Hidden-layer Neuron Redundancy—Analysis and Application in MLP's Fault Tolerance

Xu Liqin, Hu Dongcheng and Gao Jianbo
Department of Automation
Tsinghua University
Haidian District, Beijing 100084
P. R. China

Abstract — Redundancy on hidden-layer neurons has proven useful in the fault tolerance of neural networks. This approach has been applied successfully in the fault tolerance design of classification neural networks, thus the complete single-fault tolerance can be gained. But this approach can only be applied to the feed forward networks which has hard-limit activation functions in output layer. And it prove that this approach is valid only to single-fault. There are universal faults of neurons and weights in actual applications, so we evaluated this approach under universal faults in feed forward networks. We proved that the global fault-rate is reduced though the redundancy on hidden-layer neurons. Then we presented a practical and valid method of redundancy of hidden-layer neurons to gain fault tolerance.

I. INTRODUCTION

The fault tolerance of Mutilayer Perceptions (MLP) has been addressed on by many researchers in their papers [1~7]. Many researchers present their methods to form a fault-tolerant neural network. All these method can be classified into 2 groups: one is changing the training algorithm, the other is applying component redundancy after training. Some of the former group methods are addressed in [5~7]. The training objective function or training phases are modified and the final network trained is fault-tolerant to some specific faults. This kind of methods can improve the fault tolerance of networks, but it does not mean free cost. More training time is needed, and sometimes the effect is limited. While the methods in the second group can avoid these shortages. In [1~2], the hidden-layer neurons are replicated, therefore the classification MLP can tolerant all single fault. And we make some improvement to reduce the replication components needed [3].

Some shortages of this hidden-layer redundancy method is that it is only applicable to those MLPs that have hard-limit activation function in output layer. And an assumption in this method is that there's only single fault in networks. But many MLPs have differentiable functions in the output layer and the single fault is just an ideal fault model. The practical MLP is a system with universal faults. That is to say, every component may fail thus multi-faults may exist. The faults include broken weight, neuron failure and weight shifting.

So discuss MLP's fault tolerance under universal faults model is meaning. In this paper, we discuss MLP with arbitrary monotonous function in output layer under universal faults model.

II. REDUNDANCY ARCHITECTURE

We develop our research in MLP, the activation function in hidden layer is Sigmoid. We use $W_{ij}^{(n)}$ to denote the weight linked the neuron j in layer n-1 to the neuron i in layer n(include biases), $o_i^{(n)}$ to denote the output of neuron i in layer n, I_i to denote the resultant input value of neuron i in output layer, thus we have:

$$I_i = \sum_{j=1}^{N_1} (w_{ij}^{(2)} o_j^{(1)}) \tag{1}$$

$$y_i^{(2)} = f(I_i) \tag{2}$$

N_1 is the number of hidden-layer neurons, **f()** is the activation function in output layer.

Broken weight, neuron failure and weight shifting exist universally in MLP. We use $W_{ij}^{(n)}$ s-a-0 to simulate a broken weight and the relative error Δ to simulate weight shiftness. We use stochastic variable $\mathbf{X}$, Δ to denote the distributions of them respectively. Thus we have $\mathbf{w}_{ij}^{(n)} = W_{ij}^{(n)}(1+\Delta)\mathbf{X}$ to denote the actual distribution of the weight ($W_{ij}^{(n)}$ is a constant). And we use $O_i^{(n)}$ s-a-1 or s-a-(-1) to simulate a neuron failure, and use stochastic variable $\mathbf{o}_i^{(n)} = \mathbf{Z}(O_i^{(n)}) = \begin{cases} -1 \\ O_i^{(n)} \\ 1 \end{cases}$ to denote it. The output layer neuron failures are not considered in this paper because these faults are so serious that the faulty network can hardly be recovered. Generally speaking, each fault in the neural networks is independent, so we assume that Δ, $\mathbf{X}$, $\mathbf{Z}$ are independently distributed, thus it is apparent that $\mathbf{w}_{ij}^{(2)}$, $\mathbf{o}_i^{(1)}$ are independently distributed.

Phatak and Koren have presented a redundancy method in [1~2]. They give the whole hidden layer replications to gain full single-fault tolerance. But there's too many redundant components using this method, so it's not practical.

In order to reduce the total redundancy number, we give some improvement to this method. We present a new redundancy architecture [3]: give different redundancy number to different hidden neuron based on the "significance" of each hidden neuron to reduce the total redundancy number. This architecture was originally developed for classification MLP, but it can be extended to all kinds of single hidden-layer MLP, it is shown in Fig 1:

0-7803-5995-X/00/$10.00 ©2000 IEEE.

