Neurocomputing ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Letters

Prediction error of a fault tolerant neural network

John Sum [a,*], Andrew Chi-Sing Leung [b,1]

[a] Institute of E-Commerce, National Chung Hsing University, Taichung 402, Taiwan
[b] Department of Electronic Engineering, City University of Hong Kong, Kowloon Tong, KLN, Hong Kong

ARTICLE INFO

Article history:
Received 24 August 2006
Received in revised form
12 May 2008
Accepted 20 May 2008
Communicated by J. Zhang

Keywords:
Fault tolerant neural networks
Prediction error
RBF network

ABSTRACT

Prediction error is a powerful tool that measures the performance of a neural network. In this paper, we extend the technique to a kind of fault tolerant neural networks. Considering a neural network with multiple-node fault, we derive its generalized prediction error. Hence, the effective number of parameters of such a fault tolerant neural network is obtained. The difficulty in obtaining the mean prediction error is discussed. Finally, a simple procedure for estimation of the prediction error is empirically 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 in VLSI [19]. 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 [3,23] together with random node deletion and addition [7] during training is one common approach. Adding network redundancy by replicating hidden nodes/layers after trained [9,21,26], adding weight decay regularizer [7] and hard bounding the weight magnitude during training [4] are other techniques that have also been proposed in the literature. In accordance with simulation results, all these heuristic techniques have demonstrated that the trained networks are 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. [20] defined the problem as a minimax problem in which the objective function to be minimized is the maximum of the mean square errors over all possible faulty networks. Deodhare et al. [8] formulated the problem by defining the objective function to be minimized as the maximum square error over all possible faulty networks 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 possible faulty nodes cannot be larger than one. Simon and El-Sherief [24] and Phatak and Tcherner [22] 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 a fault-free network while the second term is the ensemble average of the mean square errors over all possible faulty networks.

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-node fault will become impractical. In view of the lacking of a simple objective function to formalize multiple-node fault and the lacking of an understanding of the relation between fault tolerant and generalization, Leung and Sum [11] have recently derived a simple objective function and yet another regularizer from Kullback–Leibler divergence for robust training a neural network that can optimally tolerate multiple-node fault.

In this paper, we extend the idea elucidated in [11] by deducing the mean prediction error equation for such a fault tolerant neural

* Corresponding author.
E-mail addresses: pfsum@nchu.edu.tw, pfsum@yahoo.com.hk (J. Sum), eeleungc@cityu.edu.hk (A.C.-S. Leung).

[1] The work was supported by a research grant from City University of Hong Kong (7002108).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.05.009

network model being attained. As it is believed that prediction error is an alternative measure for the performance of a neural network [15,25] and for neural network pruning [12–14]. The rest of the paper will be organized as follows. The next section will define what a node fault tolerant neural network is and present an objective function derived in [11] 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. The estimation of the prediction error for small sample size is discussed in Section 6. Then, we conclude the paper in Section 7.

2. Node fault tolerant neural network

Throughout the paper, we are given a training data set $\mathscr{D}_T = \{(x_k, y_k)\}_{k=1}^N$, where x_k and y_k are the kth input and output sample of a stochastic system, respectively. We assume that the data set $\mathscr{D}_T$ is generated by a stochastic system [2,6], given by

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

where $f(\cdot)$ is the unknown deterministic part of the stochastic system and e_k's are the random measurement noise. The noise e_k's are independent zero-mean Gaussian random variables with variance equal to S_e. Hence, the output y of the stochastic system is a dependent random variable governed by the input x. The behavior of the system is denoted by the conditional probability $P_0(y|x)$, which is the probability density function of y given the input x. Our problem is to construct a neural network to approximate the unknown mapping $f(\cdot)$ based on the data set $\mathscr{D}_T$. A radial basis function (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 $i = 1,2,\ldots,M$ are the RBFs given by

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

c_i's are the RBF centers and the positive parameter $\sigma > 0$ controls the width of the RBFs. Without loss of generality, we assume that $c_i \in R$ for all i. A network given by (2) is called a *fault-free RBF network*.

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

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

If $\beta_i = 1$, the ith node is operating normally. If $\beta_i = 0$, the ith 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 $i = 1,2,\ldots,M$ and $\beta_1,\ldots,\beta_M$ are independent random variables. Eq. (4) define a *faulty RBF network*.

