i\} \cup b$ have already been examined.

3.6. Priority formation

In REFNE, when a new rule r is extracted, instances covered by it are removed from the instance set $\mathcal{S}$. In other words, instances covered by $\mathcal{R}$ are not used in successive rule extraction. Here we call the part of instance space covered by $\mathcal{R}$ as *known space* and call the rest *unknown space*. It is obvious that the known space is always growing and the unknown space is always shrinking as the rule set $\mathcal{R}$ is maturing, i.e. the number of extracted rules is increasing. As a result, the computational cost of both the continuous attribute discretization and the attribute subset search gradually decreases. It also requires the extracted rules be expressed in priority form where the earlier the rule is extracted, the higher its priority is, because the rules could be viewed as being extracted from a series of unknown spaces where the later spaces are proper subsets of the earlier ones.

Such a priority rule form has some advantages. Firstly, since prior rules implicitly act as antecedents of later rules, rules requiring more than three antecedents could be extracted. Secondly, since priority rules contain ordering information, when the rules are in use the time-consuming conflict check, which is necessary for rules expressed in common forms to guarantee that the specific rules are not taken over by the general ones, is omissible. Moreover, the priority rule form is also the requirement of the discretization scheme employed by REFNE. The reason is that since the continuous attributes are discretized one by one and the unknown space is always shrinking, the working areas of the rules involving different continuous attributes are also different.

3.7. Fidelity evaluation

Since REFNE creates rules from only categorical attributes, the dimensionality of the unknown space could be measured as the number of available categorical attributes. Thus, discretizing continuous attributes could be viewed as increasing the dimensionality of unknown space when the rule extraction task is too difficult to be accomplished. Such a dimensionality-increasing technique has been shown to be useful in solving many problems [44]. However, it also introduces a new problem, that is, since the unknown space with lower dimensionality is transformed to a space with higher dimensionality, the distribution of instances in $\mathcal{S}$ may be "distorted". In other words, the instances may not well cover the new unknown space. Therefore the extracted rules may only be valid for a part of the unknown space. In order to solve this problem, statistics is introduced in REFNE.

When a new rule r is extracted by REFNE, the fidelity of r is evaluated before r is introduced into $\mathcal{R}$ as a member. The fidelity of r measures the closeness of the performance of r and that of the neural network ensemble $\mathcal{E}$ in present unknown space, which is computed as the percentage of predictions that r and $\mathcal{E}$ agrees.

REFNE performs the fidelity evaluation as follows. Firstly, m instances are generated by $\mathcal{E}$ in the way described in Section 3.2. Among those instances, the ones covered by $\mathcal{R}$ are replaced by newly generated instances until none of the m instances is covered by $\mathcal{R}$. Then, rule r is used to classify those m instances. If the accuracy of r is beyond a pre-set lower bound δ, r is accepted to be a new member of $\mathcal{R}$. Otherwise r is rejected and another rule will be created. Since those m instances are generated by $\mathcal{E}$ so that they could represent the function of $\mathcal{E}$ in present unknown space to some extent, r's accuracy on those instances is actually its fidelity. Experiments show that very accurate rules can be extracted when m equals to the size of $\mathcal{S}$ at that time.

Note that users could roughly determine the number of extracted rules through setting the value of δ. Since big δ will result in high frequency of rule rejection, it is obvious that the smaller the value of δ is, the larger the number of extracted rules is.

3.8. Flowchart of extraction

When REFNE is used to extract rules from a trained neural network ensemble $\mathcal{E}$, at first an instance set $\mathcal{S}$ is generated from $\mathcal{E}$ with ties broken. Then, if an appropriate subset $\mathcal{H}$ of categorical attributes can be identified, a new rule r is generated and its fidelity is tested on some new instances generated from $\mathcal{E}$. If r can be accepted, instances covered by current rule set $\mathcal{R}$ are removed from $\mathcal{S}$ and a new rule is to be generated. If such a $\mathcal{H}$ cannot be identified, a continuous attribute with the best clustering effect is discretized and regarded as a new categorical attribute for rule extraction. If all the continuous attributes have already been discretized or $\mathcal{S}$ becomes an empty set, the rule set $\mathcal{R}$ is tidied and the extraction process terminates.

In summary, the flowchart of REFNE is depicted in Fig. 1.

