A Constructive Algorithm for Neural Network Ensemble: Dynamical Change in Input Space

M. A. H. Akhand, Md. Monirul Islam* and K. Murase
Dept. of Human and Artificial Intelligent Systems, University of Fukui, Japan
* Dept. of Computer Science and Engineering, BUET, Dhaka, Bangladesh
Email: akhand@synapse.his.fukui-u.ac.jp

Abstract—**This paper presents a constructive algorithm for neural network ensemble creation using dynamical change in input space (DCIS) technique. DCIS is an automatic ensemble creation method where next component network is added in ensemble by training with previously misclassified patterns. To improve performance of this new network in the ensemble, trains with full training set but all output node values of misclassified patterns are considered as zero. DCIS creates optimized ensemble with lower computational cost and better generalization ability with respect to previously proposed hand-made ensemble creation methods such as the bagging and the boosting. It uses rather simple user-defined parameters that are the number of partial training epochs and hidden nodes per network. Results with benchmark problems exhibit better or competitive performance.**

Index Terms—**Generalization, constructive approach and negative correlation learning.**

I. INTRODUCTION

Neural Network Ensemble (NNE) is a collection of a finite number of neural networks that are trained for the same task [4]. The final output is the combination of participated network's output. NNE is known to improve classification and generalization ability [3], [4]. It is clear, however, there is no advantage to combining networks that exhibit identical generalization ability [4]. So, component networks should be diverged to compensate the failure of one network by others. On the other hand, if component networks show very bad generalization ability with diversity, performance of NNE will decrease. Thus there is a trade-off between diversity and generalization.

There are various ways we can produce diverse networks. Those include, varying the set of initial random weights, varying the topology, varying the algorithm employed, and varying the training data [4]. It is argued that networks trained on different training sets are more likely to show diversity than networks trained on the same training set from the starting point of different initial conditions, or with different numbers of hidden nodes, or using a different algorithm [5]. Data sampling or variation in training data is the common, effective and verified technique for NNE creation [8], [9], [10].

Another important issue of any NNE is how to train individual networks, since the training strategy influences the diversity and the accuracy of individual networks in the NNE [1]. There are three major approaches to train individual networks, i.e., independent training, sequential training, and simultaneous training [1]. In independent training, each individual network in NNE is trained independently to minimize the error between the target and its output. One major disadvantage of this approach is the lack of interaction among individual networks during training [2]. In sequential training, individual networks in NNE are trained one after another not only to minimize the error between the targets and their outputs, but also to decorrelate their errors from previously trained networks. In this approach, the connection weights of previously trained networks are frozen when the current network is being trained. As a result, training an individual network in the NNE cannot affect the previously trained networks. The errors of individual networks are not necessarily negatively correlated [1]. But it is known that negative correlation among individual networks in NNE is beneficial for improving NNE's performance [1]. Recently, Liu and Yao [2] proposed a simultaneous training method of individual networks in NNE with negative correlation. They introduced a correlation penalty term in error function so that individual network in NNE can be trained simultaneously and interactively. The advantage of this approach is that it can produce biased individual networks whose errors tend to be negatively correlated [2]. One disadvantage of this approach is that all individual networks in an NNE are concerned with the whole NNE error. This may cause competitions among individual networks in the NNE [1].

In DCIS, we therefore presented a constructive algorithm for NNE creation using a hybrid algorithm of sequential and simultaneous training. To add a network in the ensemble we used a simple technique; add a network that is partially trained by previously misclassified patterns only. Then we partially trained this new network in sequential fashion while changing input space that means; training is performed with full training set but assuming output node values equal to zero for the patterns that are currently misclassified by this network. Again we simultaneously trained the whole NNE via negative correlation [2].

The rest of this paper is organized as follows. Section II discusses some popular NNE creation methods and the difference with the proposed method. Section III describes the proposed method in details. Section IV presents experimental results of this new method. Finally, Section V concludes the paper with a brief summary and a few remarks.