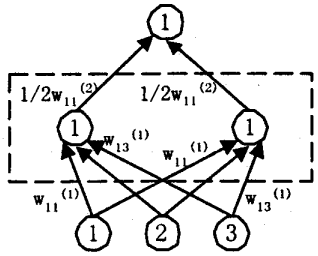


Fig 1 Hidden-layer redundancy architecture

Different redundancy number is given to different hidden-layer neuron. A hidden- layer neuron group includes all weights linked to this neuron (include weights linking input layer and weights linking output layer). We replicate a hidden layer neuron with m_i additional redundant neurons, so there's m_i+1 neurons in this group after replication. The weights linked to output layer in this group are changed to $\frac{1}{m_i+1}$ of original value, weights linked to input layer are not changed.

It should be noted that the replicated network maintains the same function as the original one when no faults exist, i.e., the output error, generalization are of the same as the original networks.

III. EFFECTS OF HIDDEN-LAYER REDUNDANCY TO THE NETWORKS UNDER UNIVERSAL FAULTS

We evaluate the fault tolerance of this redundancy architecture under universal faults model. We can use the classification rate under faults as a fault-tolerance standard of the classification MLP, but it can not be generalized to all MLP. For instance, it can not be used in a optimization or prediction MLP in which we use the mean square error(MSE) of the outputs $P_f = \frac{1}{N_p}\sum_{p=1}^{N_p} E(\frac{1}{N_o}\sum_{i=1}^{N_o}(\mathbf{y}_i^{(p)} - y_{id}^{(p)})^2)$. In order to extend our results to all MLPs, we introduce a new fault tolerance standard—the MSE of the resultant input value of the output layer neurons under faults, i.e., $P_e = \frac{1}{N_p}\sum_{p=1}^{N_p} E(\frac{1}{N_o}\sum_{i=1}^{N_o}(\mathbf{I}_i^{(p)} - I_{id}^{(p)})^2)$. $\mathbf{I}_i^{(p)}$ is a stochastic variable to denote the distribution of the resultant input value of the output layer neuron under faults. $I_{id}^{(p)}$ is the standard resultant input value of the output layer neuron under no faults. Given that the activation function of output layer is f(x), we have $y_{id}^{(p)} = f(I_{id}^{(p)})$. We can input samples into the trained network under no faults to get the resultant input value of the output layer neuron. If the network is well trained, this value approximates $I_{id}^{(p)}$.

We have analyzed the fault tolerance of MLP under single fault [3]. It is shown that the resultant network with this architecture has good fault tolerance. But the practical network is a system under universal faults model. That is to say, in a same time, there may be more than one fault.

Theorem 1: If a hidden-layer neuron is replicated with n($n \geq 1$) additional ones with the architecture shown in Fig 1, the global faulty error will decrease. And the global faulty error is a monotonously degressive function to n. The global faulty error is defined as the MSE of the resultant input value of the output layer neurons under faults model described in last section, i.e., $P_e = \frac{1}{N_p}\sum_{p=1}^{N_p} E(\frac{1}{N_o}\sum_{i=1}^{N_o}(\mathbf{I}_i^{(p)} - I_{id}^{(p)})^2)$.

Proof:

Hidden-layer neurons output o_j(j=1,…,N_h) is determined by $\mathbf{o}_j = \mathbf{Z}(\sum_{m=1}^{N_1}(\mathbf{w}_{jm}^{(1)} X_m))$. $\mathbf{w}_{jm}$ and $\mathbf{Z}$ in this equation are independently distributed if input errors are ignored. X_m is a constant, so $\mathbf{o}_j$ is independently distributed.

We add n additional redundancy to hidden-layer neuron k, then there are n+1 neurons in this hidden-layer neuron group. We use $\mathbf{o}_{k(l)}$ to denote neuron l in this hidden-layer neuron group, $\mathbf{w}_{ik(l)}$ to denote the weight linking this neuron to output layer neuron i. Obviously the expectation and variance of hidden-layer neuron output are of the same as the original one, i.e., $E\mathbf{o}_{k(l)}=E\mathbf{o}_k$, $D\mathbf{o}_{k(l)}=D\mathbf{o}_k$. Before replication $\mathbf{w}_{ik} = W_{ik}(1+\Delta)\mathbf{X}$, after replication $\mathbf{w}_{ik(l)} = \frac{1}{n}W_{ik}(1+\Delta)\mathbf{X}$, so $E\,\mathbf{w}_{ik(l)} = \frac{1}{n}E\,\mathbf{w}_{ik}$, $D\,\mathbf{w}_{ik(l)} = \frac{1}{n^2}D\,\mathbf{w}_{ik}$, and every $\mathbf{w}_{ik(l)}$, $\mathbf{o}_{k(l)}$ are independently distributed.

