Enhancing Classifiers through Neural Network Ensembles

Alexandru Onaci, Camelia Vidrighin, Mihai Cuibus, Rodica Potolea
Technical University of Cluj-Napoca
Camelia.Vidrighin@cs.utcluj.ro

Abstract

Artificial neural networks are known to have strong generalization abilities, but they entirely lack comprehensibility, due to their connectionist nature. Neural network ensembles augment this characteristic, making them less appealing in domains where comprehensibility is as important as accuracy. This paper presents the implementation of a new system based on a method for combining ensembles of neural networks with symbolic learners. The focus is on enhancing the symbolic classifiers by using a neural network ensemble as a pre-processing step for them. The results obtained during the evaluations on the new system have confirmed that the approach is suitable for enhancing the performance of symbolic classifiers.

1. Introduction

Machine learning techniques can be split into two major categories, the symbolic and the connectionist learning approaches. The symbolic techniques involve mechanisms for representing knowledge in a structured way, such as rules that can be easily comprehended by the user. In connectionist learning techniques the knowledge learned is not transparent to the user, but represented throughout a connected network of units.

A representative example of connectionist methods is that of the neural networks. During the last few years their utility has been reconfirmed in various categories of problems, such as function approximation – including time series prediction and modeling, classification – including pattern and sequence recognition, or data processing – filtering, clustering, blind source separation and compression. Applications areas include pattern and sequence recognition, medical diagnosis, stock-market prediction, credit assignment and machine monitor and control.

Neural networks work by taking a set of inputs and creating hidden interdependencies between them in order to produce an output. Therefore they can be applied in any situation where a relation between the input and the output exists, even if the relation is not obvious and cannot be easily represented as production rules.

There are two main inconveniences related to artificial neural networks, namely their *lack of transparency* and *instability*. The lack of transparency is due to the fact that they learn the separation hyper-planes by means of weights and activation functions, which encode the information inside the network architecture. This has lead to a reluctance of employing them in some application domains.

Another reason for their reduced usage in real world problems is induced by their instability. That is, small changes in the training data may result in very different models, thus affecting the performance on unseen data [12].

In order to solve this stability problem artificial neural network ensembles have been introduced. In a neural network ensemble several networks are trained to approximate the same function. The prediction of the ensemble is the combination of the results obtained by each individual network. It has been shown [6] that the generalization ability and the stability of an artificial neural network ensemble are better than that of a single network. However, the network adds a new layer of complexity and its comprehensibility is inferior to that of a single neural network.

In order to address this problem, Zhou [14] introduces C4.5 Rule PANE (C4.5 Rule Preceded by Artificial Neural Ensemble), a system which uses a neural network ensemble as a preprocess step for a rule induction algorithm. The main idea is to try and extract rules while keeping the accuracy as close as possible to that of the neural network ensemble. The rule induction algorithm used is C4.5 Rule. It is derived from Quinlan's C4.5 decision tree [8]. Zhou argues that such an approach is particularly beneficial in application areas where the comprehensibility is as

important as the generalization ability. One such domain is that of medical diagnosis and prognosis, where the large amount of patient data can be processed by classification systems in order to provide valuable support to the diagnosis process. Besides the accuracy needed in such cases, the transparency of the prediction process is also of great importance, because the physician needs to validate its assumptions against the knowledge learned by the system. To make that possible, the system must present its knowledge in structured form.

This paper adopts an approach similar to that found in [14] to construct a system which uses a neural network ensemble as a pre-processing step for a symbolic classifier. The evaluation is performed on several symbolic classifiers. Also, several methods for building the ensemble neural networks are explored.

The rest of the paper is organized as follows. Section 2 presents the theoretical concepts on which the system described in Section 3 is built. Section 4 presents the evaluations performed. The results are presented in Section 5, and the conclusions are drawn in section 6.

