On the Selection of Weight Decay Parameter for Faulty Networks

Chi Sing Leung, *Member, IEEE,* Hong-Jiang Wang, and John Sum, *Senior Member, IEEE*

***Abstract*—The weight-decay technique is an effective approach to handle overfitting and weight fault. For fault-free networks, without an appropriate value of decay parameter, the trained network is either overfitted or underfitted. However, many existing results on the selection of decay parameter focus on fault-free networks only. It is well known that the weight-decay method can also suppress the effect of weight fault. For the faulty case, using a test set to select the decay parameter is not practice because there are huge number of possible faulty networks for a trained network. This paper develops two mean prediction error (MPE) formulae for predicting the performance of faulty radial basis function (RBF) networks. Two fault models, multiplicative weight noise and open weight fault, are considered. Our MPE formulae involve the training error and trained weights only. Besides, in our method, we do not need to generate a huge number of faulty networks to measure the test error for the fault situation. The MPE formulae allow us to select appropriate values of decay parameter for faulty networks. Our experiments showed that, although there are small differences between the true test errors (from the test set) and the MPE values, the MPE formulae can accurately locate the appropriate value of the decay parameter for minimizing the true test error of faulty networks.**

***Index Terms*—Faulty network, generalization error, mean prediction error, regularization, weight decay.**

I. Introduction

The weight-decay method [13], [23], [27], [28], [31] is an effective approach to improve the generalization ability and fault tolerance of neural networks. In weight-decay, a weight-decay term is added into the objective function. With the weight-decay term, the magnitude of the trained weights are constrained to be small. Hence, the network output function is smooth and so the generalization ability is improved. Besides, when magnitude of the trained weights is small, the effect of weight faults can be suppressed.

Manuscript received March 14, 2009; revised April 21, 2010 and January 14, 2010. Date of publication June 28, 2010; date of current version August 6, 2010. This work was supported by a grant from City University of Hong Kong, under Project 7 002 588.

C. S. Leung is with the Department of Electronic Engineering, City University of Hong Kong, Kowloon 852, Hong Kong (e-mail: eeleungc@cityu.edu.hk).

H.-J. Wang was with the Department of Electronic Engineering, the City University of Hong Kong, Kowloon 852, Hong Kong, and is with South China University of Technology, Guangzhou 510641, China (e-mail: eehjwang@hotmail.coms).

J. Sum was with the Department of Electronic Engineering, the City University of Hong Kong, Kowloon 852, Hong Kong, and is with the Institute of Electronic Commerce, National Chung Hsing University, Taichung, Taiwan (e-mail: pfsum@yahoo.com.hk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2049580

From the theoretical study, the generalization error contains the bias and variance terms. Large weights usually cause an excessive variance term [18]. With an appropriate decay parameter, the weight magnitudes are limited and then the variance term can be controlled. However, when the weight-decay effect is too large, the trained network will have an excessive bias term. Therefore, the performance of weight-decay crucially depends on the selection of the decay parameter.

One approach to select an appropriate decay parameter is to use a test set. We train a number of networks with different decay parameters and then select the best trained network based on the test set. However, in many real situations, data are very scarce. Also, the process to investigate the performance of the trained networks based on test sets is very time-consuming. Another approach is final prediction error (FPE) [31]. We train a number of networks with different decay parameters and then select the best trained network based on a so-called prediction error formula that is a function of training error and trained weights. Although there are a lot of theoretical results [13], [18], [23], [31], [32], [35], [39], [43] related to the generalization ability, many of them focus on fault-free networks only.

For the fault tolerance, we are interested in, how well a trained network performs when node or weight failure appears. In the past, it was commonly assumed that neural networks have a built-in ability against node or weight failure. When some weights are deviated from their trained values, or some hidden nodes of a trained network are out of work, the degradation in the performance of the trained network should not be so large. In fact, many neural network pioneers [1], [6], [11], [34], [37], [38], [42] have investigated the property of network fault and proposed a number of methods to improve the fault tolerance.

In neural networks, if special methods are not taken during training, the fault situation could lead to a drastic performance degradation. Hence, obtaining a neural network with fault tolerance is of paramount importance. In the implementation of a neural network, network faults take place unavoidably, mainly occurring in two kinds of forms, multiplicative weight noise [6], [38], [42] and open weight fault [16], [26], [37], [47], [48].