We first consider the network when n=1:

Before redundancy on hidden-layer neuron k (for convenience, we use w_{ij} instead of $w_{ij}^{(2)}$)

$E(\mathbf{I}_i - I_{id})^2 = E(\sum_{j\neq k}\mathbf{w}_{ij}\mathbf{o}_j + \mathbf{w}_{ik}\mathbf{o}_k - I_{id})^2$, define $\mathbf{T} = \sum_{j\neq k}\mathbf{w}_{ij}\mathbf{o}_j - I_{id}$, then we have

$$E(\mathbf{I}_i - I_{id})^2 = E\,\mathbf{T}^2 + 2E\,\mathbf{T}\,E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k + E(\mathbf{w}_{ik}^2\mathbf{o}_k^2) \quad (3)$$

After redundancy,

$$\begin{aligned} E(\mathbf{I}_i' - I_{id})^2 &= E(\sum_{j\neq k}\mathbf{w}_{ij}\mathbf{o}_j + \mathbf{w}_{ik(1)}\mathbf{o}_{k(1)} + \mathbf{w}_{ik(2)}\mathbf{o}_{k(2)} - I_{id})^2 \\ &= E\,\mathbf{T}^2 + 2E\,\mathbf{T}(E\,\mathbf{w}_{ik(1)}\mathbf{o}_{k(2)} + E\,\mathbf{w}_{ik(2)}\mathbf{o}_{k(2)}) \\ &\quad + E\,\mathbf{w}_{ik(1)}^2\,E\,\mathbf{o}_{k(1)}^2 + E\,\mathbf{w}_{ik(2)}^2\,E\,\mathbf{o}_{k(2)}^2 \\ &\quad + 2E\,\mathbf{w}_{ik(1)}\,E\,\mathbf{o}_{k(2)}\,E\,\mathbf{w}_{ik(2)}\,E\,\mathbf{o}_{k(2)} \\ &= E\,\mathbf{T}^2 + 2E\,\mathbf{T}\,E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k + \frac{1}{2}E(\mathbf{w}_{ik}^2\mathbf{o}_k^2) \\ &\quad + \frac{1}{2}(E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k)^2. \end{aligned} \quad (4)$$

Because of $E(\mathbf{w}_{ik}{}^2\mathbf{o}_k{}^2) > (E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k)^2$, comparing equation (3) with (4), we know that $E(\mathbf{I}_i - I_{id})^2$ decrease after redundancy.

And when $n \geq 2$:

After redundancy:

$$E(\mathbf{I}_i - I_{id})^2 = E(\mathbf{T} + \sum_{l=1}^{n} \mathbf{w}_{ik(l)}\mathbf{o}_{k(l)})^2$$

$$= E\,\mathbf{T}^2 + 2E\,\mathbf{T}\,E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k + E(\sum_{l=1}^{n} \mathbf{w}_{ik(l)}\mathbf{o}_{k(l)})^2$$

$$= E\,\mathbf{T}^2 + 2E\,\mathbf{T}\,E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k + \frac{1}{n}E(\mathbf{w}_{ik}{}^2\mathbf{o}_k{}^2)$$

$$+ \frac{n-1}{n}(E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k)^2.$$

It is apparent that $E(\mathbf{I}_i - I_{id})^2$ also decrease after redundancy, and $E(\mathbf{I}_i - I_{id})^2$ decreases with n increases.

For all i=1,…,N_o and all samples p=1,…,N_p the same conclusion can be drawn, So our proposition is tenable.

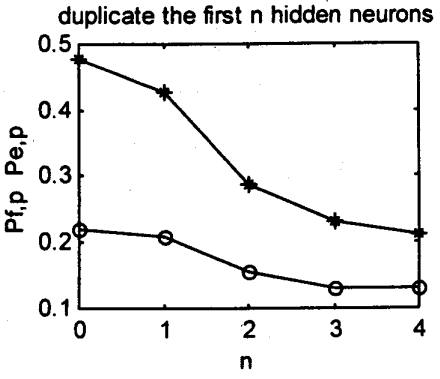
End of proof.