II. METHODS OF NNE CREATION

Two major issues of NNE are, to create component networks and to combine the output of component networks. A lot of works have been done within last decade on NNE creation [6], [8], [9], [10], [11]. Among them, the most popular methods are the bagging [9] and the boosting [8], [10].

Both bagging [9] and boosting [10] manipulate the training data in order to generate different component networks. The bagging produces replicate training sets by sampling from the training instances. The boosting uses all instances at each repetition, but maintains a weight for each instance and weight is varied to focus on different instances by different networks [8]. In both cases, voting is used to combine output of component networks to get NNE's output [8]. In the bagging each component network has the same vote, while boosting assigns different voting strengths to component networks on the basis of their accuracy [8]. But both are manual NNE creation method and use probabilistic method to produce the training set for the next component network [3]. So these methods require checking all previous networks with mathematical calculation.

Recently Md. Monirul Islam et al. [1] proposed an automatic NNE creation algorithm, called CNNE. That is the first automatic NNE creation algorithm as they claimed. CNNE changes the architecture of NNE at the training time with adding hidden nodes and networks. Instead of training data sampling procedure, it trains all component networks simultaneously with full training set via negative correlation. It starts training with two networks with single hidden node per network. First it tries to minimize the error by adding hidden node according to defined criteria, and then by adding new network with single hidden node. After adding several networks, if no improvement is found, it stops training according to a halting criteria. In that case, it starts with two identical networks; initial random weights are only difference between them. So, both networks will try to form identical functions. Another point is that it starts with single hidden node per network for all the problems but some problems required more hidden nodes, especially when number of output classes increases. So computational cost is high due to full training set for all the networks and small number of initial hidden nodes.

Our purpose is to develop an automatic NNE creation algorithm that will overcome the problems of previous methods. The proposed algorithm DCIS uses a rather simple technique to create training set for adding next component network. That is the misclassified patterns of partially trained networks. Thus any probabilistic calculation is unnecessary unlike the bagging and the boosting; only checks, which patterns are misclassified by all exiting partially trained networks. We used fixed number of hidden nodes per network for a particular problem on the basis of the number of output classes [3]**.** This further simplifies the calculation. Increasing the number of hidden nodes does not improve the performance after a particular number [5], and that's rather depends on the number of output classes. The computational cost is therefore lower than CNNE. Another advantage of the proposed algorithm is that the meaning of two user define parameters is easy understandable, while that of CNNE is rather complex [1].

III. DYNAMICAL CHANGE IN INPUT SPACE (DCIS) METHOD

In DCIS, there are three major steps to create an NNE. Firstly, it partially trains a network with the patterns that are misclassified by all exiting partially trained networks. Secondly, it trains this new network with full training set but assuming all output node values equal to zero for the patterns that are misclassified by this new network. By the second partial training it forces to produce equal output in all the output nodes for misclassified patterns. This new network therefore will not oppose other networks to classify these patterns that are presently misclassified. In another sense, this network will be diverted for these classified patterns without affecting others. The third and final training is performed with all component networks via negative correlation with full training set to minimize the total error of the whole NNE. The major steps of DCIS are summarized in Fig. 1, which are explained further below:

