Prediction error of a fault tolerant neural network

John Sum[1,2], Chi-sing Leung[2], and Kevin Ho[3⋆]

[1] Department of Information Management, Chung Shan Medical University
Taichung 402, Taiwan pfsum@csmu.edu.tw
[2] Department of Electronic Engineering, City University of Hong Kong
Kowloon Tong, KLN, Hong Kong eeleungc@cityu.edu.hk
[3] Department of Computer Science and Communication Engineering,
Providence University, Sha-Lu, Taiwan. ho@pu.edu.tw

Abstract. For more than a decade, prediction error has been one powerful tool to measure the performance of a neural network. In this paper, we extend the technique to a kind of fault tolerant neural network. Consider a neural network to be suffering from multiple-node fault, a formulae similar to that of Generalized Prediction Error has been derived. Hence, the effective number of parameter of such a fault tolerant neural network is obtained. A difficulty in obtaining the mean prediction error is discussed and then a simple procedure for estimation of the prediction error empirically is suggested.

1 Introduction

Obtaining a neural network to tolerate random node fault is of paramount important as node fault is an unavoidable factor while a neural network is implemented [13]. In view of the importance of making a neural network being fault tolerant, various researches have been conducted throughout the last decade in order to attain a fault tolerant neural network that can alleviate problems due to random node fault.

Injecting random node fault [2, 17] together with random node deletion and addition [4] during training is one common approach . Adding network redundancy by replicating hidden nodes/layers after trained [6, 15], adding weight decay regularizer [4] and hard bounding the weight magnitude during training [3] are other techniques that have also been proposed in the literature. In accordance with simulation results, all these heuristic techniques have demonstrated that the network is able to tolerate against random node fault, either single node or multiple nodes have stuck-on faults. As these techniques are heuristics, it is not clear in theory about their underlying objective function or their prediction errors being achieved. In sequel, analysis and comparison on the similarities and differences between one technique to another can hardly be accomplished except by extensive simulations.

An alternative approach in training a fault tolerant neural network is to formulate the learning problem as a constraint optimization problem. Neti *et al* [14] defined the problem as a minimax problem in which the objective function

⋆ Corresponding author.

to be minimized is maximum of the mean square errors over all fault models. Deodhare *et al* [5] formulated the problem by defining the objective function to be minimized as the maximum square error over all faulty neural models and all training samples. A drawback of the above approaches is that the complexity of solving such problem could be very complex as the number of hidden units are large and the number of faulty nodes cannot be larger than one. Simon & El-Sherief [18] and Phatak & Tcherner in [16] formulated the learning problem as an unconstraint optimization problem in which the objective function consists of two terms. The first term is the mean square errors of the fault-free model while the second term is the ensemble average of the mean square errors over all fault models.

One limitation of these formulations is that the problem being formulated can be very complicated when the number of fault nodes is large. Extend their formulations to handling multiple nodes fault will become impractical. In view of the lacking of a simple objective function to formalize multiple nodes fault and the lacking of an understanding of the relation between fault tolerant and generalization, Leung & Sum in [10] have recently derived a simple objective function and yet another regularizer from KL divergence for robust training a neural network that can optimally tolerate multiple nodes fault.

In this paper, we extend the idea elucidated in [10] by deducing the mean prediction error equation for such a fault tolerant neural network model being attained. As it is believed that prediction error is an alternative measure for the performance of a neural network [9]. The rest of the paper will be organized as follows. The next section will define what is a node fault tolerant neural network and present an objective function derived in [10] for attaining such a fault tolerant neural network. The prediction error equation (main contribution of the paper) will be derived in Section 3. Section 4 will describe how this error can be obtained in practice. Experimental results are described in Section 5. Then, we conclude this paper in Section 6.

2 Node fault tolerant neural network

Throughout the paper, we assume that training data set $D_T = \{(x_k, y_k)\}_{k=1}^N$ is extracted from an unknown stochastic system defined as follows : For $k = 1, 2, \cdots, N$

$$y_k = f(x_k) + e_k \tag{1}$$

where x and y are the input and the output of an unknown deterministic system $f(x)$; and e is a random measurement noise. Consider the output of the system is a dependent random variable governed by the input x, the behavior of the system can be denoted by the conditional probability $P_0(y|x)$.