Inference 1: If every hidden-layer neuron is replicated with n_i ($n_i \geq 0$, $\sum_i n_i \neq 0$) additional ones using the architecture shown in Fig 1, the global faulty error will decrease. And the global faulty error is a monotonously degressive function to each n_i.

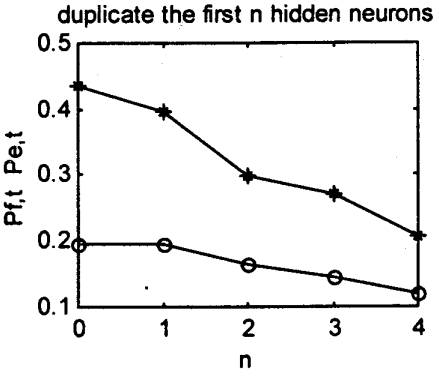
From theorem 1 and inference 1 we know that if a redundant network is formed with the redundancy architecture in Fig 1, the global faulty error will decrease. And for a single hidden-layer neuron, the more redundancy number, the more global faulty error will decrease. So neural networks fault tolerance can be gained though hidden layer neuron redundancy, the performance of neural networks will enhance greatly under universal faults.

We do simulation experiments with a prediction neural network to verify the conclusion above. This neural network is a single hidden-layer MLP, used to analyze gas spectrum and predict the concentration of gas CO. The inputs are spectrum altitude of 21 dimensions in a certain wave band, the output is a scalar of CO concentration. There are 4 hidden-layer neurons. The activation functions of hidden and output layer are sigmoid. The input and output value are united to [-1, 1]. Total samples number is 699, we take out 350 from them as training samples, 175 from them as validation samples and 174 from them as test samples. The training algorithm is adaptive BP algorithm with momentum term. We use a PC installed with Windows98. And we use MATLAB5.3 as our simulation platform.

The following parameters of fault model are adopted: the relative weight error $\Delta \sim N(0, 0.02)$, the probability of weight s-a-0 is 0.05, i.e., $P\{\mathbf{X}=0\}=0.05$, the probabilities of hidden-layer neuron s-a-1 and s-a-(-1) are the same, $P\{\mathbf{Z}(O_i)=-1\} = P\{\mathbf{Z}(O_i)=1\}=0.05$.



(a) training samples

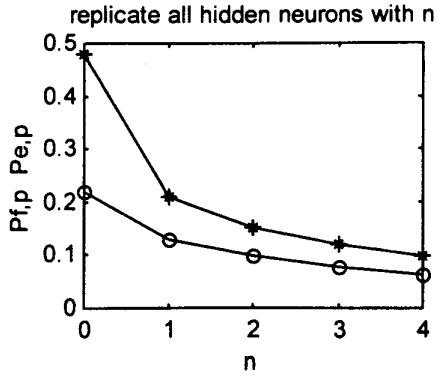


(b) test samples

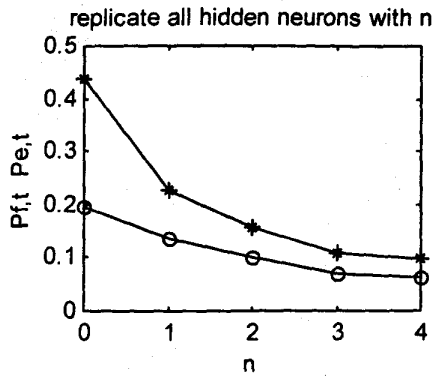
Fig 2 Results of duplicating each neuron of the n first hidden-layer neurons ($0 \leq n \leq 4$)

—*—*—*— $P_{e,}$ —o—o—o— P_f

We add one additional redundant neuron to each hidden-layer neuron group of the n first neurons in hidden-layer($0 \leq n \leq 4$). The global faulty error $P_{e,p}$ and output error $P_{f,p}$ of training samples and test samples were calculated respectively. And $P_{e,t}$ and output error $P_{f,t}$ of test samples were also calculated respectively. Each simulation were done for 1000 times, the mean value was adopted. The results are shown in Fig 2.



(a) training samples



(b) test samples

Fig 3 Results of adding n redundancy to each hidden-layer neuron

——*— P_e; —o—o—o— P_f

Then we add n redundant neurons to every hidden-layer neuron, calculate $P_{e,p}$, $P_{f,p}$, $P_{e,t}$ and $P_{f,t}$ shown in Fig 3.