2. Theoretical Background

2.1 Artificial Neural Network Ensemble

In [6] it is shown that the performance of systems based on neural networks can be further improved by training several networks and combining their predictions. The process of building an ensemble consists of two steps. First, each network in the ensemble is trained individually, in slightly different conditions. Second, when making a prediction, the outputs of the individual components are combined.

In order for the ensemble to perform better than the individual networks, each component must have a high degree of accuracy, and the ensemble must have a certain degree of diversity among its members. Each network component may be good at classifying some type of instances but weaker at classifying others. This kind of diversity allows for a higher variety of instances to be classified accurately by the ensemble.

The most prominent approaches for constructing the ensemble are Breiman's bagging and Schapire's boosting. Bagging [1] generates each component network by taking a bootstrap sample (with replacement) from the original dataset S. The sampled dataset S_B has the same size as the original dataset. Therefore some instances from S may be missing, while other may appear multiple times. Instances that are not in S_B can be used as a validation set for the component network that is being constructed.

Boosting [5] is an iterative process. During each iteration, or boosting phase, an individual model is built by varying the weights of the instances used for training. The algorithm re-weights the training set at the end of each boosting phase: the weights of the instances that have been incorrectly classified in the current phase are increased, while those of the correctly classified instances are decreased.

Output combination is typically achieved through *averaging* or *voting*. Averaging is employed in prediction problems, while voting is appropriate for classification problems.

2.2 Symbolic Classifiers

The category of symbolic classifiers encompasses those algorithms which present knowledge in a structured, comprehensible form. Examples include decision trees, rule based systems or naïve Bayes learners. This section briefly describes the algorithms we have used in building our system.

The C4.5 algorithm is an extension to the original ID3 algorithm. ID3 uses a top-down inductive approach, selecting at each step the attribute that best classifies the data. The measure used to quantify attribute strength is "borrowed" from information theory and it called the *information gain*. It measures how well an attribute separates the training data with respect to the class they have.

Developed by Eibe and Witten, PART [4] is an algorithm which infers rules by repeatedly generating partial decision trees. It works by obtaining a rule from the "best" leaf of the partial tree obtained at each step. The algorithm combines two important paradigms for rule learning, namely creating rules from decision trees and using the separate-and-conquer technique.

AdaBoost.M1 is the most popular and historically most significant boosting algorithm. Introduced by Freund and Schapire [5], it was the first algorithm to employ the boosting approach in building classifier ensembles. An interesting remark related to AdaBoost.M1 is that it has been mathematically proved that the error on the un-weighted training examples approaches zero exponentially with an increasing number of boosting steps. Still, its success is mostly due to its outstanding performance in practice. Although conventionally AdaBoost.M1 is not a symbolic classifier, we have included it in our work to study the degree to which the neural network ensemble can improve the performance of a simple classifier.

2.3 The Basic Algorithm

The main idea of the current approach is to use the neural network ensemble as a pre-processing step in training the symbolic classifier. That is we use the ensemble to re-label the data which will provide the input of the symbolic classifier.

The algorithm flow can be divided into the following steps:

1. Train the neural network ensemble using the available training data.
2. Use the trained ensemble to classify the available data.
3. [*optional*] Generate random data and use the trained ensemble to classify it.
4. Run a symbolic learning algorithm on the data generated in steps 2 and 3

Because neural networks ensembles have good generalization ability, the output dataset of the ensemble may be better for classification than the original dataset, because some "bad ingredients" in the original dataset might have been removed (e.g. noise).

3. System Description

There are several particularities related to the current implementation. They concern both the neural network ensemble and the symbolic classifier.

Suppose we have a dataset $S = \{(v_1, c_1), (v_2,c_2)\ldots.(v_n,c_n)\}$, where v_i is the feature vector for the i-th instance and c_i is the class of that instance. From this dataset a certain percent of the instances will be used in training. Denote the training dataset with T. The remaining data will form the evaluation set, denoted by V.