4. Experiments

4.1. Summary

In this section, we report the experiments on REFNE on six UCI data sets [4] including *balance scale*, *congressional voting records*, *hepatitis*, *iris plant*, *statlog australian credit approval*, and *statlog german credit*. Since we do not want to test the ability of REFNE in dealing with incomplete information, instances having missing values are removed. Information on the data sets used in our experiments is shown in Table 1. For convenience of explanation, in the rest of this paper we use the abbreviated names shown in Table 1 to refer to those data sets.

We run 5-fold cross validation in the experiments. That is, for each data set we divide it into five subsets with similar sizes and similar distribution of classes. Then we perform the tests for the five runs, each with a different subset as the test set and with the union of the other four subsets as the training set. The averaged results of those five runs are recorded as the final results.

In each run, REFNE is utilized to extract rules from a trained neural network ensemble that comprises five single-hidden-layered BP [31] networks combined via plurality voting. The default value of the size of $\mathcal{S}$ is twice of that of the training set of the ensemble, and the default value of δ is 0.9. The training sets of the BP networks are generated by bootstrap sampling [8] from the training set of the run. Note that bootstrap sampling uses only about 63% instances in an original data set to generate a new data set [5]. Therefore for each BP network we use the instances that are not included in its training set as its validation set. During the training process, the generalization error of the network is estimated in each epoch on its validation set. If the validation error does not change in consecutive 5 epochs, the training of the network is terminated in order to avoid overfitting.

4.2. REFNE against other approaches

In this section, REFNE is with its default configuration, i.e. with additional data and ties broken. In other words, it utilizes a random set of input vectors as described in Section 3.2, and breaks the ties as described in Section 3.3. The size of the random data set is twice of that of the training set.

Since the main goal of REFNE is to improve the comprehensibility of trained neural network ensembles, the fidelity is one of the most important measurements. It is computed as the percentage of the predictions on the test set for which the extracted rules and the trained ensemble agrees. The results are tabulated in Table 2. Note that as described in Section 4.1, the results are attained via 5-fold cross validation.

Table 2 shows that the average fidelity of the rules extracted via REFNE with its default configuration is only slightly better than 86%, which is not very high.

Then, we measure the generalization error of the extracted rules on the test set. For comparison, we also measure the generalization error of the trained neural network ensemble and that of a single neural network. Since no other approaches extracting rules from trained neural network ensembles is available, we include a popular rule learning approach, i.e. *C4.5 Rule* [30] for comparison. The results are tabulated in Table 3. Note that all the results in Table 3 are attained via 5-fold cross validation.

Table 3 shows that the generalization ability of the neural network ensemble is better than that of single neural network and that of C4.5 Rule. However, the generalization ability of REFNE rules is even better than that of the trained ensemble by more than ten percentage points. This result is so impressive that it is even beyond our anticipation. Comparing Table 2 and Table 3, we conjecture that the fidelity of REFNE rules is not very high because the generalization ability of those rules is far better than that of the trained neural network ensemble. In other words, it is the test instances that are wrongly classified by the trained ensemble but are correctly classified by the rules cause the rules lose fidelity. This conjecture is further explored in Section 4.3.

Moreover, in order to see the conciseness of the rules extracted by REFNE, we count the rule number and the average, maximum, and minimum