An RBF network consisting of M hidden nodes is defined as follows :

$$\hat{f}(x, \theta) = \sum_{i=1}^{M} \theta_i \phi_i(x), \tag{2}$$

where $\phi_i(x)$ for all $i = 1, 2, \cdots, M$ are the radial basis functions given by

$$\phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{\sigma}\right); \tag{3}$$

c_is are the radial basis function centers and the positive parameter $\sigma > 0$ controls the width of the radial basis functions. Without loss of generality, we assume that $c_i \in R$ for all i. Model (2) is called *fault-free RBF network*.

Next, we assume that a node fault is a stuck-on-zero node fault. That is to say, the output of the node will permanently be stuck on zero value once it has became faulty. Therefore, a faulty RBF denoted by $\hat{f}$ could be written as a summation of $\phi_i(x)$ times θ_i and a random binary variable β_i :

$$\hat{f}(x, \theta, \beta) = \sum_{i=1}^{M} \beta_i \theta_i \phi_i(x). \tag{4}$$

If $\beta_i = 1$, the i^{th} node is normal. Otherwise, the i^{th} node is faulty. Furthermore, it is assumed that all hidden nodes are of equal fault rate p, i.e. $P(\beta_i) = p$ if $\beta_i = 0$ and $P(\beta_i) = (1 - p)$ if $\beta_i = 1$, for all $i = 1, 2, \cdots, M$ and $\beta_1, \cdots, \beta_M$ are independent random variables. Model (4) is called *faulty RBF network*.

In sequel, we approximate the unknown system (1) by the following stochastic system :

$$y = \hat{f}(x, \theta, \beta) + e, \tag{5}$$

where e is a mean zero Gaussian noise defined in (1), x and y are respectively the input and the output of the faulty RBF. Represent the behavior of this faulty RBF by a conditional probability $P(y|x, \theta, \beta)$ and let $\tilde{\theta} = (\beta_1\theta_1, \cdots, \beta_M\theta_M)$, the conditional probability of a faulty RBF given x as input could be denoted by $P(y|x, \tilde{\theta})$.

Let $P_0(x)$ be probability distribution of input x, the joint probability distributions for the unknown system and the faulty RBF can be given by

$$P_0(x, y) = P_0(y|x)P_0(x), \quad P(x, y|\tilde{\theta}) = P(y|x, \tilde{\theta})P_0(x).$$

To measure the discrepancy between a faulty RBF $\tilde{\theta}$ and the unknown system, one can apply the Kullback-Leibler divergence [8] :

$$D(P_0||P_{\tilde{\theta}}) = \int\int P_0(x, y) \log \frac{P_0(x, y)}{P(x, y|\tilde{\theta})} dxdy. \tag{6}$$

Since $\tilde{\theta}$ is an unknown and it is depended on the fault-free model θ, the average distance of all fault models (all possible $\beta \in \{0, 1\}^M$) with reference to the true distribution $P_0(x, y)$ can be defined as

$$\bar{D}(P_0||P_\theta) = \int \left\{ \int\int P_0(x, y) \log \frac{P_0(x, y)}{P(x, y|\tilde{\theta})} dxdy \right\} P(\tilde{\theta}|\theta) d\tilde{\theta} \tag{7}$$

$$= \left\langle \int\int P_0(x, y) \log \frac{P_0(x, y)}{P(x, y|\tilde{\theta})} dxdy \right\rangle_{\Omega_\beta}. \tag{8}$$

Here Ω_β corresponds to the set consisting all the possible β.

It can be shown [10] that maximizing $\bar{D}(P_0||P_\theta)$ is equivalent to minimizing the following objective function :

$$E(\theta, p) = \frac{1}{N} \sum_{k=1}^{N} y_k^2 - 2(1 - p) \frac{1}{N} \sum_{k=1}^{N} y_k \phi^T(x_k)\theta + (1 - p)\theta^T \left\{ (1 - p)\hat{H}_\phi + p\hat{G} \right\} \theta,$$

where p is the node fault rate, $\hat{H}_\phi = \frac{1}{N}\sum_{k=1}^{N}\phi(x_k)\phi^T(x_k)$,

$$\hat{G} = \mathbf{diag}\left\{\frac{1}{N}\sum_{k=1}^{N}\phi_1^2(x_k), \cdots, \frac{1}{N}\sum_{k=1}^{N}\phi_M^2(x_k)\right\}.$$