In the first step an artificial neural network ensemble is trained using T. The method used in [14] is bagging with un-weighted majority vote. Our implementation employs both bagging and boosting.

In what follows, we shall denote the number of neural networks in the ensemble by E

The Bagging Approach

We have to obtain E bootstrap samples from T, denoted with B_1, B_2 … B_E. Each neural network in the ensemble is trained using one of the $B_1 \ldots B_E$ sets as its training set. Let N* denote the trained neural network ensemble containing E neural networks denoted by N_1, $N_2 \ldots N_E$. The output of N* is the class label that has received the most number of votes (majority voting technique).

The Boosting Approach

A boosting algorithm begins by setting the same weight for each instance in T. Then, a neural network B_1 is built using T. The weights of the instances that are correctly classified by B_1 are decreased, while those of the incorrectly classified instances are increased. Next a second neural network B_2 is built on the modified weighted dataset, focusing on instances with high weights. The algorithm continues until either the error rate of the neural network exceeds 50% or it is below a given threshold. The learning process can stop also when the maximum number of models in the ensemble is reached.

Once the neural network ensemble N* is built, using bagging or boosting, the feature vectors of T are passed through N* in order to obtain their final class labels – step two in the algorithm. Each feature vector in T, $f_i = (v_i, c_i)$, i=1..n is supplied to the trained ensemble N*, which predicts a class for it – c_i*. The predicted class could be the same as the original class c_i, but this is not guaranteed.

Using the above mentioned process, a new training dataset is obtained and denoted by T_1 that has the same number of instances as S, but newly created class labels for each of them. Every feature vector in T_1 has the form $f_i = (v_i, c_i^*)$ where i=1…n.

Because the neural network ensemble N* has a good generalization ability Zhou claims that this newly created dataset T_1 may be better for rule induction than T because some bad ingredients like noise, have been removed.

Using the trained ensemble N* new additional data can be created by randomly generating feature vectors and processing them using N* in order to obtain their class labels. This dataset is called the additional training dataset and is denoted by T_2. For every feature vector f_2 in T_2 we have $f_2 = (v_2, c_2)$. The feature vectors from T_2 are passed through the trained ensemble N* and a new class c_2* is obtained, thus making the features vectors in T_2 having the form $f_2 = (v_2, c_2^*)$. The size of this additional dataset is multiple of the original training set T. The exact size must be manually determined for each dataset used in testing, in order to obtain good performances.

The two training datasets T_1 and T_2 are then combined in a new dataset D which is used as the training dataset for a machine learning algorithm. The obtained results are validated using the evaluation set V.

4. Experimental Work

The experiments performed in this paper evaluate the new system on several machine learning classical benchmarks. The primary purpose in devising the evaluation procedures was to check if the idea of preceding symbolic classifiers with neural network ensembles leads to the improvement of the initial learning algorithm. A second concern is associated with the idea formulated in [14], regarding the positive impact that random data can have on the learning capabilities of the symbolic classifier. The procedure presented there is very simple: data is generated randomly and is fed to the neural network ensemble, which predicts class labels for each feature vector. The newly obtained data is combined with existing, re-labeled training set and provided to the symbolic learner for training. In [14] it was concluded that this was a good technique for reducing the error of the symbolic classifier. Furthermore, the tests performed in [14] showed that adding a number of instances equal to three times the size of the original training set lead to the highest error reduction.

In order to answer the above questions, tests on five different datasets have been performed. The datasets are freely available at the UCI Machine Learning Repository. Four of the datasets are from the medical domain, and one is from the car manufacturer industry. Instances with missing values were removed from each dataset. Information about the datasets can be found in Table 1.

The employed symbolic learning algorithms are C4.5, PART and AdaBoost.M1. Although AdaBoost.M1 is not an actual symbolic classifier, it was included in our evaluation in order to better evaluate the impact of the neural network ensemble on the performance of the "simple" classifier.