From the results above we can see that the global faulty error decrease when the redundancy increase, and the MSE of output value of the network with Sigmoid output function also decrease. This verified **Theorem 1** and **Inference 1**. We can add redundant neurons to hidden-layer to increase the fault tolerance of this MLP and the more redundant neurons added the more fault tolerance increased.

IV. A PRACTICAL REDUNDANCY METHOD

Different neuron redundancy gives different contribution to fault tolerance. From **Fig 2** we can see that duplicating the second hidden-layer neuron gives most significant contribution. Some neurons are very "significant", so giving redundancy to these neurons is very effective. While redundancy to some other neurons make less effect. In practical applications, the total redundant neurons number is limited. In order to gain most effective fault tolerance in the limited redundancy number, we must select the most "significant" neurons to replicate.

From the proof of **Theorem 1** we know that the global faulty error reduced while adding an additional redundant neuron to hidden-layer neuron k is $\frac{1}{N_p * N_o}\sum_{i,p}\frac{1}{2}\left[E(\mathbf{w}_{ik}^2 \mathbf{o}_k^{(p)2}) - (E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k^{(p)})^2\right]$. For general networks and samples it is very difficult to evaluate it. We give some simplification. Let's assume every $\mathbf{o}_k^{(p)}$ has an identical distribution, use $\mathbf{O}$ to denote it, then

$$\frac{1}{N_p * N_o}\sum_{i,p}\frac{1}{2}\left[E(\mathbf{w}_{ik}^2 \mathbf{o}_k^{(p)2}) - (E\,\mathbf{w}_{ik}\,E\,\mathbf{o}_k^{(p)})^2\right]$$

$$= \frac{1}{2N_o}\sum_i \left\{W_{ik}^2\, E[(1+\Delta)\mathbf{XO}]^2 - W_{ik}^2[E((1+\Delta)\mathbf{XO})]^2\right\}$$

$$= \frac{1}{2N_o} D[(1+\Delta)\mathbf{XO}]\sum_i W_{ik}^2$$

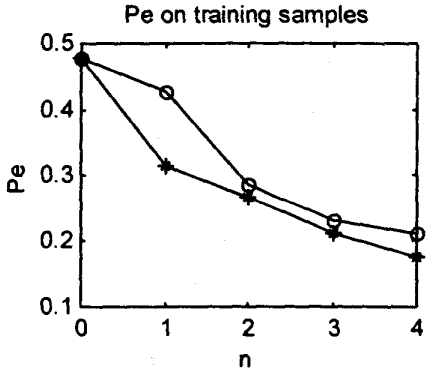
Therefore we can determine the "significance" of each hidden-layer neuron by calculating $\sum_i W_{ik}^2$ of neuron k. Then we have a optimized and easy-implementing hidden-layer neurons redundancy algorithm:

(1) Set the maximal redundancy number R, initial redundancy number R(0)=0, initial step t=0;

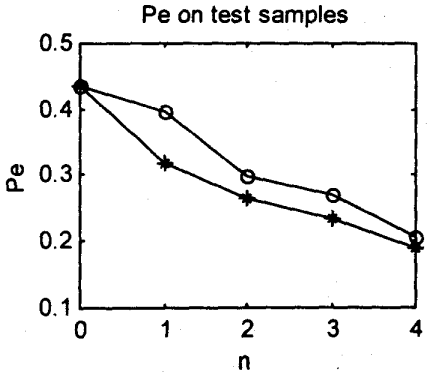
(2) Calculate $\sum_i W_{ik}^2$ of each hidden-layer neuron k, select the maximal one among them, give this neuron one additional redundant neuron, set R(t)=R(t)+1;

(3) If R(t)>R, end; Otherwise set t=t+1, go to (2).

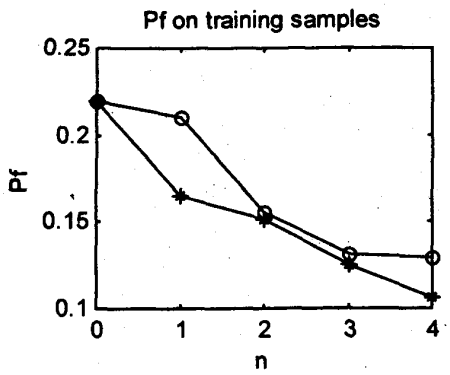
This algorithm may not give a global optimal redundancy result, but it has surely been optimized and is easy to implement. We still use the example in last section to do our contrast simulation. We duplicate each neuron of the n first hidden-layer neurons ($0 \le n \le 4$) to compare with the optimized algorithm above(R=n). Then we calculate P_f, P_e respectively. The results are shown in **Fig 4**.