In sequel, the unknown deterministic system $f(\cdot)$ is approximated by the RBF network $\hat{f}_\beta(x,\theta)$. Based on the stochastic model in neural networks [2], the stochastic system, given by (1), is approximated by

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

where e is a mean zero Gaussian noise defined in (1). The behavior of this stochastic faulty RBF network is described by a conditional probability $P(y|x,\theta,\beta)$. Let $\tilde{\theta} = (\beta_1\theta_1,\ldots,\beta_M\theta_M)$. Now, the conditional probability of a faulty RBF network 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 distribution of the input x and the output y of the stochastic system (1) is given by

$$P_0(x,y) = P_0(y|x)P_0(x). \tag{6}$$

For the stochastic RBF network (5), the joint probability distribution is given by

$$P(x,y|\tilde{\theta}) = P(y|x,\tilde{\theta})P_0(x). \tag{7}$$

To measure the discrepancy between the two distributions (the faulty RBF network and the data set (the stochastic system)), we use the Kullback–Leibler divergence [10], given by

$$D(P_0\|P_{\tilde{\theta}}) = \iint P_0(x,y)\log\frac{P_0(x,y)}{P(x,y|\tilde{\theta})}\,dx\,dy. \tag{8}$$

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

$$\bar{D}(P_0\|P_\theta) = \int\left\{\iint P_0(x,y)\log\frac{P_0(x,y)}{P(x,y|\tilde{\theta})}\,dx\,dy\right\}P(\tilde{\theta}|\theta)\,d\tilde{\theta} \tag{9}$$

$$= \left\langle \iint P_0(x,y)\log\frac{P_0(x,y)}{P(x,y|\tilde{\theta})}\,dx\,dy\right\rangle_{\Omega_\beta}. \tag{10}$$

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

It can be shown [11] that minimizing $\bar{D}(P_0\|P_\theta)$ is equivalent to minimizing the following objective function:

$$E(\theta,p) = \frac{1}{N}\sum_{k=1}^{N} y_k^2 - 2(1-p)\frac{1}{N}\sum_{k=1}^{N} y_k\phi^{\mathrm{T}}(x_k)\theta + (1-p)\theta^{\mathrm{T}}\{(1-p)H_\phi + pG\}\theta, \tag{11}$$

$$H_\phi = \frac{1}{N}\sum_{k=1}^{N}\phi(x_k)\phi^{\mathrm{T}}(x_k),$$

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

where $\{(x_k,y_k)\}_{k=1}^N$ is the training data set and p is the node fault rate. Taking the first derivative of $E(\theta,p)$ with respect to θ and setting the derivative to zero, the corresponding optimal fault tolerant RBF will be given by

$$\hat{\theta} = (H_\phi + p(G - H_\phi))^{-1}\frac{1}{N}\sum_{k=1}^{N} y_k\phi(x_k). \tag{12}$$

Since H_ϕ and G are functions of $\phi(x_1),\ldots,\phi(x_N)$, $\hat{\theta}$ can be obtained as long as $\{x_k,y_k\}_{k=1}^N$ are given. Now, $\hat{f}_\beta(x,\hat{\theta})$ defines an optimal fault tolerant RBF network.

3. Mean prediction error

It should be noticed that minimizing the training square error does not mean that the network will perform well on an unseen test set. As mentioned by Moody [16,17], estimating the generalization performance from the training error is very important. It allows us not only to predict the performance of a trained network but also to select the model from various settings. It should be noticed that in the real situation data are very valuable and we may not have a test set for model selection. 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

Please cite this article as: J. Sum, A.C.-S. Leung, Prediction error of a fault tolerant neural network, Neurocomputing (2008), doi:10.1016/j.neucom.2008.05.009

AIC [1], GPE [16] or NIC [18]. For presentation clarity, a summary of the notations being used is depicted in Table 1.

Given the estimated weight vector $\hat{\theta}$ and an input x, the mean square error between the output of the stochastic system and the faulty network output is given by

$$\langle (y-\hat{f}_\beta(x,\theta))^2\rangle = y^2 - 2(1-p)y\phi^{\mathrm{T}}(x)\hat{\theta} + (1-p)\hat{\theta}^{\mathrm{T}}\{(1-p)H_\phi + pG\}\hat{\theta}. \quad (13)$$