For each test 80% of the available instances have been used to train the neural network, and the rest of 20% have been kept for evaluation. Results have been averaged over 100 runs.

The bootstrap sampling only uses about 63% of the available instances to generate new data, and thus the rest of the instances not used by bagging can be used as a validation set for each neural network in the ensemble. During each epoch in the training process of each neural network its generalization error is estimated on the validation dataset, and if the error does not change for a number of consecutive epochs the training process stops in order to avoid overfitting.

Experimentally we have observed that the optimum number of epochs is five. This value coincides with what was observed in [14].

Table 1 – Dataset Information

Dataset	Instances	Classes	Attributes
Liver disorder	345	2	6
Heart disease	303	5	13
Diabetes	768	2	8
Breast cancer	699	2	9
Cars	1728	4	6

Other general parameters related to the neural networks were obtained empirically, in a process which is generally known as fine tuning. The neural networks are standard feed-forward networks trained using the *back propagation* algorithm, with one hidden layer. During our experiments we have observed that the best results are obtained when the number of hidden units varied between 9 and 12, the learning rate was about 0.6 and the momentum varied between 0.2 and 0.3.

The neural network ensemble contains 11 members in the case of bagging, and may contain less than 11 members when boosting is performed (due to the nature of the approach). This value was also obtained experimentally. For bagging ensembles we used both weighted and un-weighted voting. Boosting employed weighted voting.

In order to validate the idea related to additional data we incrementally increased the size of the additional training dataset between 0% and 500% of the original dataset.

5. Results

The experimental results using bagging with un-weighted majority vote are shown in Table 2. We have used the following abbreviations: NE for the algorithm enhanced by the neural network ensemble and NNE for neural network ensemble alone. We have considered the tests with no additional data, such as to evaluate the improvement produced by the neural network ensemble. As it can be observed from Table 2, the neural network ensemble yields the lowest error rates on all datasets. Significant improvements can be observed for C4.5 and PART when the neural network is used as a pre-processing step, on all datasets. An interesting result is the fact that the improvement in the case of AdaBoost.M1 is no better than the improvement obtained for the other classifiers. On average, PART achieves the highest relative error reduction (12%), with an impressive 32% on the Cars dataset.

Table 2 – Error rates obtained with bagging using un-weighted majority vote

Algorithm	*Bupa (%)*	*Cleveland (%)*	*Pima (%)*	*Wisconsin (%)*	*Cars (%)*
C4.5	39.38	47.35	26.58	5.81	8.85
C4.5 NE	36.70	45.56	24.88	5.11	7.20
PART	38.45	48.83	27.07	4.32	4.87
PART NE	36.16	45.11	25.10	4.00	3.31
AdaBoost	39.44	47.46	27.22	3.48	4.87
AdaBoostNE	36.02	44.42	23.94	3.35	4.86
NNE	35.25	42.02	23.73	3.31	0.66

Table 3 – Error rates obtained with bagging using weighted majority vote

Algorithm	*Bupa (%)*	*Cleveland (%)*	*Pima (%)*	*Wisconsin (%)*	*Cars (%)*
C4.5	39.69	48.13	26.67	5.20	8.85
C4.5 NE	36.55	45.39	25.05	4.72	8.77
PART	38.36	47.61	27.37	5.94	4.99
PART NE	36.20	45.93	25.32	5.91	4.91
AdaBoost	38.55	49.56	25.65	4.66	5.91
AdaBoostNE	34.30	49.50	24.32	4.55	5.59
NNE	34.85	42.18	24.07	3.58	0.73

The experimental results for a weighted majority vote are shown in Table 3. Again, no additional data was used. Once more, the neural network outperforms all other algorithms.

The results are quite similar to that of the un-weighted majority vote. The improvement for the Cars dataset is not as significant as before. A reason why such similar results have been obtained can be found in the nature of neural networks: they are known to be powerful classifiers, with very low error rates.