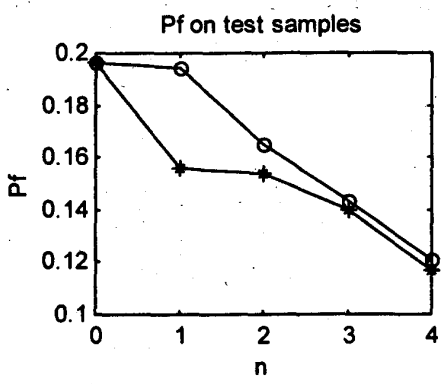
(a) The global faulty error of training samples



(b) The global faulty error of test samples



(c) The MSE of outputs of training samples



(d) The MSE of outputs of test samples

Fig 4 The comparison of optimized redundancy and normal redundancy
—*—*—*— Optimized redundancy algorithm
—o—o—o— Normal redundancy algorithm

In each step of the optimized algorithm, we select a optimized neuron to replicate. So the result is better than the normal algorithm without redundant neuron select. Using this algorithm, hidden-layer neuron redundancy can be done quickly and effectively. The fault tolerance of network will increase greatly in a limited redundancy number. And it is also very practical because of the easy-implement of the algorithm.

V. CONCLUSION

Redundancy on hidden-layer neurons has proven effective in the fault tolerance of neural networks. But the researches under universal faults are limited. Some researchers draw the conclusion that the fault tolerance of network is inversely proportional to the number of hidden-layer neurons in [4]. This conclusion makes the hidden-layer redundancy method suspicious. So we analyze the hidden-layer redundancy method under universal faults.

We prove that the global faulty error will decrease with the increase of redundancy number using the redundant architecture we presented. Thus the fault tolerance increased. The basis of the conclusion drawn in [4] is that the additional hidden-layer neurons are not redundant neurons of the other neurons. In fact, they are even not redundant neurons. The architecture is different from our redundancy architecture. So they have drawn a contrary conclusion. The redundancy architecture we presented can maintain the performance of the network when no fault happens. The output error and generalization of the redundant network are of the same as the original one. When faults happen, the global faulty error can greatly decrease.

We also present the practical hidden-layer redundancy algorithm based on our theoretical analysis. In each step of the algorithm, the hidden-layer neuron which is to be replicated is chosen based on the standard we presented. The result is better than that of the normal method in which no neuron is chosen. The redundancy algorithm is effective and easy to implement.

VI. ACKNOWLEDGEMENT

We would like to acknowledge the National Science Foundation of China and Doctoral Dissertation Foundation of Tsinghua University for their support of our endeavors in this area.

VII. REFERENCES

[1] D.S. Phatak and I. Koren, "Complete and partial fault tolerance of feedforward Neural Nets", *IEEE Trans. on Neural Networks*, vol. 6, no. 2, Feb. 1995, pp. 446~456.

[2] D.S. Phatak and I. Koren, "Fault tolerance of feedforward Neural Nets for classification tasks" *in Proceedings of the 1992 International Joint Conference of Neural Networks*, pp. II-386~391

[3] L. Xu, D. Hu, "A new component redundancy method in MLP's fault tolerance", *In Proceedings of the 1999 International Conference of Electronics Measurement and Instrument*, pp. I-72~76

[4] T. Zhang, *Theory and application of fault-tolerance analysis and design of artificial Neural Networks*. [Doctoral Dissertation](Chinese). Beijing; 1999. pp. 20~38

[5] P. Kerlinzin and P. Refregier, "Theoretical investigation of the robustness of multilayer perceptrons: Analysis of the linear case and extension to nonlinear networks", *IEEE Trans. on Neural Networks*, vol. 6, no. 3. Mar. 1995, pp. 560~571

[6] B.S. Arad and A. El-Anway, "On Fault Tolerant Training of Feedforward Neural Networks", *Neural Networks*, vol. 10, no. 3, Mar. 1997, pp. 539~557

[7] D. Deodhare, M. Vidyasagar and S.S. Keerthi, "Synthesis of fault-tolerant feedforward Neural networks using minimax optimization", *IEEE Trans. on Neural Networks*, vol. 9, no. 5, May. 1998, pp. 891~900