Let $\{(x_k,y_k)\}_{k=1}^N$ and $\{(x'_k,y'_k)\}_{k=1}^{N'}$ be the training set and the testing set, respectively. The mean training error $E(\mathscr{D}_{\mathrm{T}}|\hat{\theta})$ and the mean prediction error $E(\mathscr{D}_{\mathrm{F}}|\hat{\theta})$ are given by

$$E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}) = \langle y^2\rangle_{\mathscr{D}_{\mathrm{T}}} - 2(1-p)\langle y\phi^{\mathrm{T}}(x)\hat{\theta}\rangle_{\mathscr{D}_{\mathrm{T}}} + (1-p)\hat{\theta}^{\mathrm{T}}\{(1-p)H_\phi + pG\}\hat{\theta}, \quad (14)$$

$$E(\mathscr{D}_{\mathrm{F}}|\hat{\theta}) = \langle y'^2\rangle_{\mathscr{D}_{\mathrm{F}}} - 2(1-p)\langle y'\phi^{\mathrm{T}}(x')\hat{\theta}\rangle_{\mathscr{D}_{\mathrm{F}}} + (1-p)\hat{\theta}^{\mathrm{T}}\{(1-p)H'_\phi + pG'\}\hat{\theta}, \quad (15)$$

where $H_\phi = (1/N)\sum_{k=1}^N \phi(x_k)\phi^{\mathrm{T}}(x_k)$, $H'_\phi = (1/N')\sum_{k=1}^{N'}\phi(x'_k)\phi^{\mathrm{T}}(x'_k)$

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

and

$$G' = \mathbf{diag}\left\{\frac{1}{N'}\sum_{k=1}^{N'} \phi_1^2(x'_k),\ldots,\frac{1}{N'}\sum_{k=1}^{N'} \phi_M^2(x'_k)\right\}.$$

Assuming that N and N' are large, $H'_\phi \approx H_\phi$, $G' \approx G$ and $\langle y^2\rangle_{\mathscr{D}_{\mathrm{T}}} \approx \langle y'^2\rangle_{\mathscr{D}_{\mathrm{F}}}$. So, the difference between $E(\mathscr{D}_{\mathrm{F}}|\hat{\theta})$ and $E(\mathscr{D}_{\mathrm{T}}|\hat{\theta})$ lies in the difference between their second terms.

Following the same technique as using in [15,18], we assume that there is a θ_0 such that

$$y_k = \theta_0^{\mathrm{T}}\phi(x_k) + e_k, \quad (16)$$

$$y'_k = \theta_0^{\mathrm{T}}\phi(x'_k) + e'_k, \quad (17)$$

where e_k's and e'_k's are independent zero-mean Gaussian random variables with variance equal to S_e. One should further note that $\hat{\theta}$ is obtained entirely by $\mathscr{D}_{\mathrm{T}}$, which is independent of $\mathscr{D}_{\mathrm{F}}$. Therefore, we can have

$$\langle y'\phi^{\mathrm{T}}(x')\hat{\theta}\rangle_{\mathscr{D}_{\mathrm{F}}} = \left(\frac{1}{N'}\sum_{k=1}^{N'} y'_k\phi^{\mathrm{T}}(x'_k)\right)\hat{\theta}. \quad (18)$$

The second term in $E(\mathscr{D}_{\mathrm{F}}|\hat{\theta})$ can thus be given by

$$-2(1-p)\langle y'\phi^{\mathrm{T}}(x')\hat{\theta}\rangle_{\mathscr{D}_{\mathrm{F}}} = -2(1-p)\left(\frac{1}{N'}\sum_{k=1}^{N'} y'_k\phi^{\mathrm{T}}(x'_k)\right)(H_\phi + p(G-H_\phi))^{-1} \times\left(\frac{1}{N}\sum_{k=1}^N y_k\phi(x_k)\right). \quad (19)$$

From (16) and (17), the second term in $E(\mathscr{D}_{\mathrm{F}}|\hat{\theta})$ becomes

$$-2(1-p)\theta_0^{\mathrm{T}}H_\phi((1-p)H_\phi + pG)^{-1}H_\phi\theta_0. \quad (20)$$

Using a similar method, the second term in $E(\mathscr{D}_{\mathrm{T}}|\hat{\theta})$ is given by

$$-2(1-p)\frac{S_e}{N}\mathbf{Tr}\{H_\phi((1-p)H_\phi + pG)^{-1}\} - 2(1-p)\theta_0^{\mathrm{T}}H_\phi((1-p)H_\phi + pG)^{-1}H_\phi\theta_0. \quad (21)$$

As a result, the difference between the mean prediction error and mean training error which is given by

$$E(\mathscr{D}_{\mathrm{F}}|\hat{\theta}) - E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}) = 2(1-p)\langle y\phi^{\mathrm{T}}(x)\hat{\theta}\rangle_{\mathscr{D}_{\mathrm{T}}} - 2(1-p)\langle y'\phi^{\mathrm{T}}(x')\hat{\theta}\rangle_{\mathscr{D}_{\mathrm{F}}}. \quad (22)$$