Take the first derivative of $E(\theta, p)$ and set it to zero, the corresponding optimal fault tolerant RBF $\hat{\theta}$ will be given by

$$\hat{\theta} = \left(\hat{H}_\phi + p\left(\hat{G} - \hat{H}_\phi\right)\right)^{-1}\frac{1}{N}\sum_{k=1}^{N} y_k\phi(x_k). \tag{9}$$

Since $\hat{H}_\phi$ and $\hat{G}$ are functions of $\phi(x_1), \cdots, \phi(x_N)$, $\hat{\theta}$ can be obtained as long as $\{x_k, y_k\}_{k=1}^{N}$ are given. Model $\hat{f}(x, \hat{\theta}, \beta)$ is a *node fault tolerant RBF network.*

3 Mean prediction error

To attain a fault tolerant model, the estimation of mean prediction error is accomplished by using a testing data set. But it is known that testing data set is sometimes unavailable in particular the available data set is not large. It is not able to split the data set into training set and data set. In such case, the performance of a fault tolerant neural network could be estimated by a *mean prediction error* equation, a formula similar to that of AIC [1], GPE [11] or NIC [12].

For presentation clarity, a summary of the notations being used is depicted in Table 1. $\langle E(\mathcal{D}_T|\hat{\theta}(p, \mathcal{I}-\mathcal{K}))\rangle$ and $\langle E(\mathcal{D}_F|\hat{\theta}(p, \mathcal{I}-\mathcal{K}))\rangle$ are defined as follows :

$$\begin{aligned}\langle E(\mathcal{D}_F|\hat{\theta}(p,\cdot))\rangle &= \left\langle f^2(x')\right\rangle_{\mathcal{D}_F} - 2(1-p)\left\langle f(x')\phi^T(x')\right\rangle_{\mathcal{D}_F}\hat{\theta} \\ &\quad +(1-p)\hat{\theta}^T\left\{(1-p)H'_\phi + pG'\right\}\hat{\theta}. \\ \langle E(\mathcal{D}_T|\hat{\theta}(p,\cdot))\rangle &= \left\langle f^2(x)\right\rangle_{\mathcal{D}_T} - 2(1-p)\left\langle f(x)\phi^T(x)\right\rangle_{\mathcal{D}_T}\hat{\theta} \\ &\quad +(1-p)\hat{\theta}^T\left\{(1-p)H_\phi + pG\right\}\hat{\theta}.\end{aligned}$$

Again, we assume that N is large. We replace Ω_x by $\mathcal{D}_F$ and $\mathcal{D}_T$, which are equal to $\{(x'_k, y'_k)\}_{k=1}^{N}$ and $\{(x_k, y_k)\}_{k=1}^{N}$, to reflect that the data are from testing set and training set respectively. Obviously, the difference between $\langle E(\mathcal{D}_F|\hat{\theta}(p,\cdot))\rangle$ and $\langle E(\mathcal{D}_T|\hat{\theta}(p,\cdot))\rangle$ lies in the difference between their second terms. It is because $H'_\phi = H_\phi$ and $G' = G$ for larger N. Their first terms and third terms are identical. One should further note that $\hat{\theta}$ is obtained entirely by $\mathcal{D}_T$, which is independent of $\mathcal{D}_F$. Therefore, for large N, we can have

$$\left\langle y\phi^T(x)\hat{\theta}\right\rangle_{\mathcal{D}_F} = \left\langle \frac{1}{N}\sum_{k=1}^{N} y'_k\phi^T(x'_k)\right\rangle_{\mathcal{D}_F}\hat{\theta}$$

where x'_k are not from $\mathcal{D}_T$. Following the same technique as using in [12] and [9], we assume that there is an optimal θ_0 such that