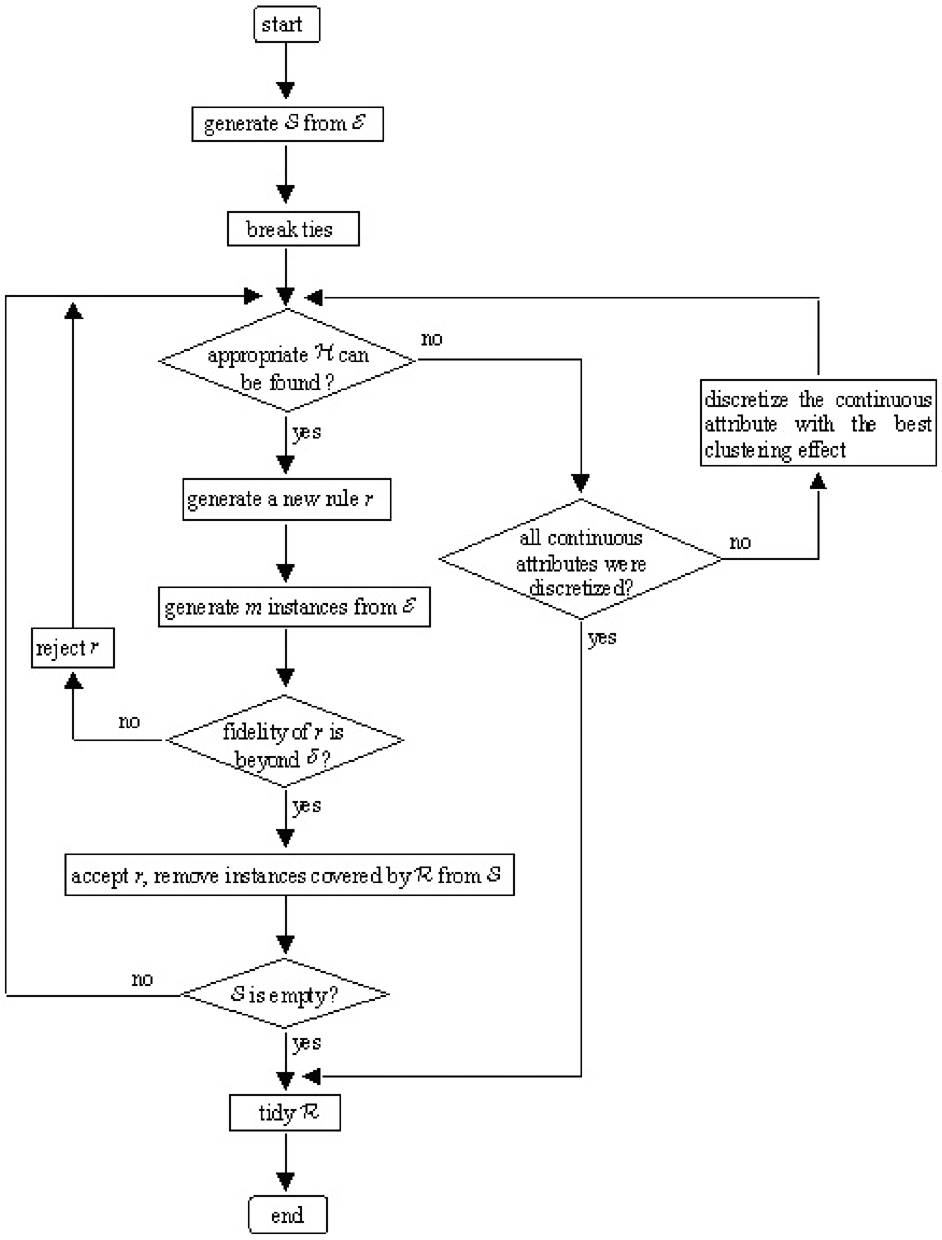
start
generate $\mathcal{S}$ from $\mathcal{E}$
break ties
appropriate $\mathcal{H}$ can be found ?
no
yes
generate a new rule r
generate m instances from $\mathcal{E}$
fidelity of r is beyond δ ?
no
reject r
yes
accept r, remove instances covered by $\mathcal{R}$ from $\mathcal{S}$
$\mathcal{S}$ is empty?
no
yes
all continuous attributes were discretized?
no
discretize the continuous attribute with the best clustering effect
yes
tidy $\mathcal{R}$
end

Fig. 1. Flowchart of REFNE.

number of antecedents of a single rule, of both the rules extracted by REFNE and those generated by C4.5 Rule. The results are shown in Table 4, where C4.5 Rule is abbreviated as C4.5R. Note that all the results in Table 4 are attained via 5-fold cross validation.