Fig. 1 illustrates the effect of multiplicative weight noise on a trained radial basis function (RBF) network that has 50 nodes. The network is trained to learn the Hermite function [35]. Fig. 1(a) shows the network output of an RBF network trained by the least square method. Clearly, the trained

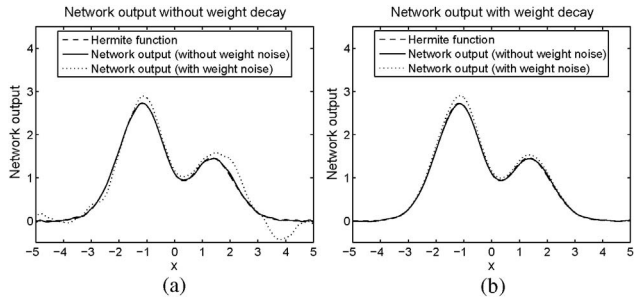

Fig. 1. Illustration example of the effect of multiplicative weight noise. (a) Least square method. (b) Weight decay method.

network fits well to the Hermite function when there is no weight noise. However, when the multiplicative weight noise (its variance equals 0.01) is presented, the output is greatly distorted. To solve the problem, we can use the weight-decay method. Fig. 1(b) shows the network output of an RBF networks trained by the weight-decay method. Clearly, the weight-decay method can greatly suppress the distortion.

Several algorithms for handling the two fault models have been developed [5]–[7], [14], [16], [26], [37], [40], [47], [48]. Some of these methods include injecting random node fault [40] or injecting weight noise [15], [34]. However, the training speed is quite slow. The replication technique [2], [21], [37], [44] in which hidden nodes are replicated from a trained network, is an effective method to improve the fault tolerance but this approach needs to use additional sources. In [12], the equicontinuous properties of sigmoidal feedforward networks were analyzed. The result show that the generally used arbitrary weight sigmoidal networks will lead to nonequicontinuous properties and then the fault tolerance of trained network becomes very poor. Besides, the fault-tolerance behavior of the networks is analyzed and error bounds for the induced errors are established based on the equicontinuous concept. However, there was no experimental result in [12] to support the analysis and the analysis focuses on the training error only.

Nevertheless, most of the mentioned results focused on the training error only. For instance, a regularizer approach [26] was proposed for handling open weight fault but the objective is to minimize the training set error over all possible faulty networks. Although they can be modified to handle the generalization error based on subset selection or test set, the modifications are not practice because there are huge number of possible faulty networks.

In particular, the weight-decay technique [5], [7], [10], [14] was proposed to handle these two fault models. Since the output of a trained network is very sensitive to large weights, the weight-decay technique can limit the weight magnitude and then improves the fault tolerance. To optimize the generalization ability under the fault situation, we need to train a number of trained networks with difference weight-decay parameters. Afterwards, for each trained network, we generate a huge number of faulty networks to study the generalization ability by feeding the test set to those faulty networks. Clearly, this test set-based method is computationally intensive.

To the best of our knowledge, the theoretical result of the generalization error on the weight-decay trained networks under the fault situation has not yet been explored. Hence, it will be useful to develop mean prediction error (MPE) formulae for the fault situation. Based on these formulae, we can perform the parameter selection for the weight-decay approach.

This paper develops two MPE formulae for RBF networks under the fault situation. One is used for handling multiplicative weight noise. Another one is used for handling open weight fault. Our experiments showed that, although there are small differences between the true test errors (from the test set and generated faulty networks) and MPE values, the two MPE formulae can accurately locate the appropriate value of decay parameter for minimizing the true test error of faulty networks. To improve the searching method on the decay parameter, we also discuss the way to estimate the gradients of MPE (with respect to the decay parameter).

In Section II, the background knowledge about weight-decay for RBF networks is presented. Then, the effect of fault situations on the trained RBF networks is analyzed in Section III. The derivation of the MPE error formulae is presented in Section IV. Section V discusses the way to estimate the gradients of MPE and presents the searching method on the decay parameter. In Section VI, we present the simulation results. Section VII concludes our results.

II. Background

The training dataset is denoted as: $\mathcal{D}_t = \{(\mathbf{x}_j, y_j) : \mathbf{x}_j \in \Re^K, y_j \in \Re, j = 1, \ldots, N.\}$, where $\mathbf{x}_j$ and y_j are the input and output samples of an unknown system, respectively. It is generated by a stochastic system [3], [13], given by

$$y_j = f(\mathbf{x}_j) + e_j \tag{1}$$

where $f(\cdot)$ is the unknown mapping, and e_j is the zero-mean Gaussian noise with variance equal to S_e.