By (20) and (21), the mean prediction error is given as follows:

$$E(\mathscr{D}_{\mathrm{F}}|\hat{\theta}) = E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}) + 2\frac{S_e}{N}\mathbf{Tr}\{(1-p)H_\phi((1-p)H_\phi + pG)^{-1}\}. \quad (23)$$

Let

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

This parameter can be interpreted as the effective number of parameter of an RBF network of $(1-p)M$ number of nodes as the way in [16]. Therefore, the true S_e can be approximated by the following equation:

$$S_e \approx \frac{N}{N-M_{\mathrm{eff}}}E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}).$$

The prediction error can then be approximated by

$$E(\mathscr{D}_{\mathrm{F}}|\hat{\theta}) = \frac{N+M_{\mathrm{eff}}}{N-M_{\mathrm{eff}}}E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}). \quad (24)$$

To use this approximation, the simulation to be conducted is a bit not as usual. Suppose we have a set of measure data, $\mathscr{D}_{\mathrm{T}}$. After a robust network is thus obtained by Eq. (12), as many as possible faulty RBF networks are generated. Their average training error is thus obtained by simulation. This average value is regarded as $E(\mathscr{D}_{\mathrm{T}}|\hat{\theta})$ that is used for predicting $E(\mathscr{D}_{\mathrm{F}}|\hat{\theta})$ based on Eq. (24) immediately.

4. Estimation of MPE

Given a trained network, obtaining the true value of $E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}(p,\cdot))$ is very expensive. This is because the number of faulty networks follows a binomial probability distribution. For example, for a trained network with 50 RBF nodes and five faulty nodes, the number of possible faulty networks with five faulty nodes is equal to $50!/(5!\times 45!)$. Hence examining all faulty networks for all possible faulty node numbers is nearly impossible. So, we only approximate the average training error by the sampling average.

If S_e and p are given, a number of faulty networks are generated uniformly random. The same set of training data is thus fed into the networks. The average value of the training errors will thus be used as an approximation of $E(\mathscr{D}_{\mathrm{T}}|\hat{\theta})$. It is equivalent to approximate the prediction error by the following equation:

$$E(\mathscr{D}_{\mathrm{F}}|\hat{\theta}) \approx E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}) + 2\frac{S_e}{N}\mathbf{Tr}\{(1-p)H_\phi((1-p)H_\phi + pG)^{-1}\}, \quad (25)$$

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

$$E(\mathscr{D}_{\mathrm{F}}|\hat{\theta}) \approx \frac{N+M_{\mathrm{eff}}}{N-M_{\mathrm{eff}}}E(\mathscr{D}_{\mathrm{T}}|\hat{\theta}). \quad (26)$$

Table 1
Key notations

Notation	Description
$\mathscr{D}_{\mathrm{T}}$	Training data set
$\mathscr{D}_{\mathrm{F}}$	Testing data set
p	Fault rate—probability that a node will be failure
M	Number of radial basis functions (nodes)
$\hat{\theta}$	Weight vector obtained by Eq. (12)
$\langle\cdot\rangle$	Expectation operator
$E(\mathscr{D}_{\mathrm{T}}\|\hat{\theta})$	Mean square training errors of the faulty network
$E(\mathscr{D}_{\mathrm{F}}\|\hat{\theta})$	Mean prediction error of the faulty network

Please cite this article as: J. Sum, A.C.-S. Leung, Prediction error of a fault tolerant neural network, Neurocomputing (2008), doi:10.1016/j.neucom.2008.05.009

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 networks in accordance with the fault rate p.
(4) Obtain the mean training error for each faulty network.
(5) The average mean training error is evaluated by the sample average of all these mean training errors.
(6) Estimate $E(\mathscr{D}_F|\hat{\theta})$ either by Eq. (25) or (26).

The faulty network specified in Step (3) is realized by independently setting each of the weights to zero with probability p, so as to mimic a multiple-nodes fault effect.

5. Experimental results

To validate the usefulness of the mean prediction error derived, a simulated experiment has been carried out. The first experiment demonstrates the viability of the mean prediction error deduced in approximating the actual prediction error. The second experiment shows how the deduced mean prediction error can be applied to select the width of the RBFs.

5.1. Function approximation

In this experiment, 20 RBF networks are generated to approximate a simple noisy function

$$f(x) = \tanh(x) + e \quad \text{where } e \sim \mathcal{N}(0, 0.01)$$

a mean zero Gaussian 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. Twenty independent training data sets are generated for each of the RBF networks. Each training set consists of 50 training data, with inputs are uniformly randomly generated in the range $[-4, 4]$ and noises are randomly generated in accordance with Gaussian distribution. An extra data set consisting of 100 data is also generated as the testing set for the evaluation of prediction error.