Table 4 shows that single rules extracted by REFNE are more concise than those generated by C4.5 Rule because the single rules extracted by REFNE have smaller number of average, maximum, and minimum antecedents. Table 4 also shows that the entire rule set extracted by REFNE is not as concise as that generated by C4.5 Rule, because the rule number of the former is about

Table 1
Data sets used in experiments

				number of attributes		
data set	*abbr.*	*size*	*class*	*total*	*categorical*	*continuous*
balance scale	*balance*	625	3	4	0	4
congressional voting records	*voting*	232	2	16	16	0
hepatitis	*hepatitis*	80	2	19	13	6
iris plant	*iris*	150	3	4	0	4
statlog australian credit approval	*credit-a*	690	2	15	9	6
statlog german credit	*credit-g*	1,000	2	20	13	7

Table 2
Fidelity of rules extracted via REFNE

data set	*balance*	*voting*	*hepatitis*	*iris*	*credit-a*	*credit-g*	*average*
fidelity	87.88%	89.26%	84.50%	96.25%	84.13%	74.10%	86.02%

Table 3
Comparison of generalization error

data set	*REFNE*	*ensemble*	*single NN*	*C4.5 Rule*
balance	6.72%	9.76%	9.12%	37.12%
voting	0.00%	10.74%	4.32%	3.88%
hepatitis	2.50%	15.00%	16.25%	16.26%
iris	2.67%	2.00%	6.00%	4.00%
credit-a	0.87%	15.20%	16.23%	14.93%
credit-g	3.80%	25.10%	28.00%	17.30%
average	2.76%	12.97%	13.32%	15.58%

Table 4
Comparison of rules extracted via REFNE and that generated by C4.5 Rule

	rule number		*ave. antecedents*		*max antecedents*		*min antecedents*	
data set	*REFNE*	*C4.5R*	*REFNE*	*C4.5R*	*REFNE*	*C4.5R*	*REFNE*	*C4.5R*
balance	23.4	17.6	1.18	1.70	2.0	2.4	1.0	1.0
voting	30.0	3.0	2.08	1.26	3.0	1.4	1.0	1.0
hepatitis	36.0	4.8	2.39	1.74	3.0	3.2	1.0	1.0
iris	17.4	4.2	1.59	1.46	2.4	2.6	1.0	1.0
credit-a	39.0	9.8	2.14	2.77	3.0	5.2	1.0	1.0
credit-g	39.4	34.0	1.79	3.07	3.0	6.0	1.0	1.0
average	30.9	12.2	1.86	2.00	2.7	3.5	1.0	1.0

2.5 times as that of the latter. However, we believe that the impressive generalization ability of REFNE rules can greatly offset its weakness in conciseness.

A rule set extracted via REFNE in *voting* task is shown in Table 5 as an example of the extracted rules, which is the rule set with the smallest number of rules in 5 runs.

4.3. REFNE with different configurations

In order to explore why the generalization ability of REFNE rules is so impressive, and to explore whether REFNE can extract rules with better fidelity, we perform more experiments reported in this section.

The default configuration of REFNE that was tested in Section 4.2 is with additional data and ties broken, i.e. it utilizes a random set of input vectors and breaks the ties. Here REFNE with

Table 5

A rule set extracted via REFNE in *voting* task

no.	*rule*
1	physician-fee-freeze → democrat
2	¬education-spending → republican
3	¬handicapped-infants ∧ adoption-of-the-budget-resolution → republican
4	¬handicapped-infants ∧ aid-to-nicaraguan-contras → republican
5	¬water-project-cost-sharing ∧ adoption-of-the-budget-resolution → republican
6	water-project-cost-sharing ∧ mx-missile → republican
7	¬handicapped-infants ∧ ¬ superfund-right-to-sue → democrat
8	¬handicapped-infants ∧ ¬mx-missile → republican
9	¬water-project-cost-sharing ∧ religious-groups-in-schools → republican
10	water-project-cost-sharing ∧ ¬anti-satellite-test-ban → republican
11	water-project-cost-sharing ∧ ¬adoption-of-the-budget-resolution → democrat
12	water-project-cost-sharing ∧ ¬crime → democrat
13	handicapped-infants ∧ synfuels-corporation-cutback → republican
14	¬crime → democrat
15	handicapped-infants ∧ immigration → republican
16	¬religious-groups-in-schools ∧ export-administration-act-south-africa → democrat
17	anti-satellite-test-ban → republican
18	handicapped-infants → democrat

such a configuration is abbreviated as REFNE-*at*. There are also other kinds of configurations, i.e. REFNE with additional data but without ties broken (abbreviated as REFNE-*a*), REFNE without additional data but with ties broken (abbreviated as REFNE-*t*), and REFNE without additional data and without ties broken (abbreviated as REFNE-*n*).