We would like to approximate the mapping $f(\cdot)$ by an RBF network [36], [41], given by

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) \tag{2}$$

where $\mathbf{w} = [w_1, \ldots, w_M]^T$ is the weight vector, and $\phi_j(\cdot)$ is the jth basis function, given by $\phi_j(\mathbf{x}) = \exp(-\frac{\|\mathbf{x}-\mathbf{c}_j\|^2}{\Delta})$. Vectors $\mathbf{c}_j$'s are the RBF centers. The parameter Δ (> 0) controls the width of basis functions. In the vector-matrix form, (2) can be written as

$$\hat{f}(\mathbf{x}, \mathbf{w}) = \Phi^T(\mathbf{x})\, \mathbf{w} \tag{3}$$

where $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_M(\mathbf{x})]^T$. We use the RBF model to approximate the unknown system $f(\cdot)$ (an incomplete modeling approach [24]) because we do not know the exact mathematical model of the unknown system. This will introduce a small non-computable error term in the estimation of the test error. In general, the learning task is to find a weight vector that minimizes the following mean square error

$\mathcal{J}(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N}(y_j - \Phi^T(\mathbf{x}_j)\,\mathbf{w})^2$. On the other hand, the objective function of weight-decay is given by

$$\mathcal{J}(\mathbf{w}, \lambda) = \frac{1}{N}\sum_{j=1}^{N}(y_j - \Phi^T(\mathbf{x}_j)\,\mathbf{w})^2 + \lambda \mathbf{w}^T\mathbf{w} \tag{4}$$

where λ is the decay parameter. Given a fixed λ, the optimal weight vector $\mathbf{w}$ is given by

$$\mathbf{w} = \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1} \frac{1}{N}\sum_{j=1}^{N}\Phi(\mathbf{x}_j) y_j \tag{5}$$

where $\mathbf{I}$ is the identity matrix, and $\mathbf{H}_\phi = \frac{1}{N}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)\Phi^T(\mathbf{x}_j)$.

III. Faulty Network

Network faults can appear in many different forms, such as weight noise and weight fault. The multiplicative weight noise results from the finite precision representation of trained weights in the implementation [9]. For example, to implement a neural network in digital circuits, such as FPGA, the trained weights are usually obtained first by a high-precision computer. Then, the trained weights are encoded in digital implementation [22]. From [29], the rounding error is proportional to the magnitude of the weights. The loss of precision in the encoding process in the FPGA can be modeled as the multiplicative noise.

Distinguishing from the multiplicative weight noise, in open weight fault some RBF nodes are disconnected to the output layer. For example, in VLSI implementation, some physical faults, such as defects in silicon, open circuits in metals, and holes in oxides used in transistors [8], may appear. Those implementation defects cause the weights to be failed.

This section first introduces two fault models: multiplicative weight noise and open weight fault. Afterwards, we derive the mean training error (MTE) expressions for the two models.

A. Multiplicative Weight Noise

In multiplicative weight noise, each implemented weight deviates from its nominal value by a random percent, i.e., $\tilde{w}_{i,b} = w_i + b_i\, w_i \;\; \forall \;\; i = 1, 2, \ldots, M$. The matrix-vector form of the weight noise model is given by

$$\tilde{\mathbf{w}}_\mathbf{b} = \mathbf{w} + \mathbf{w} \otimes \mathbf{b} \tag{6}$$

where $\tilde{\mathbf{w}}_\mathbf{b} = [\tilde{w}_{1,b}, \ldots, \tilde{w}_{M,b}]^T$, $\mathbf{b} = [b_1, \ldots, b_M]$, $\otimes$ is the element-wise multiplication operator, and b_i's are identical independent mean zero random variables with variance S_b. The density function of b_i's are symmetric. In the matrix-vector form, the output of a faulty network is given by

$$\hat{f}(\mathbf{x}, \tilde{\mathbf{w}}_\mathbf{b}) = \Phi^T(\mathbf{x})\,\tilde{\mathbf{w}}_\mathbf{b}\,. \tag{7}$$