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 $0.01, 0.02, 0.03, \ldots, 0.2$. For each p, $\hat{\theta}$ is obtained after H_ϕ and G have been calculated. Then 100 faulty networks are generated and their training errors are measured. With this setup, we have generated 20×100 faulty networks.

The estimated mean prediction error $E(\mathscr{D}_F|\hat{\theta})$ is estimated by Eq. (25). Finally, the actual prediction error is obtained simply by feeding the testing data set to these 100 faulty networks again and taking their average. The actual prediction error against the estimated prediction error for different values of p is thus shown in Fig. 1. The solid line, $y = x$, is used for reference. It is clearly that the points lie symmetrically along the solid straight line. For reference, Fig. 2 shows the results comparing the training error and actual mean prediction error. It should be noted that a shift of the data points to left-hand side of the figure could be found.

5.2. Selection of RBF width

Selection of an appropriate value for the RBF width (i.e. σ) is always a crucial step leading the success of application. In this experiment, we make use of a nonlinear time series that is presented in [5] as an example and demonstrate how the deduced mean prediction error can be applied to select a good value of σ for a fault tolerant RBF.

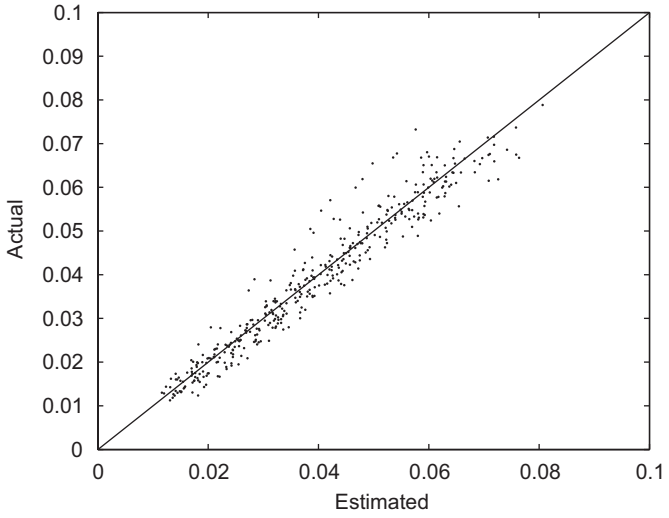

Fig. 1. Actual MPE versus estimated MPE.

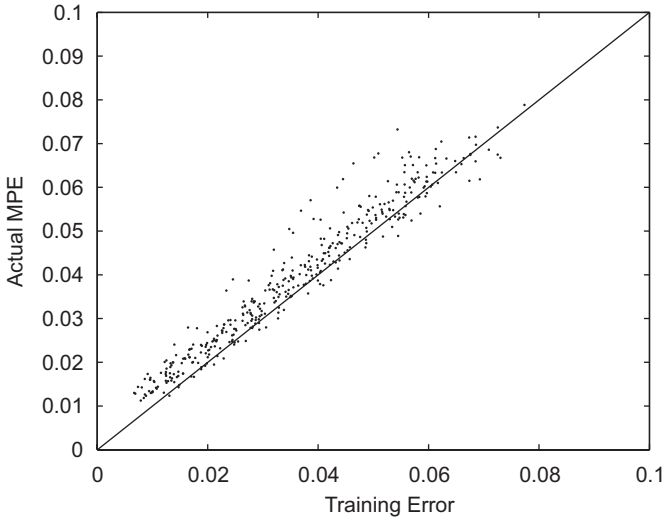

Fig. 2. Actual MPE versus training error.

The nonlinear time series is defined as follows:

$$\begin{aligned} y_k &= (0.8 - 0.5\exp(-y_{k-1}^2))y_{k-1} \\ &\quad - (0.3 + 0.9\exp(-y_{k-1}^2))y_{k-2} \\ &\quad + 0.1\sin(\pi y_{k-1}) + e_k, \end{aligned} \tag{27}$$

where e_k is a mean zero Gaussian noise with variance equals to 0.04.

One thousand samples $(y_1, y_2, \ldots, y_{1000})$ are generated by using Eq. (27) and setting $y_{-1} = y_0 = 0.1$. The first 500 samples are used for training and the other 500 samples are used for testing. We consider an RBF as a two input one output nonlinear model defined as follows:

$$\begin{aligned} y_k &= \hat{f}(y_{k-1}, y_{k-2}, \theta, \sigma) + e_k \\ &= \sum_{i=1}^{M} \theta_i \phi_i(y_{k-1}, y_{k-2}, \sigma) + e_k, \end{aligned}$$

where σ specifies the width of the basis functions and M is the number of basis functions being included in the network.