In REFNE-*t* and REFNE-*n*, the random set of input vectors utilized by REFNE-*at* and REFNE-*a* is not used, that is, only the input vectors of the original training set of the neural network ensemble are fed to the trained ensemble to generate instances for rule extraction. On the other hand, the number of instances used by REFNE-*a* and REFNE-*n* in rule extraction is usually bigger than that of REFNE-*at* and REFNE-*t* respectively because REFNE-*at* and REFNE-*t* may exclude some instances generated by the ensemble for ties broken. Therefore according to the number of instances used in rule extraction, the four configurations can be sorted as REFNE-*t*, REFNE-*n*, REFNE-*at*, and REFNE-*a*.

The experimental results on REFNE-*a*, REFNE-*t*, and REFNE-*n* are shown in Table 6 to Table 11. For comparison, here we also include the results on REFNE-*at* that was reported in Section 4.2. Note that all the results are attained via 5-fold cross validation.

Table 6 shows that REFNE-*n*, i.e. REFNE without additional data and without ties broken, attains the best fidelity that is better than 91%. Table 6 also reveals that the use of additional data significantly reduces the fidelity of extracted rules because the fidelity of the rules extracted via REFNE-*at* and REFNE-*a* are significantly worse than those extracted via REFNE-*t* and REFNE-*n* respectively. However, Table 7 shows that the use of additional data significantly improves the generalization ability of the extracted rules because the generalization error of REFNE-*at* and REFNE-*a* is significantly smaller than that of REFNE-*t* and REFNE-*n* respectively. Moreover, Table 6 and Table 7 also reveal that breaking the ties is helpful in improving the generalization ability of the extracted rules with the cost of lowering the fidelity of those rules. Those observations justify our conjecture that the unsatisfying fidelity of the rules extracted via REFNE-*at* is caused by that the generalization ability of those rules is far better than that of the trained neural network ensemble.

Table 8 to Table 10 show that employing different configurations of REFNE does not significantly impact the conciseness of the extracted rules because the average, maximum, and minimum antecedents of an extracted single rule does not significantly vary. However, the conciseness of the extracted rule set is significantly impacted

Table 6
Fidelity of rules extracted via REFNE with different configurations

	balance	*voting*	*hepatitis*	*iris*	*credit-a*	*credit-g*	*average*
REFNE-t	90.80%	92.69%	87.50%	97.60%	87.26%	87.60%	90.58%
REFNE-n	93.60%	92.69%	87.50%	98.33%	87.26%	87.60%	91.16%
REFNE-at	87.88%	89.26%	84.50%	96.25%	84.13%	74.10%	86.02%
REFNE-a	88.10%	89.26%	84.50%	97.00%	84.13%	74.10%	86.18%

Table 7
Comparison of generalization error of rules extracted via REFNE with different configurations

data set	*REFNE-t*	*REFNE-n*	*REFNE-at*	*REFNE-a*
balance	10.16%	11.92%	6.72%	8.68%
voting	3.43%	3.43%	0.00%	0.00%
hepatitis	5.00%	5.00%	2.50%	2.50%
iris	2.67%	3.33%	2.67%	3.33%
credit-a	2.46%	2.46%	0.87%	0.87%
credit-g	13.40%	13.40%	3.80%	3.80%
average	6.19%	6.59%	2.76%	3.20%

Table 8
Comparison of the number of rules extracted via REFNE with different configurations

data set	*REFNE-t*	*REFNE-n*	*REFNE-at*	*REFNE-a*
balance	19.2	20.6	23.4	26.8
voting	8.4	8.4	30.0	30.0
hepatitis	12.6	12.6	36.0	36.0
iris	9.6	9.6	17.4	18.6
credit-a	33.6	33.6	39.0	39.0
credit-g	41.0	41.0	39.4	39.4
average	20.7	21.0	30.9	31.6

Table 9
Comparison of the average number of antecedents of a single rule extracted via REFNE with different configurations

data set	*REFNE-t*	*REFNE-n*	*REFNE-at*	*REFNE-a*
balance	1.45	1.49	1.18	1.28
voting	1.26	1.26	2.08	2.08
hepatitis	1.45	1.45	2.39	2.39
iris	1.56	1.56	1.59	1.68
credit-a	2.09	2.09	2.14	2.14
credit-g	1.96	1.96	1.79	1.79
average	1.63	1.64	1.86	1.89

by employing different configurations of REFNE. The most concise rule set is the one extracted via REFNE-*t* while the most tedious rule set is the one extracted via REFNE-*a*. In general, the smaller the number of instances used for rule learning, the more concise the extracted rule set is.