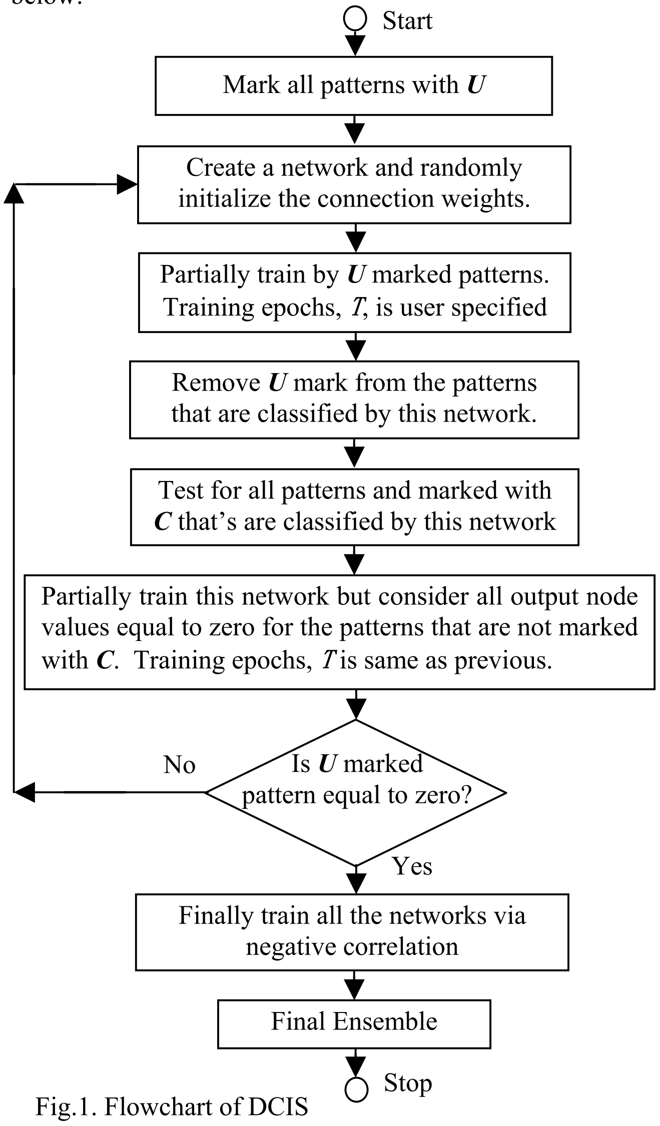
Start
Mark all patterns with ***U***
Create a network and randomly initialize the connection weights.
Partially train by ***U*** marked patterns. Training epochs, *T*, is user specified
Remove ***U*** mark from the patterns that are classified by this network.
Test for all patterns and marked with ***C*** that's are classified by this network
Partially train this network but consider all output node values equal to zero for the patterns that are not marked with ***C***. Training epochs, *T* is same as previous.
Is ***U*** marked pattern equal to zero?
No
Yes
Finally train all the networks via negative correlation
Final Ensemble
Stop

Fig.1. Flowchart of DCIS

Step 1) Divide input patterns into three sets: a training set, a validation set and a test set. Mark all training patterns with ***U*** to indicate unclassified by all exiting partially trained networks in the NNE.

Step 2) Create a network with a defined number of hidden nodes and randomly initialize all connection weights. The number of nodes in the input and output layers are the same as the number of inputs and outputs of the problem, respectively.

Step 3) Partially train this network with the ***U*** marked patterns. The number of partial training epochs (***T***) is a user-defined parameter.

Step 4) Remove the ***U*** mark from the patterns that are classified by this network.

Step 5) Check this network for full training set, how many patterns are truly classified. Mark truly classified patterns with ***C*** for this network.

Step 6) Partially train this network but change input space by assuming all output node values equal to zero for the patterns that are not marked with ***C***. The number partial training epochs is equal to step 3.

Step 7) Calculate the ***U*** marked patterns. If the number of ***U*** marked patterns is zero i.e., all patterns are truly classified by one or more component networks; then proceed step 8, other wise go to step 2 to add another component network.

Step 8) Finally train all the networks via negative correlation to minimize the NNE error E [1] on the *validation* set. Training stops when the error reaches within the acceptable range or when the error status to increase. E is defined by

$$E = 100\frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}\left(\frac{1}{M}\sum_{i=1}^{M}F_i(n) - d(n)\right)^2 \qquad (1)$$

Where N is the number of patterns in the validation set, M is the number of component networks in the NNE, $F_i(n)$ is the output of network i on the nth training pattern, and $d(n)$ represents the desired output for the nth pattern.

Step 9) Determine the accuracy of the NNE using the test set.