Please cite this article as: J. Sum, A.C.-S. Leung, Prediction error of a fault tolerant neural network, Neurocomputing (2008), doi:10.1016/j.neucom.2008.05.009

Nine different values of σ are examined: 0.01, 0.04, 0.09, 0.16, 0.25, 0.36, 0.49, 0.64 and 0.81. For each value of σ, we apply LROLS method [5] to select the significant samples from the training samples to be the centers of the basis functions. As a result, nine different sets of significant samples are generated to constitute nine different RBF networks.

Given a value for p (the fault rate), the output weights of an RBF network can thus be obtained by Eq. (12). Its performance in terms of average mean training error $E(\mathscr{D}_T|\hat{\theta})$, average mean testing error $E(\mathscr{D}_F|\hat{\theta})$ and mean prediction error, Eq. (25), can be evaluated by the following procedure:

(1) Given p and $\hat{\theta} = (\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_M)^T$.
(2) For $j = 1, 2, \ldots,$ Run.
  (2.1) Generate M uniformly random numbers, say $U_1, U_2, \ldots, U_M$.
  (2.2) For $i = 1, 2, \ldots, M$, set $\beta_i = 1$ if $U_i \leqslant p$ and zero otherwise.
  (2.3) Generate a fault model $\tilde{\theta}$, in which $\tilde{\theta}_i = \beta_i \hat{\theta}_i$ for all $i = 1, \ldots, M$.
  (2.4) $E_{\text{train}}(j)$ is the mean training error.
  (2.5) $E_{\text{test}}(j)$ is the mean testing error.
  (2.6) Evaluate $PE(j)$ by Eq. (25).
(3) $E(\mathscr{D}_T|\hat{\theta}) = (1/\text{Run})\sum_{j=1}^{\text{Run}} E_{\text{train}}(j)$.
(4) $E(\mathscr{D}_F|\hat{\theta}) = (1/\text{Run})\sum_{j=1}^{\text{Run}} E_{\text{test}}(j)$.
(5) Mean prediction error $= (1/\text{Run})\sum_{j=1}^{\text{Run}} PE(j)$.

In our experiment, Run is set to 6000. The results for $p = 0.05$, 0.10, 0.15 and 0.20 are depicted in Table 2.

In the table, the data in bold face are the smallest average error within the column. It is readily found that the value of σ selected based on $E(\mathscr{D}_T|\hat{\theta})$ is either 0.01 or 0.04. The value of σ selected based on $E(\mathscr{D}_F|\hat{\theta})$ is 0.36, and the value selected based on Eq. (25) is 0.16. The values being selected based on training error will lead to poor performance. While the value being selected based on our approach can lead to an RBF with performance similar to that of the best choice: (i) 0.0695 versus 0.0682 for $p = 0.05$, (ii) 0.0789 versus 0.0771 for $p = 0.10$, (iii) 0.0889 versus 0.0864 for $p = 0.15$ and (iv) 0.0989 versus 0.0957 for $p = 0.20$. The percentage is less than 4%.

Table 2
Results for the RBF width selection problem

σ	$E(\mathscr{D}_T\|\hat{\theta})$	$E(\mathscr{D}_F\|\hat{\theta})$	Eq. (25)
$p = 0.05$			
0.01	**0.0336**	0.1797	0.0648
0.04	0.0419	0.0875	0.0562
0.09	0.0468	0.0786	0.0538
0.16	0.0475	0.0695	**0.0523**
0.25	0.0524	0.0698	0.0560
0.36	0.0518	**0.0682**	0.0547
0.49	0.0555	0.0734	0.0580
0.64	0.0545	0.0687	0.0566
0.81	0.0568	0.0718	0.0588
$p = 0.10$			
0.01	**0.0471**	0.1903	0.0754
0.04	0.0506	0.0962	0.0634
0.09	0.0555	0.0903	0.0617
0.16	0.0554	0.0789	**0.0596**
0.25	0.0605	0.0795	0.0636
0.36	0.0590	**0.0771**	0.0616
0.49	0.0641	0.0847	0.0662
0.64	0.0631	0.0795	0.0649
0.81	0.0653	0.0825	0.0670
$p = 0.15$			
0.01	0.0607	0.2019	0.0868
0.04	**0.0592**	0.1056	0.0708
0.09	0.0646	0.1018	0.0703
0.16	0.0635	0.0889	**0.0674**
0.25	0.0678	0.0886	0.0707
0.36	0.0664	**0.0864**	0.0687
0.49	0.0725	0.0954	0.0744
0.64	0.0716	0.0903	0.0733
0.81	0.0745	0.0936	0.0760
$p = 0.20$			
0.01	0.0745	0.2138	0.0987
0.04	**0.0681**	0.1157	0.0789
0.09	0.0738	0.1135	0.0790
0.16	0.0717	0.0989	**0.0752**
0.25	0.0759	0.0989	0.0785
0.36	0.0739	**0.0957**	0.0760
0.49	0.0808	0.1060	0.0826
0.64	0.0791	0.0994	0.0806
0.81	0.0821	0.1028	0.0835