Then, the training error of an implementation $\tilde{\mathbf{w}}_\mathbf{b}$ is given by

$$\begin{aligned}\mathcal{E}(\mathcal{D}_t)_\mathbf{b} &= \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \Phi^T(\mathbf{x}_j)\,\tilde{\mathbf{w}}_\mathbf{b}\right)^2 \\ &= \frac{1}{N}\sum_{j=1}^{N}\left[\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j) w_i\right)^2 \right. \\ &\quad - 2\left(\sum_{i=1}^{M}\phi_i(\mathbf{x}_j) b_i w_i\right)\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j) w_i\right) \\ &\quad \left. + \sum_{i=1}^{M}\sum_{i'=1}^{M}\phi_i(\mathbf{x}_j)\phi_{i'}(\mathbf{x}_j) b_i b_{i'} w_i w_{i'}\right]. \end{aligned} \tag{8}$$

Since b_i's are zero mean *i.i.d.* random variables with symmetric density, the expectation on $\mathcal{E}(\mathcal{D})_\mathbf{b}$ is

$$\begin{aligned}\bar{\mathcal{E}}(\mathcal{D}_t)_\mathbf{b} &= \langle \mathcal{E}(\mathcal{D}_t)_\mathbf{b}\rangle_\mathbf{b} \\ &= \left\langle \frac{1}{N}\sum_{j=1}^{N}\left[\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j) w_i\right)^2 \right.\right. \\ &\quad \left.\left. + \sum_{i=1}^{M}\sum_{i'=1}^{M}\phi_i(\mathbf{x}_j)\phi_{i'}(\mathbf{x}_j) b_i b_{i'} w_i w_{i'}\right]\right\rangle_\mathbf{b} \\ &= \mathcal{E}(\mathcal{D}_t) + S_b \mathbf{w}^T\mathbf{G}\mathbf{w} \end{aligned} \tag{9}$$

where $\langle\cdot\rangle$ is the expectation operator, $\langle\cdot\rangle_\mathbf{b}$ means taking over all possible $\mathbf{b}$'s, and $\mathbf{G} = \mathbf{diag}(\mathbf{H}_\phi)$. In (9), $\mathcal{E}(\mathcal{D}_t) = \frac{1}{N}\sum_{j=1}^{N}(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)\,w_i)^2$ is the training error of a fault-free network, and the second term of the right side is the error created by weight noise.

B. Open Weight Fault

In open weight fault, the implementation of a weight is given by $\tilde{w}_{i,\beta} = \beta_i w_i\,, \quad \forall \quad i = 1, \ldots, M$, where the faulty factor β_i that describes whether the ith weight operates properly or not. The ith weight is out of work when $\beta_i = 0$; otherwise, the ith weight operates properly. In vector-matrix notation, the open weight fault model can be rewritten as

$$\tilde{\mathbf{w}}_\beta = \boldsymbol{\beta} \otimes \mathbf{w} \tag{10}$$

where $\tilde{\mathbf{w}}_\beta = [\tilde{w}_{1,\beta}, \ldots, \tilde{w}_{M,\beta}]^T$, and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_M]^T$. We assume that the faulty factors β_i's are identical independent binary random variables with $\text{Prob}(\beta_i = 0) = p$ and $\text{Prob}(\beta_i = 1) = 1 - p$. Hence, we have

$$\langle\beta_i\rangle = \langle\beta_i^2\rangle = 1 - p, \;\text{ and }\; \langle\beta_i\beta_{i'}\rangle = (1-p)^2 \;\forall\, i \neq i'. \tag{11}$$

Given a fault vector, the training error is given by

$$\begin{aligned}\mathcal{E}(\mathcal{D}_t)_\beta &= \frac{1}{N}\sum_{j=1}^{N}\left[y_j^2 - 2y_j\sum_{i=1}^{M}\beta_i w_i\phi_i(\mathbf{x}_j) \right. \\ &\quad \left. + \sum_{i=1}^{M}\sum_{i'=1}^{M}\beta_i\beta_{i'} w_i w_{i'}\phi_i(\mathbf{x}_j)\phi_{i'}(\mathbf{x}_j)\right]. \end{aligned} \tag{12}$$

From (11), the expectation on $\mathcal{E}(\mathcal{D}_t)_\beta$ over all possible fault patterns is given by

$$\bar{\mathcal{E}}(\mathcal{D}_t)_\beta = \langle \mathcal{E}(\mathcal{D}_t)_\beta \rangle_\beta = (1-p)\mathcal{E}(\mathcal{D}_t) + \frac{p}{N}\sum_{j=1}^{N} y_j^2 + (p^2-p)\mathbf{w}^T\left(\mathbf{H}_\phi - \mathbf{G}\right)\mathbf{w}. \quad (13)$$

Similar to (9), in (13) the first term on the right-hand side is related to the training error of a fault-free network while the second and third terms are related to the error created by the open weight fault.