IV. EXPERIMENTAL STUDIES

To evaluate the performance, we tested DCIS on five well-known real world benchmark classification problems. These are Australian Credit Card (ACC) problem, Breast Cancer Wisconsin (BCW) problem, Diabetes (DB) problem, Heart Disease Cleveland (HDC) problem and Iris Plants (IP) problem. Origin of these data sets is the UCI machine learning benchmark repository. The detailed descriptions are available in UCI web site (http://www.ics.uci.edu/~mlearn/). But in DCIS we used preprocessed Proben1 datasets that were obtained from (ftp://ftp.ira.uka.de/pub/neuron/) except Iris Plants problem.

We preprocessed UCI Iris Plans data by rescaling input attribute values between 0.0 and 1 by a linear function. Also we randomly distributed individual Iris Plan, and then equally distributed three Iris Plans. For all problems we followed the neural network benchmark methodology [7]. All datasets are partitioned into three sets: a training set, a validation set, and a testing set. The testing set is used to evaluate the generalization performance of the trained NNE and is not seen by any individual network during the whole training process. It is known that the experimental results may vary significantly for different partitioning of the same data collection, even when the number of examples in each set is the same [7]. TABLE I shows the characteristics and partitions of data sets.

TABLE I
CHARACTERISTICS AND PARTITIONS OF DATASETS

Prob-lem	Input Attributes	Output Classes	Total and Set Wise Examples			
			Total	Training	Validation	Test
ACC	51	2	690	345	173	172
BCW	9	2	699	350	175	174
DB	8	2	768	384	192	192
HDC	35	2	303	152	76	75
IP	4	3	150	75	38	37

A. Experimental Setup

DCIS is an automatic NNE creation algorithm with two user-defined parameters; one is the number of partial training epochs (T) and another is the number of hidden nodes per network. The number of networks in NNE and error rate depends upon these two parameters. For our experiment, we set these two parameters to create three networks in an NNE. We trained individual neural networks using the standard back propagation [BP] learning [12]. Parameter settings of the BP learning are; the learning rate 0.1 and initial random weights between -0.1 & 0.1. At the time of final training of NNE we selected the penalty term of negative co-relation learning [2] as 0.5. To combine output from component networks, we used simple average [4].

B. Results and Comparison

TABLE II compares the average test set error rate with other NNE methods. TABLE III shows the comparison of the architecture produced by DCIS with those by other methods. DCIS's result is the average of ten independent runs. The results of Arc Boosting, Ada Boosting and Bagging were taken from Opitz and Maclin[3]. No result has been reported for Iris Plants for CNNE [1]. Form the Table II, it is observed that DCIS is the best for Brest Cancer Wisconsin and Iris Plants, and is better or compatible for other problems. As seen in TABLE III, it is clear that the NNE created by DCIS is much more compact than any other methods.

TABLE II
AVERAGE TEST SET ERROR COMPARISON AMONG DCIS, ARC BOOSTING, ADA BOOSTING, BAGGING AND CNNE

Prob-lem	Setting		DCIS	Arc Boos-ting	Ada Boos-ting	Bagg-ing	CNNE
	T	# HN					
ACC	200	3	0.139	0.158	0.138	0.157	0.092
BCW	15	1	0.012	0.038	0.034	0.040	0.013
DB	50	3	0.234	0.244	0.228	0.233	0.198
HDC	50	3	0.164	0.207	0.170	0.211	0.134
IP	30	5	0.0	0.037	0.040	0.039	---

TABLE III
ARCHITECTURE COMPARISON AMONG NNE CREATED BY DCIS, ARC BOOSTING, ADA BOOSTING, BAGGING AND CNNE

Prob-lem	Average Network Numbers			Average Hidden Nodes Per Network		
	DCIS	Boos-ting/ Bagg-ing	CNNE	DCIS	Boos-ting/ Bagg-ing	CNNE
ACC	2.5	25	7.8	3	10	4.7
BCW	3	25	4.8	1	5	2.9
DB	3	25	6.5	3	5	3.4
HDC	3	25	5.5	3	5	4.9
IP	3	25	---	5	5	---