6. Discussion

The success of the estimation of the mean prediction errors relies very much on the assumption that $H'_\phi \approx H_\phi$ and $G' \approx G$. It happens when the number of samples is large enough, i.e. N and N' are large. For small number of samples, the mean prediction errors would be given by the following equation:

$$E(\mathscr{D}_F|\hat{\theta}) \approx E(\mathscr{D}_T|\hat{\theta}) + (1-p)\hat{\theta}^T((1-p)\Delta H_\phi + p\Delta G)\hat{\theta} - 2(1-p)\theta_0^T \Delta H_\phi((1-p)H_\phi + pG)^{-1}H_\phi\theta_0 + 2\frac{S_e}{N}\mathbf{Tr}\{(1-p)H_\phi((1-p)H_\phi + pG)^{-1}\}. \quad (28)$$

Here $\Delta H_\phi = H'_\phi - H_\phi$ and $\Delta G = G'_\phi - G_\phi$. In this equation, one should note that it requires information other than the training data to evaluate the factors ΔH_ϕ and ΔG. However, these information are assumed to be unavailable during time of training. As our objective is to estimate the performance of an RBF network right after the network has been trained, Eq. (28) is not suitable for application.

Statistical analysis on the properties of ΔH_ϕ and ΔG might help. Nice approximations to these factors might be obtained and accurate estimation of the mean prediction error for a fault tolerant RBF could be deduced. We leave this problem, in regard to small sample size situation, open for further investigation.

7. Conclusion

Following the objective function we have derived in [11], 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 [11]. We have demonstrated how to use the prediction error to select the width for a fault tolerant RBF network. Finally, the estimation of the mean prediction error in small sample size situation is discussed. Approach to refine the equation is suggested for future research.

Please cite this article as: J. Sum, A.C.-S. Leung, Prediction error of a fault tolerant neural network, Neurocomputing (2008), doi:10.1016/j.neucom.2008.05.009

Acknowledgments

The authors would like to thank for the reviewers for their valuable comments. In particular, one reviewer has addressed the problem of our estimation in small sample size. The work was supported by a research grant from City University of Hong Kong (7002108).