Table 4 – Error rates obtained with boosting

Algorithm	*Bupa (%)*	*Cleveland (%)*	*Pima (%)*	*Wisconsin (%)*	*Cars (%)*
C4.5	40.36	47.25	26.68	5.45	8.82
C4.5 NE	38.30	44.70	25.69	4.66	20.29
PART	38.27	48.23	27.20	4.60	5.04
PART NE	36.38	43.77	25.30	4.07	16.13
AdaBoost	38.26	48.36	25.49	5.05	5.24
AdaBoostNE	35.88	45.89	24.96	3.76	29.34
NNE	35.16	42.60	24.62	3.77	2.35

Table 4 shows the results obtained using boosting as the method for constructing the neural network ensemble. The overall relative reduction in the error rate is about 8%. For the Cars dataset, the neural network ensemble does not improve the performance of the symbolic classifier. This happens because the boosting stops when the error exceeds 50% or when it goes below a given threshold (in this case it was 1%). Because a single neural network has a very low error rate on this dataset (1-2%) the number of networks created through boosting is usually very small (1 or 2). This has a large impact on the diversity of the ensemble, affecting its generalization ability. This phenomenon cannot be observed on the other datasets, because the error of the single classifiers does not converge to zero so quickly. An important result would be if we could develop a way to set the threshold for the error rate automatically. Clearly this threshold is dependent on the dataset and its value has an important role in the performance of the algorithm. Finding this value automatically is a problem of future interest.

Therefore a weighted voting scheme does not produce significant changes to the ensemble, when compared to an un-weighted scheme.

During the experiments we have observed that the time complexity reaches the highest value when the datasets contains only nominal attributes (Cars dataset). Also, when the dataset contains entirely numerical attributes, the time complexity is low and the improvement in the generalization ability of the algorithm is much better, because decision trees algorithms have problems dealing with numeric attributes.

Table 5 – Relative error reduction obtained with bagging using un-weighted majority vote

Algorithm	*Bupa (%)*	*Cleveland (%)*	*Pima (%)*	*Wisconsin (%)*	*Cars (%)*
C4.5	6	3	6	12	18
PART	6	7	7	7	32
AdaBoost	8	6	12	3	1

Table 6 – Relative error reduction obtained with bagging using weighted majority vote

Algorithm	*Bupa (%)*	*Cleveland (%)*	*Pima (%)*	*Wisconsin (%)*	*Cars (%)*
C4.5	8	5	6	9	1
PART	5	3	7	1	1
AdaBoost	11	1	5	2	5

Table 7 – Relative error reduction obtained with boosting

Algorithm	*Bupa (%)*	*Cleveland (%)*	*Pima (%)*	*Wisconsin (%)*	*Cars (%)*
C4.5	5	5	4	14	-130
PART	5	9	7	12	-220
AdaBoost	6	5	2	25	-460

In order to gain statistical insight into the results, the relative error reduction has been calculated. We have used the following formula for C4.5 (same for PART and AdaBoost.M1):

$$Err\ reduction = \frac{C4.5\ Error - C4.5NNE\ Error}{C4.5\ Error} \quad (1)$$

The values obtained are presented in Tables 5-7. The best results are obtained when using bagging with an un-weighted voting method. In this case, the performance increases by 9%, with a peak value of 32% on the Cars dataset.

The results obtained when using bagging and a weighted voting method are quite similar to those obtained using the un-weighted voting scheme, except for the Cars dataset.

The boosting approach of the algorithm creates a serious drop in performance on the Cars dataset. This drop is expected, since it has previously been observed when comparing error rates also. This behaviour is related to how the boosting algorithm works. The relative error reduction proves once again that setting the correct lower error threshold is a critical step towards performance improvement.

The second batch of tests was related to the assumption that the generalization ability of the output classifiers can be further improved by using additional training data. The results obtained for C4.5 using bagging with un-weighted vote are shown in figures 1-5.