TABLE IV
EFFECT OF PARTIAL TRAINING ASSUMING THAT ALL OUTPUT NODE VALUES ARE EQUAL TO ZERO FOR MISCLASSIFIED PATTERNS ON AUSTRALIAN CREDIT CARD PROBLEM

Set	Before NNE Train		After NNE Train	
	With	Without	With	Without
Training	311	210	321	311
Validation	152	98	154	154
Test	144	98	149	148

C. Effect of partial training assuming that all output node values are equal to zero for misclassified patterns

As a sample, we tested the effect of the partial training assuming that all output node values are equal to zero for misclassified patterns (second partial training) on Australian Credit Card Problem. Same initial random weights were used and the setting of DCIS is as like Table II. As shown in Table IV, with the second partial training, DCIS truly classify 311 & 321 patterns before and after the final NNE training (the third train with negative correlation) respectively from training set. On the other hand, without second partial training the numbers are 210 & 311. So the second partial train allows DCIS to classify 10 more patterns from the training set and also 1 more pattern from the test set.

D. Variation effect of partial training epochs (T)

Figs. 2-6 show the variation effect of partial training epochs (T) on the number of networks and the error rate for various problems from same initial random weights. From these figures, it is observed that, if number of networks created is same for different values of T but the error rate changed due to variation of T. The number of hidden nodes is same as that of Table II and the final NNE training minimized error for the validation set. If error minimizes on the training set, the error rate decreased for the training set but increase for the validation and the test set. Figs. 2-6 may vary due to initial random weights.

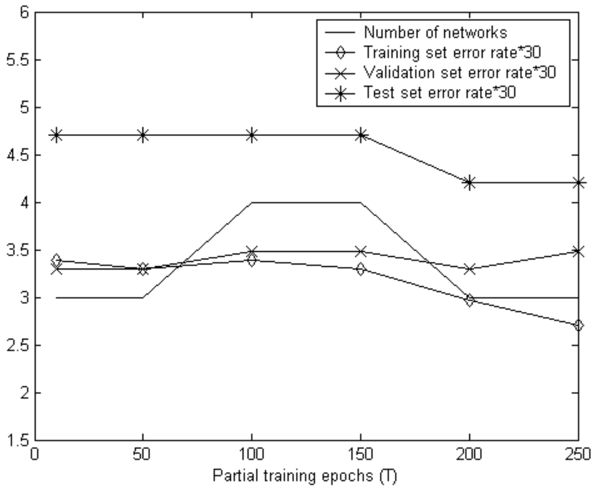

Fig. 2. Australian Credit Card problem.

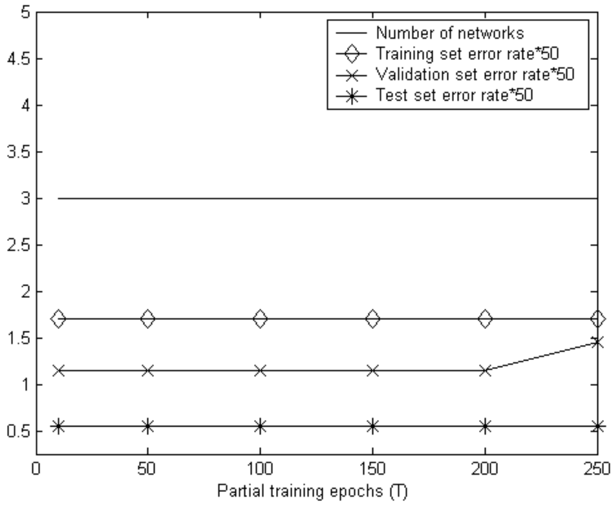

Fig. 3. Breast Cancer Wisconsin problem.