Based on those observations, we believe that REFNE-*at* can extract rules with the best generalization ability while REFNE-*t* can extract rules with the best fidelity and the best conciseness.

5. Discussion

Following observation on the experimental results presented in Section 4.3 is very interesting. That is, if we rank REFNE with different configurations according to the fidelity of the ex-

Table 10
Comparison of the maximum number of antecedents of a single rule extracted via REFNE with different configurations

data set	*REFNE-t*	*REFNE-n*	*REFNE-at*	*REFNE-a*
balance	3.0	3.0	2.0	2.0
voting	1.6	1.6	3.0	3.0
hepatitis	1.8	1.8	3.0	3.0
iris	2.4	2.4	2.4	2.2
credit-a	3.0	3.0	3.0	3.0
credit-g	3.0	3.0	3.0	3.0
average	2.5	2.5	2.7	2.7

Table 11
Comparison of the minimum number of antecedents of a single rule extracted via REFNE with different configurations

data set	*REFNE-t*	*REFNE-n*	*REFNE-at*	*REFNE-a*
balance	1.0	1.0	1.0	1.0
voting	1.0	1.0	1.0	1.0
hepatitis	1.0	1.0	1.0	1.0
iris	1.0	1.0	1.0	1.0
credit-a	1.0	1.0	1.0	1.0
credit-g	1.0	1.0	1.0	1.0
average	1.0	1.0	1.0	1.0

tracted rules, then the descending order should be REFNE-*n*, REFNE-*t*, REFNE-*a*, and REFNE-*at* where REFNE-*n* is the best and REFNE-*at* is the worst; but if we rank them according to the generalization ability of the extracted rules, then the order should be the reverse, i.e. REFNE-*at*, REFNE-*a*, REFNE-*t*, and REFNE-*n*, where REFNE-*at* is the best and REFNE-*n* is the worst.

We believe that the observation can be understood or explained as follows. On one hand, when ties appear, the ensembles do not exactly know which class should be predicted. With ties broken, such kind of ambiguity caused by ties may be precluded from the extracted rules so that the generalization ability of the rules may be improved. But since such kind of ambiguity may not be captured by the extracted rules, the fidelity of the rules may be worsened. On the other hand, since the resource for rule learning may be increased by utilizing additional data generated from the trained ensembles, the generalization ability of the extracted rules has great chances to be improved. But at this time the fidelity of the extracted rules may be worsened because the rules may be far more accurate than the ensembles.

In recent years, many approaches for rule extraction from trained neural networks have been developed. As reviewed in Section 2.2, those approaches could be roughly categorized into two classes, i.e. function-analysis-based or architecture-based approaches. The function-analysis-based approaches extract rules via regarding the trained networks as entities, which can be easily modified for extracting rules from trained ensembles through regarding the ensembles instead of the networks as entities. In fact, REFNE is derived from a function-analysis-based approach, i.e. STARE [47], for extracting rules from trained neural networks.

The architecture-analysis-based approaches extract rules via disassembling the architectures of the trained neural networks, which is relatively difficult to be modified for extracting rules from trained ensembles. The reason is that even rules could be extracted from each individual network, it is still difficult to unite them into a consistent rule set that could well explain the ability of the ensemble because simply gathering them together can only result in a mussy hodgepodge of rules.

However, no matter where the approaches for rule extraction from neural network ensembles are derived from, they must pay attention to some specific characteristics of ensembles, e.g. ambiguity in prediction caused by ties.

6. Conclusions

In this paper, an approach named REFNE that extracts symbolic rules from trained neural network ensembles that perform classification tasks is proposed. With different configurations, REFNE can extract rules with strong generalization ability or with high fidelity and conciseness. Since it can generate rules with impressive generalization ability, it can also be used as an alternative to present rule learning approaches.

In the near future, we want to develop some techniques to measure the strength of interactions among continuous attributes and incorporate some mechanisms for dealing with strong interactions in REFNE. Moreover, we want to design other kinds of approaches for rule extraction from trained neural network ensembles. Furthermore, we hope to explore the chances of applying neural network ensembles to data mining tasks with the help of extracting rules from trained ensembles.