The values on the x axis represent the size of the randomly generated data, being a multiple of the initial set size.

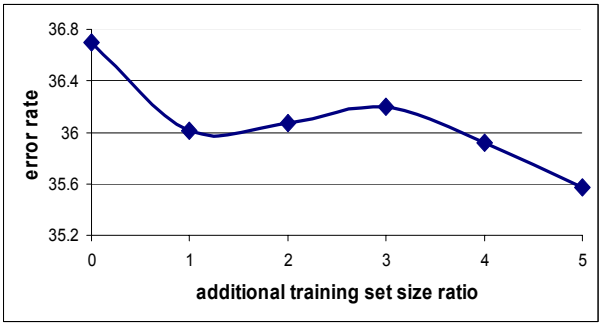

Figure 1 – Error variation for different sizes of the additional data on Bupa

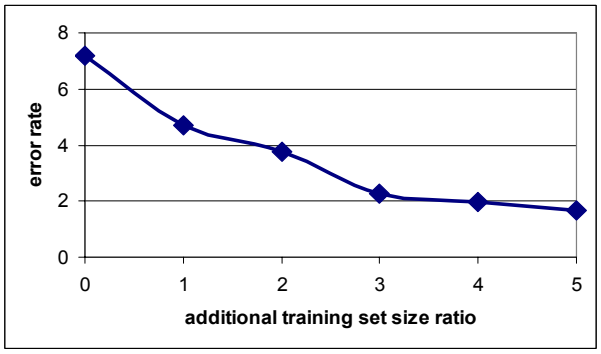

Figure 2 – Error variation for different sizes of the additional data on Cars

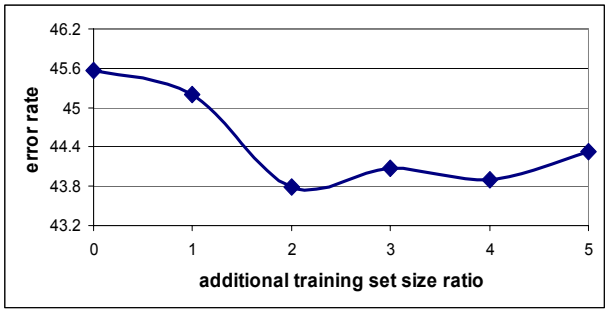

Figure 3 - Error variation for different sizes of the additional data on Cleveland

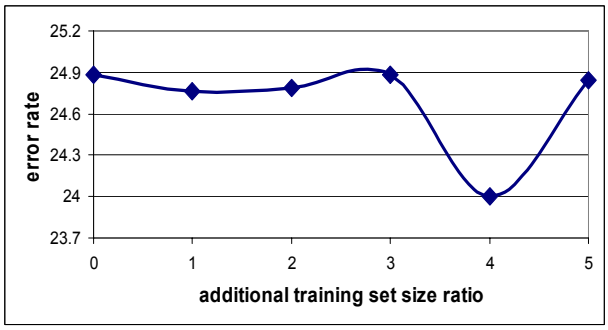

Figure 4 – Error variation for different sizes of the additional data on Pima

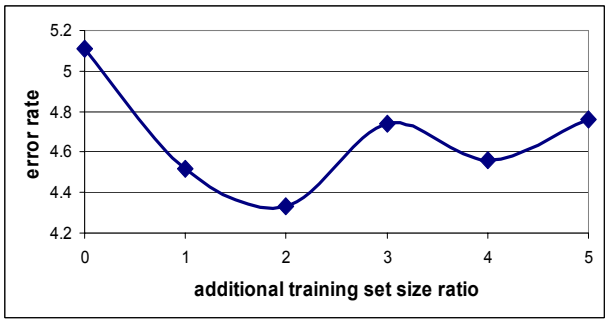

Figure 5 – Error variation for different sizes of the additional data on Wisconsin

A first remark is that additional training data can reduce the error rate of the symbolic classifier. The value of the ratio of additional data for which the best performance is obtained depends on the dataset. We observed that increasing the data with an amount equal to 4 or 5 times the initial dataset size usually leads to a significant relative error reduction (up to 70% for the Cars dataset). However, increasing this ratio further may not lead to improved performance.