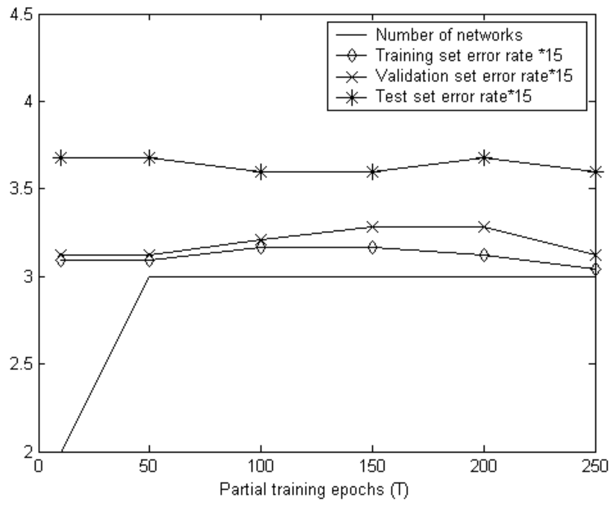

Fig. 4. Diabetes problem.

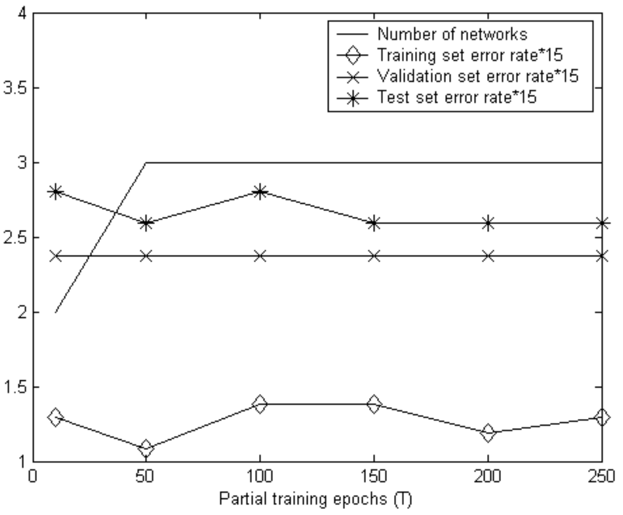

Fig. 5. Heart Disease Cleveland problem.

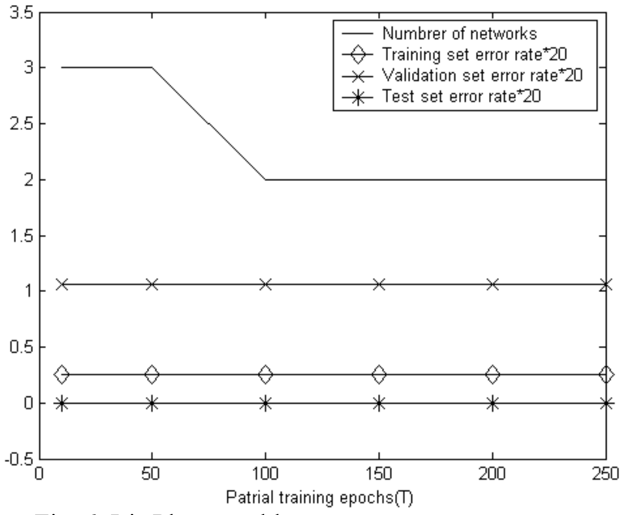

Fig. 6. Iris Plants problem

E. Variation effect of hidden nodes per network

Figs. 7-11 show the variation effect of hidden nodes per network for same initial random weights. Here T is same as that of Table II. From these figures, it is observed that when the number of hidden nodes per network is less than the number of output classes, error rate is high in general. When the number is increased up to a certain level, the error rate does not improve any further. It is observed that optimized number of hidden nodes per network is related to the number output classes [3]. Figs. 7-11 may vary due to initial random weights.

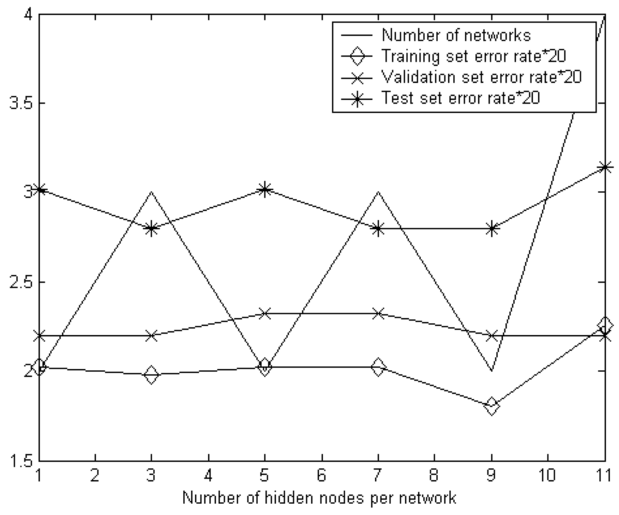

Fig. 7. Australian Credit Card problem.