References

[1] H. Akaike, A new look at the statistical model identification, IEEE Trans. Autom. Control 19 (1974) 716–723.
[2] S.I. Amari, N. Murata, K.R. Muller, M. Finke, H.H. Yang, Asymptotic statistical theory of overtraining and cross-validation, IEEE Trans. Neural Networks 8 (1997) 985–996.
[3] G. Bolt, Fault tolerant in multi-layer perceptrons, Ph.D. Thesis, University of York, UK, 1992.
[4] S. Cavalieri, O. Mirabella, A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks, Neural Networks 12 (1999) 91–106.
[5] S. Chen, Local regularization assisted orthogonal least squares regression, Neurocomputing 69 (4–6) (2006) 559–585.
[6] S. Chen, X. Hong, C.J. Harris, P.M. Sharkey, Sparse modelling using orthogonal forward regression with press statistic and regularization, IEEE Trans. Systems Man Cybern. Part B (2004) 898–911.
[7] C.T. Chiu, et al., Modifying training algorithms for improved fault tolerance, in: ICNN94, vol. I, 1994, pp. 333–338.
[8] D. Deodhare, M. Vidyasagar, S. Sathiya Keerthi, Synthesis of fault-tolerant feedforward neural networks using minimax optimization, IEEE Trans. Neural Networks 9 (5) (1998) 891–900.
[9] M.D. Emmerson, R.I. Damper, Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application, IEEE Trans. Neural Networks 4 (1993) 788–793.
[10] S. Kullback, Information Theory and Statistics, Wiley, New York, 1959.
[11] C.S. Leung, J. Sum, A fault tolerant regularizer for RBF networks, IEEE Trans. Neural Networks 19 (3) (2008) 493–507.
[12] C.S. Leung, P.F. Sum, A.C. Tsoi, L.W. Chan, Several aspects of pruning methods in recursive least square algorithms for neural networks, in: K. Wong, I. King, D.Y. Yeung (Eds.), Theoretical Aspects of Neural Computation: A Multi-disciplinary Perspective, Lecture Notes in Computer Science, Singapore Pvt. Ltd., Springer, Berlin, 1997, pp. 71–80.
[13] C.S. Leung, K.W. Wong, J. Sum, L.W. Chan, On-line training and pruning for RLS algorithms, Electron. Lett. 32 (23) (1996) 2152–2153.
[14] C.S. Leung, K.W. Wong, P.F. Sum, L.W. Chan, A pruning method for recursive least squared algorithm, Neural Networks 14 (2) (2001) 147–174.
[15] C.S. Leung, G.H. Young, J. Sum, W.K. Kan, On the regularization of forgetting recursive least square, IEEE Trans. Neural Networks 10 (6) (1999) 1842–1846.
[16] J.E. Moody, Note on generalization, regularization, and architecture selection in nonlinear learning systems, in: First IEEE-SP Workshop on Neural Networks for Signal Processing, 1991.
[17] J.E. Moody, A smoothing regularizer for feedforward and recurrent neural networks, Neural Comput. 8 (1996) 461–489.
[18] N. Murata, S. Yoshizawa, S. Amari, Network information criterion—determining the number of hidden units for an artificial neural network model, IEEE Trans. Neural Networks 5 (6) (1994) 865–872.
[19] A.F. Murray, P.J. Edwards, Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training, IEEE Trans. Neural Networks 5 (5) (1994) 792–802.
[20] C. Neti, M.H. Schneider, E.D. Young, Maximally fault tolerance neural networks, IEEE Trans. Neural Networks 3 (1) (1992) 14–23.
[21] D.S. Phatak, I. Koren, Complete and partial fault tolerance of feedforward neural nets, IEEE Trans. Neural Networks 6 (1995) 446–456.
[22] D.S. Phatak, E. Tcherner, Synthesis of fault tolerance neural networks, in: Proceedings of the IJCNN02, 2002, pp. 1475–1480.
[23] C.H. Sequin, R.D. Clay, Fault tolerance in feedforward artificial neural networks, Neural Networks 4 (1991) 111–141.
[24] D. Simon, H. El-Sherief, Fault-tolerance training for optimal interpolative nets, IEEE Trans. Neural Networks 6 (1995) 1531–1535.
[25] J. Sum, K. Ho, On-line estimation of the final prediction error via recursive least square method, Neurocomputing 69 (2006) 2420–2424.
[26] E.B. Tchernev, R.G. Mulvaney, D.S. Phatak, Investigating the fault tolerance of neural networks, Neural Comput. 17 (2005) 1646–1664.

John Sum received the B.Eng. in Electronic Engineering from the Hong Kong Polytechnic University in 1992, M.Phil. and Ph.D. in CSE from the Chinese University of Hong Kong in 1995 and 1998. John spent 6 years teaching in several universities in Hong Kong, including the Hong Kong Baptist University, the Open University of Hong Kong and the Hong Kong Polytechnic University. In 2005, John moved to Taiwan and started to teach in Chung Shan Medical University. Currently, he is an Assistant Professor in the Institute of E-Commerce, the National Chung Hsing University, Taichung, ROC. His research interests include neural computation, mobile sensor networks and scale-free network. John Sum is a senior member of IEEE and an associate editor of the International Journal of Computers and Applications.

Chi-Sing Leung received the B.Sci. degree in electronics, the M.Phil. degree in Information Engineering, and the Ph.D. degree in Computer Science from the Chinese University of Hong Kong in 1989, 1991, and 1995, respectively. He is currently an Associate Professor in the Department of Electronic Engineering, City University of Hong Kong. His research interests include neural computing, data mining, and computer graphics. In 2005, he received the 2005 IEEE Transactions on Multimedia Prize Paper Award for his paper titled, "The Plenoptic Illumination Function" published in 2002. In 2007, he gave an one hour lecture, "Is there anything comparable to spherical harmonics but simpler?," in Game Developers Conference 2007 San Francisco. He is also a governing board member of the Asian Pacific Neural Network Assembly (APNNA).

Please cite this article as: J. Sum, A.C.-S. Leung, Prediction error of a fault tolerant neural network, Neurocomputing (2008), doi:10.1016/j.neucom.2008.05.009