IV. Mean Prediction Error for Faulty Networks

For the faulty case, we are interested in estimating the MPE of the faulty network from the training error of a fault-free network.

A. Multiplicative Weight Noise

Let $\mathcal{D}_t = \{(\mathbf{x}_j, y_j)\}_{j=1}^{N}$ and $\mathcal{D}_f = \{(\mathbf{x}'_j, y'_j)\}_{j=1}^{N'}$, be the training set and the test set, respectively. With weight noise, the MTE $\bar{\mathcal{E}}(\mathcal{D}_t)_\mathbf{b}$ and the MPE $\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b}$ are given by

$$\bar{\mathcal{E}}(\mathcal{D}_t)_\mathbf{b} = \left\langle \left(y - \Phi^T(\mathbf{x})\tilde{\mathbf{w}}_\mathbf{b}\right)^2 \right\rangle_{\mathcal{D}_t,\mathbf{b}} = \left\langle y^2 \right\rangle_{\mathcal{D}_t} - 2\left\langle y\Phi^T(\mathbf{x})\mathbf{w}\right\rangle_{\mathcal{D}_t} + \mathbf{w}^T\left(\mathbf{H}_\phi + S_b\mathbf{G}\right)\mathbf{w} \quad (14)$$

and

$$\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b} = \langle y'^2\rangle_{\mathcal{D}_f} - 2\langle y'\Phi^T(\mathbf{x}')\mathbf{w}\rangle_{\mathcal{D}_f} + \mathbf{w}^T(\mathbf{H}'_\phi + S_b\mathbf{G}')\mathbf{w} \quad (15)$$

respectively, where $\mathbf{H}'_\phi = \frac{1}{N'}\sum_{j=1}^{N'}\Phi(\mathbf{x}'_j)\Phi^T(\mathbf{x}'_j)$, $\mathbf{G}' = \mathbf{diag}\left\{\mathbf{H}'_\phi\right\}$.

Following the same technique as in [27] and [33], we assume that there is an optimal $\mathbf{w}_o$ such that

$$y_j = \Phi^T(\mathbf{x}_j)\mathbf{w}_o + e_j \quad \text{and} \quad y'_j = \Phi^T(\mathbf{x}'_j)\mathbf{w}_o + e'_j \quad (16)$$

where e_j's and e'_j's are independent zero-mean Gaussian random variables with variance equal to S_e. Note that $\mathbf{w}$ is obtained entirely by $\mathcal{D}_t$, which is independent of $\mathcal{D}_f$. Therefore, we have

$$\left\langle y'\phi^T(x')\mathbf{w}\right\rangle_{\mathcal{D}_f} = \left(\frac{1}{N'}\sum_{k=1}^{N'} y'_k\phi^T(x'_k)\right)\mathbf{w}. \quad (17)$$

The second term in $\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b}$ in (15) can thus be given by

$$-2\left(\frac{1}{N'}\sum_{j=1}^{N'} y'_j\Phi^T(\mathbf{x}'_j)\right)(\mathbf{H}_\phi + \lambda\mathbf{I})^{-1}\left(\frac{1}{N}\sum_{j=1}^{N} y_j\Phi(\mathbf{x}_j)\right). \quad (18)$$

Furthermore, since e'_j's and e_j's are independent, the terms in (18) can be simplified to

$$\frac{1}{N'}\sum_{j=1}^{N'} y'_j\Phi^T(\mathbf{x}'_j) = \frac{1}{N'}\sum_{j=1}^{N'}\left((\Phi^T(\mathbf{x}'_j)\mathbf{w}_o + e'_j)\cdot\Phi^T(\mathbf{x}'_j)\right) = \mathbf{w}_o^T\mathbf{H}'_\phi \quad (19)$$

$$\frac{1}{N}\sum_{j=1}^{N} y_j\Phi(\mathbf{x}_j) = \frac{1}{N}\sum_{j=1}^{N}\left((\Phi^T(\mathbf{x}_j)\mathbf{w}_o + e_j)\cdot\Phi(\mathbf{x}_j)\right) = \mathbf{H}_\phi\mathbf{w}_o. \quad (20)$$