Acknowledgements

The comments and suggestions from the anonymous reviewers greatly improve this paper. The National Natural Science Foundation of P.R.China and the Natural Science Foundation of Jiangsu Province, P.R.China, supported this research.

References

[1] R. Andrews, J. Diederich, and A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems* **8** (1995), 373-389.

[2] E. B. Baum and D. Haussler, What size net gives valid generalization? *Neural Computation* **1** (1989), 151-160.

[3] J. M. Benitez, J. L. Castro, and I. Requena, Are artificial neural networks black boxes? *IEEE Transactions on Neural Networks* **8** (1997): 1156-1164.

[4] C. Blake, E. Keogh, and C. J. Merz, UCI repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.htm], Department of Information and Computer Science, University of California, Irvine, CA, 1998.

[5] L. Breiman, Bagging predictors, *Machine Learning* **24** (1996): 123-140.

[6] K. J. Cherkauer, Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks, in: *Proceedings of the 13th AAAI Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, Portland, OR, 1996, pp.15-21.

[7] M. W. Craven and J. W. Shavlik, Using sampling and queries to extract rules from trained neural networks, in: *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, NJ, 1994, pp.37-45.

[8] B. Efron and R. Tibshirani, *An introduction to the bootstrap*, Chapman & Hall, New York, 1993.

[9] Y. Freund, Boosting a weak algorithm by majority, *Information and Computation* **121** (1995): 256-285.

[10] Y. Freund and R. E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, *Journal of Computer and System Sciences* **55** (1997): 119-139.

[11] L. Fu, Rule learning by searching on adapted nets, in: *Proceedings of the 9th National Conference on Artificial Intelligence*, Anaheim, CA, 1991, pp.590-595.

[12] S. I. Gallant, Connectionist expert systems, *Communications of the ACM* **31** (1988): 152-169.

[13] A. S. D. Garcez, K. Broda, and D. M. Gabbay, Symbolic knowledge extraction from trained neural networks: A sound approach, *Artificial Intelligence* **125** (2001): 155-207.

[14] S. Gutta and H. Wechsler, Face recognition using hybrid classifier systems, in: *Proceedings of the IEEE International Conference on Neural Networks*, Washington, DC, 1996, pp.1017-1022.

[15] J. Hampshire and A. Waibel, A novel objective function for improved phoneme recognition using time-delay neural networks, *IEEE Transactions on Neural Networks* **1** (1990): 216-228.

[16] L. K. Hansen and P. Salamon, Neural network ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12** (1990): 993-1001.

[17] L. K. Hansen, L. Liisberg, and P. Salamon, Ensemble methods for handwritten digit recognition, in: *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, Copenhagen, Denmark, 1992, pp. 333-342.

[18] F. J. Huang, Z.-H. Zhou, H.-J. Zhang, and T. Chen, Pose invariant face recognition, in: *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, 2000, pp.245-250.

[19] M. Ishikawa, Rule extraction by successive regularization, *Neural Networks* **13** (2000): 1171-1183.

[20] D. Jimenez, Dynamically weighted ensemble neural networks for classification, in: *Proceedings of the IEEE International Joint Conference on Neural Networks*, Anchorage, AK, 1998, vol.1, pp.753-756.

[21] R. Kerber, Chi-Merge: Discretization of numeric attributes, in: *Proceedings of the 9th National Conference on Artificial Intelligence*, San Jose, CA, 1992, pp.123-128.

[22] R. Krishnan, A systematic method for decompositional rule extraction from neural networks, in: *Proceedings of the NIPS96 Workshop on Rule Extraction from Trained Artificial Neural Networks*, Queensland, Australia, 1996, pp.38-45.

[23] A. Krogh and J. Vedelsby, Neural network ensembles, cross validation, and active learning, in: *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, eds, MIT Press, Cambridge, MA, 1995, pp.231-238.

[24] H. Liu and R. Setiono, Feature selection via discretization of numeric attributes, *IEEE Transactions on Knowledge and Data Engineering* **9** (1997): 642-645.

[25] R. Maclin and J. W. Shavlik, Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks, in: *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995, pp.524-530.