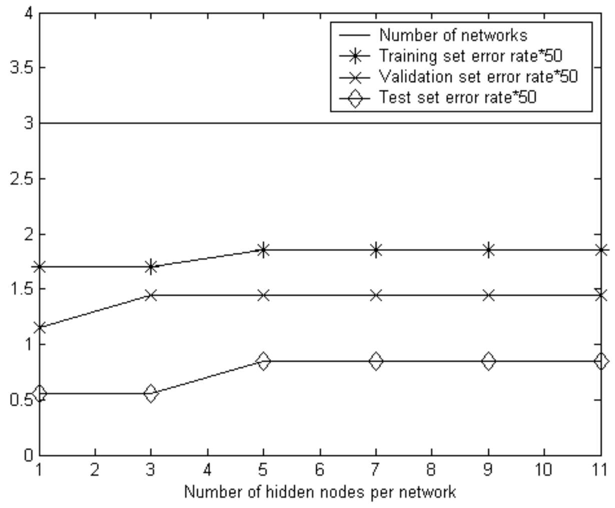

Fig. 8. Breast Cancer Wisconsin problem.

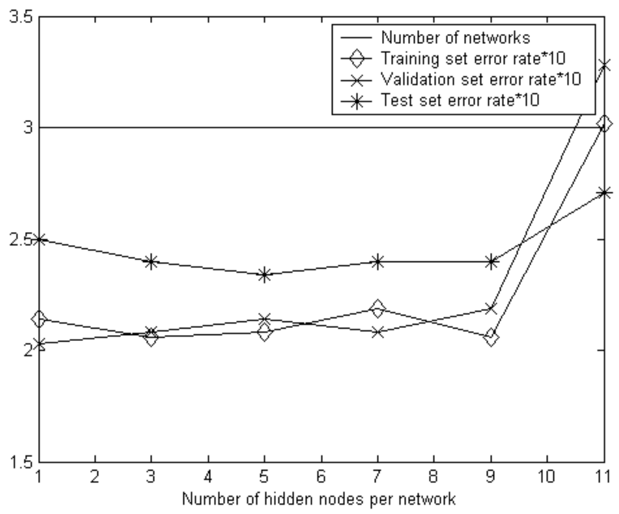

Fig. 9. Diabetes problem

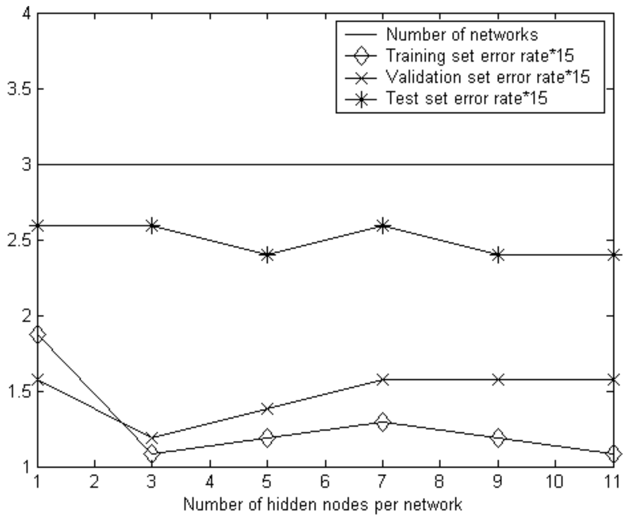

Fig. 10. Heart Disease Cleveland problem

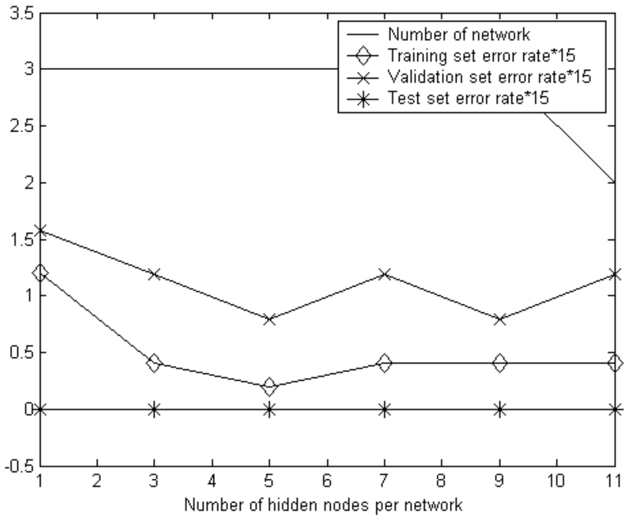

Fig. 11. Iris Plants problem

V. CONCLUSIONS

We have proposed a new automatic NNE creation algorithm in this paper. The novelty of the proposed method is that this method does not require probabilistic calculation to add next component network unlike the previously proposed methods such as the bagging and the boosting. And also the procedure is straightforward and simple. Performance of this method is better or equivalent to that of other hand-made NNE algorithms. Though the performance is inferior to CNNE's for most of the cases, the final architecture of NNE are much more compact in the proposed method.

There are three training steps in the proposed NNE creation method. Firstly, partial train with misclassified patterns, secondly partial train by changing input space, and thirdly, NNE training via negative correlation. For the two partial trainings, we used same training epochs for simplicity. So further conclusion may be drawn altering the second partial training epochs.

REFERENCES

[1] Md. Monirul Islam, X.Yao and K. Murase. A Constructive Algorithm for Training neural Network Ensembles, *IEEE Transactions on Neural Networks*, vol. 14, no. 4, 2003, pp. 820-834.

[2] Y. Liu and X. Yao. Ensemble learning via negative correlation, *Neural Networks*, vol. 12, 1999, pp. 1399-1404.