$$y_k = \phi^T(x_k)\theta_0 + e_k; \quad y'_k = \phi^T(x'_k)\theta_0 + e'_k.$$

Notation	Description
$\mathcal{D}_T$	Training data set
$\mathcal{D}_F$	Testing data set
p	Fault rate — probability that a node will be failure
M	Number of radial basis functions (nodes)
$\mathcal{I}$	Index set, $\{1, 2, \cdots, M\}$ for the nodes
$\mathcal{K}$	Set of nodes being removed
$\mathcal{I} - \mathcal{K}$	Remaining node set
$\hat{\theta}(p, \mathcal{I})$	Model obtained by Equation (9)
$\hat{\theta}(p, \mathcal{I} - \mathcal{K})$	$\hat{\theta}(p, \mathcal{I} - \mathcal{K})_i = \hat{\theta}(p, \mathcal{I})_i$ if i is not in $\mathcal{K}$; zero otherwise
$E(\mathcal{D}_T \vert \hat{\theta}(p, \mathcal{I} - \mathcal{K}))$	Mean square training errors due to $\hat{\theta}(p, \mathcal{I} - \mathcal{K})$
$E(\mathcal{D}_F \vert \hat{\theta}(p, \mathcal{I} - \mathcal{K}))$	Mean square testing errors due to $\hat{\theta}(p, \mathcal{I} - \mathcal{K})$
$\langle E(\cdot \vert \cdot) \rangle$	Average mean square errors

(a)

	$\hat{\theta}(p, \mathcal{I})$	$\hat{\theta}(p, \mathcal{I} - \mathcal{K})$
$\mathcal{D}_T$	$E(\mathcal{D}_T \vert \hat{\theta}(p, \mathcal{I}))$	$E(\mathcal{D}_T \vert \hat{\theta}(p, \mathcal{I} - \mathcal{K}))$
$\mathcal{D}_F$	$E(\mathcal{D}_F \vert \hat{\theta}(p, \mathcal{I}))$	$E(\mathcal{D}_F \vert \hat{\theta}(p, \mathcal{I} - \mathcal{K}))$

(b)

Table 1. Notation (a) and the relations between different error terms (b). Note that $E(\mathcal{D}_T|\hat{\theta}(p, \mathcal{I}))$ and $E(\mathcal{D}_F|\hat{\theta}(p, \mathcal{I}))$ are not we are interested because they assume no node removed.

where $e_k, e'_k \sim \mathcal{N}(0, S_e)$ for all $k = 1, 2, \cdots, N$. The second term in $\langle E(\mathcal{D}_F|\hat{\theta}(p, \cdot))\rangle$ can thus be given by

$$-2(1-p)\theta_0^T H_\phi((1-p)H_\phi + pG)^{-1} H_\phi \theta_0$$

while the second term in $\langle E(\mathcal{D}_T|\hat{\theta}(p, \cdot))\rangle$ is given by

$$-2(1-p)\frac{S_e}{N}\mathbf{Tr}\left\{H_\phi((1-p)H_\phi + pG)^{-1}\right\} - 2(1-p)\theta_0^T H_\phi((1-p)H_\phi + pG)^{-1} H_\phi \theta_0.$$

As a result, the difference between the mean prediction error and mean training error can be written as follows :

$$\begin{aligned}\langle E(\mathcal{D}_F|\hat{\theta}(p, \cdot))\rangle = &\langle E(\mathcal{D}_T|\hat{\theta}(p, \cdot))\rangle \\ &+ 2\frac{S_e}{N}\mathbf{Tr}\left\{(1-p)H_\phi((1-p)H_\phi + pG)^{-1}\right\}. \qquad (10)\end{aligned}$$

Let

$$M_{eff} = \mathbf{Tr}\left\{(1-p)H_\phi((1-p)H_\phi + pG)^{-1}\right\}.$$

It can be interpreted as the effective number of parameter of an RBF of $(1-p)M$ number of nodes as the way in [11]. Therefore, the true S_e can be approximated by the following equation

$$S_e \approx \frac{N}{N - M_{eff}}\langle E(\mathcal{D}_T|\hat{\theta}(p, \cdot))\rangle.$$

The prediction error can then be approximated by the following equation,

$$\langle E(\mathcal{D}_F|\hat{\theta}(p,\cdot))\rangle = \frac{N+M_{eff}}{N-M_{eff}}\langle E(\mathcal{D}_T|\hat{\theta}(p,\cdot))\rangle. \tag{11}$$

To use this approximation, the simulation to be conducted is a bit not as usual. Suppose we have a set of measure data, $\mathcal{D}_T$. After robust model is thus obtained by Equation (9), as many as possible fault RBF models are generated. Their $E(\mathcal{D}_T|\hat{\theta}(p,\cdot))$s are thus obtained by simulation. The average of the $E(\mathcal{D}_T|\hat{\theta}(p,\cdot))$s can then be used as the value for $\langle E(\mathcal{D}_T|\hat{\theta}(p,\cdot))\rangle$ and the prediction error $\langle E(\mathcal{D}_F|\hat{\theta}(p,\cdot))\rangle$ can then estimated by Equation (11) immediately.