6. Conclusions and future work

Neural networks and neural network ensembles are known to perform well in classification tasks, but their usage in certain application areas has been restricted due to their lack of transparency in the prediction process. By utilizing a neural network ensemble as a pre-process step for a symbolic algorithm, the black-box property of the neural network is eliminated.

This approach is investigated in the current paper. We develop a system which uses a neural network ensemble as a pre-processing step for a symbolic learning algorithm. The system is intended to be beneficial in domains where the generalization ability is as important as the comprehensibility of the decision process.

We have tackled several techniques for building the ensemble: bagging with un-weighted vote, bagging with weighted vote, and boosting. As symbolic output classifier we used the C4.5 decision tree and the PART rule learner. We have also included AdaBoost.M1 in our evaluation to study the impact of the ensemble neural network on one of the most robust classifiers.

Evaluations were conducted in order to investigate the performance variation produced by the neural network ensemble on the output classifiers. Also, tests have been carried out to study the impact of additional data on the performance of the single classifiers.

The results obtained have shown that the current implementation can be a powerful tool for improving the performance of several machine learning algorithms while keeping the comprehensibility of their outputs. This conclusion is supported by the overall 8% relative reduction of the error. Also, improvements have been observed in the generalization ability of the output classifiers when additional data was used. Establishing the ratio of added data is still an open problem, and a problem of interest to us.

References

[1] L. Breiman, "Bagging Predictors", *Machine Learning*, 1996.

[2] P. Cunningham, J. Carney, and S. Jacob, "Stability problems with artificial neural networks and the ensemble solution," *Artificial Intelligence in Medicine*, vol. 20, no.3, pp.217-225, 2000.

[3] P. Domingos, "Knowledge Discovery via Multiple Models", *Intelligent Data Analysis*, 2, 187-202, 1998.

[4] E. Frank and I. Witten. "Generating Accurate Rule Sets Without Global Optimization", *Machine Learning:*

Proceedings of the Fifteenth International Conference, Morgan Kaufmann Publishers, San Francisco, 1998.

[5] Y. Freund and R. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting", *Journal of Computer & System Sciences*, 1997.

[6] L. K. Hansen and P. Salamon. "Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.12, no.10, pp.993-1001,1990.

[7] J. Mao, "A case study on bagging, boosting and basic ensembles of neural networks for OCR," in Proc. IEEE Int. J. Conf. Neural Networks, vol.3, Anchorage, AK, 1998, pp.1828-1833.

[8] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[9] J.R. Quinlan, "Comparing Connectionist and Symbolic Learning Methods", *In Computational Learning Theory and Natural Learning, R. Rivest* 1994, 445-456.

[10] R. Setiono, "Extracting rules from pruned neural networks for breast cancer diagnosis," *Artificial Intelligence in Medicine*, vol.8, no.1, pp.37-51, 1996.

[11] D. Sharkey, Ed. Combining Artificial Neural Nets: Ensemble and Modular Multi-net Systems. London: Springer-Verlag, 1999.

[12] R. Wall and P. Cunningham, "Exploring the potential for rule extraction from ensembles of neural networks", *11th Irish Conference on Artificial Intelligence &Cognitive Science,*2000.

[13] R. Wall, P. Cunningham and P. Walsh, "Explaining predictions form a neural network ensemble one at a time", *Lecture Notes in Computer Science*, 2000.

[14] Z.H. Zhou and Y. Jiang, "Medical Diagnosis with C4.5 Rule Preceded by Artificial Neural Network Ensemble', *IEEE Transactions on Information Technology in Biomedicine,* 2002.