[3] D. W. Opitz and R. Maclin. Popular ensemble methods: An empirical study, *Journal of Artificial Intelligence Research*, vol. 11, 1999, pp. 169-198.

[4] A. J. C. Sharkey. On combining artificial neural nets, *Connection Science*, vol. 8, no. 3/4, 1996, pp. 299-314.

[5] A. J. C. Sharkey, Combining diverse neural nets, *Knowledge Engineering Review*, vol. 12 no. 3, 1997, pp. 299-314.

[6] B. E. Rosen. Ensemble learning using decorrelated neural networks, *Connection Science*, vol. 8, 1996, pp. 373-383.

[7] L. Prechelt. Proben1- a set of benchmarks and benching rules for neural network training algorithms, *Tech. Rep. 21/94, Fakultat fur Informatik, University of Karlsruhe, Germany*, 1994.

[8] J. R. Quinlan. Bagging, boosting, and C4.5*, in Proc. Of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference, Menlo park,* August 4-8, 1996, AAAI Press/MIT Press, 1996, pp. 725-730.

[9] L. Breiman. Bagging predictors, *Machine Learning*, vol. 24, 1996, pp. 123-140.

[10] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm, *in Proc. 13th International Conference on Machine Learning, Morgan kaufmann,* 1996, pp. 148-156.

[11] A. J. C. Sharkey, N. E. Sharkey, U. Gerecke and G. O. Chandroth. The `test and select' approach to ensemble combination, *in Multiple Classifier Systems, LNCS 1857, J. Kittler and F. Roli, Eds.,* 30-44 (Springer-Verlag, 2000).

[12] D. Rumelhart, G. Hinton, & R. Williams. Learning internal representations by error propagation*, Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations, MIT Press, Cambridge, MA,* 1986, pp. 318-363