4 Estimation of MPE

It should be noted that obtaining the value $\langle E(\mathcal{D}_T|\hat{\theta}(p,\cdot))\rangle$ could be very expensive. Take $M = 50$ and $p = 0.1$ as an example. The total number of faulty models are approximately $50!/(5! \times 45!)$, i.e. 2118760. Extensive simulation is infeasible. Under such circumstance, one could only approximate the average mean training error by the sample average.

If S_e and p are given, a number of faulty models are generated uniformly random. The same set of training data is thus fed into the models. The avearge of their mean square errors will thus be used as an approximation of $\langle E(\mathcal{D}_T|\hat{\theta}(p,\cdot))\rangle$. It is equivalent to approximate the prediction error by the following equation :

$$\begin{aligned} E(\mathcal{D}_F|\hat{\theta}(p,\mathcal{I}-\mathcal{K})) \approx\ & E(\mathcal{D}_T|\hat{\theta}(p,\mathcal{I}-\mathcal{K})) \\ & + 2\frac{S_e}{N}\mathbf{Tr}\left\{(1-p)H_\phi((1-p)H_\phi+pG)^{-1}\right\}, \end{aligned} \tag{12}$$

where H_ϕ and G could be obtained by using the training data only. If S_e is not given, the prediction error could be estimated by

$$E(\mathcal{D}_F|\hat{\theta}(p,\mathcal{I}-\mathcal{K})) \approx \frac{N+M_{eff}}{N-M_{eff}}E(\mathcal{D}_T|\hat{\theta}(p,\mathcal{I}-\mathcal{K})). \tag{13}$$

As a result, the mean prediction error can thus be estimated by the following steps : *(1) Calculate H_ϕ and G based on the training data. (2) Obtain $\hat{\theta}$ based on the value of p. (3) Random generate a sample set of faulty models in accordance with the fault rate p. (4) Obtain the mean training errors for these faulty models. (5) Approximate the average mean training error by the sample average of these faulty models. (6) Estimate $E(\mathcal{D}_F|\hat{\theta}(p,\mathcal{I}-\mathcal{K}))$ either by Equation (12) or Equation (13).*

5 Experimental results

Figure 1 shows an experimental results comparing the estimated and actual mean prediction error. In this experiment, 20 RBF networks are generated to approximate a simple noisy function, $f(x) = \tanh(x) + e$, where $e \sim \mathcal{N}(0, 0.01)$

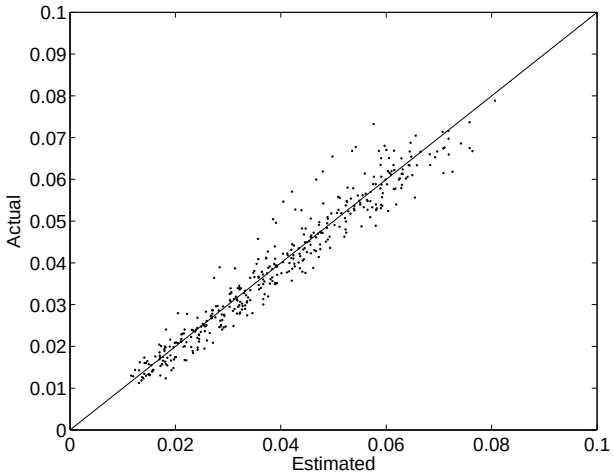

Fig. 1. Actual MPE versus estimated MPE.

a mean zero Guassian noise. Each RBF network consists of 17 centers generated uniformly in the range of $[-4, 4]$ with 0.5 distance apart. The width of a basis function, i.e. σ, is set to 0.49. 20 independent training data sets are generated for each of the RBF networks. Each training set consists of 50 training data, with xs are uniformly randomly generated in the range $[-4.4]$ and es are randomly generated in accordance with Guassian distribution. An extra data sets consisting of 100 data is also generated as the testing set.