[26] J. Mao, A case study on bagging, boosting and basic ensembles of neural networks for OCR, in: *Proceedings of the IEEE International Joint Conference on Neural Networks*, Anchorage, AK, 1998, vol.3, pp.1828-1833.

[27] D. W. Opitz and J. W. Shavlik, Actively searching for an effective neural network ensemble, *Connection Science* **8** (1996): 337-353.

[28] D. W. Opitz and J. W. Shavlik, Generating accurate and diverse members of a neural network ensemble, in: *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. Mozer, and M. Hasselmo, eds, MIT Press, Cambridge, MA, 1996, pp.535-541.

[29] M. P. Perrone and L. N. Cooper, When networks disagree: Ensemble method for neural networks, in: *Artificial Neural Networks for Speech and Vision*, R. J. Mammone, ed, Chapman & Hall, New York, 1993, pp.126-142.

[30] J. R. Quinlan, *C4.5: Programs for machine learning*, Morgan Kaufmann, San Mateo, CA, 1993.

[31] D. Rumelhart, G. Hinton, and R. Williams, Learning representations by backpropagating errors, *Nature* **323** (1986): 318-362.

[32] K. Saito and R. Nakano, Rule extraction from facts and neural networks, in: *Proceedings of the International Neural Network Conference*, San Diego, CA, 1990, pp.379-382.

[33] R. E. Schapire, The strength of weak learnability, *Machine Learning* **5** (1990): 197-227.

[34] G. P. J. Schmitz, C. Aldrich, and F. S. Gouws, ANN-DT: An algorithm for extraction of decision trees from artificial neural networks, *IEEE Transactions on Neural Networks* **10** (1999): 1392-1401.

[35] S. Sestito and T. Dillon, Knowledge acquisition of conjunctive rules using multilayered neural networks, *International Journal of Intelligent Systems* **8** (1993): 779-805.

[36] R. Setiono, Extracting rules from neural networks by pruning and hidden-unit splitting, *Neural Computation* **9** (1997): 205-225.

[37] R. Setiono and W. K. Loew, FERNN: An algorithm for fast extraction of rules from neural networks, *Applied Intelligence* **12** (2000): 15-25.

[38] D. Sharkey, ed, *Combined artificial neural nets: Ensemble and modular multi-net systems*, Springer-Verlag, London, 1999.

[39] Y. Shimshoni and N. Intrator, Classification of seismic signals by integrating ensembles of neural networks, *IEEE Transactions on Signal Processing* **46** (1998): 1194-1201.

[40] K. Tanaka, M. Nii, and H. Ishibuchi, Learning from linguistic rules and rule extraction for function approximation by neural networks, in: *Lecture Notes in Artificial Intelligence 1585*, B. McKay, X. Yao, C. S. Newton, J.-H. Kim, and T. Furuhashi, eds, Springer-Verlag, Berlin, 1999, pp.317-324.

[41] S. Thrun, Extracting rules from artificial neural networks with distributed representations, in: *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, eds, MIT Press, Cambridge, MA, 1995, pp.505-512.

[42] G. Towell and J. W. Shavlik, The extraction of refined rules from knowledge-based neural networks, *Machine Learning* **13** (1993): 71-101.

[43] N. Ueda, Optimal linear combination of neural networks for improving classification performance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22** (2000): 207-215.

[44] V. Vapnik, *The nature of statistical learning theory*, Springer-Verlag, New York, 1995.

[45] D. H. Wolpert, Stacked generalization, *Neural Networks* **5** (1992): 241-259.

[46] X. Yao and Y. Liu, Making use of population information in evolutionary artificial neural networks, *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics* **28** (1998): 417-425.

[47] Z.-H. Zhou, S.-F. Chen, and Z.-Q. Chen, A statistics based approach for extracting priority rules from trained neural networks, in: *Proceedings of the IEEE-INNS-ENNS International Conference on Neural Networks*, Como, Italy, 2000, vol.3, pp.401-406.

[48] Z.-H. Zhou, Y. Jiang, Y.-B. Yang, and S.-F. Chen, Lung cancer cell identification based on artificial neural network ensembles, *Artificial Intelligence in Medicine* **24** (2002): 25-36.

[49] Z.-H. Zhou, J.-X. Wu, Y. Jiang, and S.-F. Chen, Genetic algorithm based selective neural network ensemble, in: *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001, vol.2, pp.797-802.