Follow the steps described above, each network is trained with its own training data set for different fault rates. Here, the fault rate is set to be $0.01, 0.02, 0.03 \cdots, 0.2$. For each p, $\hat{\theta}$ is obtained after H_ϕ and G have been calculated. Then 100 faulty network models are generated and their training errors are measured. The estimated mean prediction error $E(\mathcal{D}_F|\hat{\theta}(p, \mathcal{I} - \mathcal{K}))$ is estimated by Equation (12). Finally, the actual prediction error is obtained simply by feeding the testing data set to these 100 faulty network models again and taking their average. The actual prediction error against the estimated prediction error for different values of p is thus shown in Figure. The solid line, $y = x$, is used for reference. It is clearly that the points lie symmetically along the solid straight line.

6 Conclusion

Following the objective function we have derived in [10], we have analyzed in this paper the mean prediction error for such a fault tolerant neural network being attained and then derived a simple procedure to estimate such value after training. As mean prediction error is in fact a measure on the performance of a neural network towards the future data, the equation and the estimation procedure derived can be used as a mean to estimate the generalization ability of such a (multiple-nodes) fault tolerant neural network after trained by the robust learning algorithm we derived in [10].

Acknowledgement

The work is supported by a research grant from City University of Hong Kong (Project No. 7001850).

References

1. Akaike H., A new look at the statistical model identification, *IEEE Transactions on Automatic Control*, Vol.19, 716-23, 1974.
2. Bolt G., *Fault tolerant in multi-layer Perceptrons*. PhD Thesis, University of York, UK, 1992.
3. Cavalieri S. and O. Mirabella, A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks, *Neural Networks*, Vol.12, 91-106, 1999.
4. Chiu C.T. *et al.*, Modifying training algorithms for improved fault tolerance, ICNN'94 Vol.I, 333-338, 1994.
5. Deodhare D., M. Vidyasagar and S. Sathiya Keerthi, Sythesis of fault-tolerant feedforward neural networks using minimax optimization, *IEEE Transactions on Neural Networks*, Vol.9(5), 891-900, 1998.
6. Emmerson M.D. and R.I. Damper, Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application, *IEEE Transactions on Neural Networks*, Vol.4, 788-793, 1993.
7. Fontenla-Romero O. *et al*, A measure of fault tolerance for functional networks, *Neurocomputing*, Vol.62, 327-347, 2004.
8. Kullback S., *Information Theory and Statistics*, Wiley, 1959.
9. Leung C.S., G.H. Young, J. Sum and W.K. Kan, On the regularization of forgetting recursive least square, *IEEE Transactions on Neural Networks*, Vol.10, 1842-1846, 1999.
10. Leung C.S. and J. Sum, A fault tolerant regularizer for RBF networks, in submission.
11. Moody J.E., Note on generalization, regularization, and architecture selection in nonlinear learning systems, *First IEEE-SP Workshop on Neural Networks for Signal Processing*, 1991.
12. Murata N., S. Yoshizawa and S. Amari. Network information criterion–Determining the number of hidden units for an artificial neural network model, *IEEE Transactions on Neural Networks*, Vol.5(6), pp.865-872, 1994.
13. Murray A.F. and P.J. Edwards, Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training, *IEEE Transactions on Neural Networks*, Vol.5(5), 792-802, 1994.
14. Neti C. M.H. Schneider and E.D. Young, Maximally fault tolerance neural networks, *IEEE Transactions on Neural Networks*, Vol.3(1), 14-23, 1992.
15. Phatak D.S. and I. Koren, Complete and partial fault tolerance of feedforward neural nets., *IEEE Transactions on Neural Networks*, Vol.6, 446-456, 1995.
16. Phatak D.S. and E. Tcherner, Synthesis of fault tolerance neural networks, *Proc. IJCNN'02*, 1475-1480, 2002.
17. Sequin C.H. and R.D. Clay, Fault tolerance in feedforward artificial neural networks, *Neural Networks*, Vol.4, 111-141, 1991.
18. Simon D. and H. El-Sherief, Fault-tolerance training for optimal interpolative nets, *IEEE Transactions on Neural Networks*, Vol.6, 1531-1535, 1995.