From (19) and (20), the second term $-2\left\langle y'\Phi^T(\mathbf{x}')\mathbf{w}\right\rangle_{\mathcal{D}_f}$ in (15) becomes

$$2\mathbf{w}_o^T\mathbf{H}'_\phi\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o. \quad (21)$$

In a similar method, the second term $-2\left\langle y\Phi^T(\mathbf{x})\mathbf{w}\right\rangle_{\mathcal{D}_t}$ in (14) is given by (see Appendix A)

$$-2\left(\frac{S_e}{N}\mathrm{Tr}\{\mathbf{H}_\phi(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\} + \mathbf{w}_o^T\mathbf{H}_\phi(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\mathbf{H}_\phi\mathbf{w}_o\right) \quad (22)$$

where $\mathrm{Tr}\{\cdot\}$ denotes the trace operation, and S_e is the variance of the measured noise. As a result, the MPE of a faulty network can be in terms of the MTE of the faulty network, given by

$$\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b} = \bar{\mathcal{E}}(\mathcal{D}_t)_\mathbf{b} + \left\langle y'^2\right\rangle_{\mathcal{D}_f} - \left\langle y^2\right\rangle_{\mathcal{D}_t} + 2\frac{S_e}{N}\mathrm{Tr}\left\{\mathbf{H}_\phi(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\right\} + \mathbf{w}^T\left((\mathbf{H}'_\phi - \mathbf{H}_\phi) + S_b(\mathbf{G}'-\mathbf{G})\right)\mathbf{w} - 2\mathbf{w}_o^T(\mathbf{H}'_\phi - \mathbf{H}_\phi)\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o. \quad (23)$$

Since there is no way to know the exact $\mathbf{H}'_\phi$ and $\mathbf{G}'$, we assume that $\mathbf{H}'_\phi = \mathbf{H}_\phi$, and $\mathbf{G}' \approx \mathbf{G}$. This assumption is used for developing the MPE formulae for faulty networks only. With this assumption, our MPE results can still be used for selecting appropriate value of decay parameter to minimize the test error of faulty networks. Hence, the MPE can be expressed as

$$\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b} \approx \bar{\mathcal{E}}(\mathcal{D}_t)_\mathbf{b} + 2\frac{S_e}{N}\mathrm{Tr}\left\{\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\right\}. \quad (24)$$

From (9), the MPE of a faulty network with weight noise can be in terms of the MTE of the fault-free network, given by

$$\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b} = \mathcal{E}(\mathcal{D}_t) + 2\frac{S_e}{N}\mathrm{Tr}\left\{\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\right\} + S_b\mathbf{w}^T\mathbf{G}\mathbf{w} \quad (25)$$

where the term $\mathcal{E}(\mathcal{D}_t)$ is the training error of the trained fault-free network. Besides, $\mathbf{H}_\phi$ and $\mathbf{G}$ can be obtained from the training set while the weight noise S_b is assumed to be known. The only unknown variable is the variance of the measurement noise S_e but it can be estimated from the Fedorov's method [17] or Moody's method [32], given by

$$S_e \approx \frac{1}{N-M}\sum_{j=1}^{N}\left(y_j - \Phi^T(\mathbf{x}_j)\,\mathbf{H}_\phi^{-1}\frac{1}{N}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)y_j\right)^2. \quad (26)$$

From (25), we can directly estimate the generalization ability based on the training error of a fault-free network, the trained weights, and the training set.

B. Open Weight Fault

Following the similar method used in weight noise, the MPE $\bar{\mathcal{E}}(\mathcal{D}_f)_\beta$ of a faulty network with open weight fault can be also

in terms of the training error of a fault-free network. Firstly, the MTE $\bar{\mathcal{E}}(\mathcal{D}_t)_\beta$ and the MPE $\bar{\mathcal{E}}(\mathcal{D}_f)_\beta$ can be expressed as

$$\begin{aligned}\bar{\mathcal{E}}(\mathcal{D}_t)_\beta &= \left\langle \left(y - \Phi^T(\mathbf{x})\tilde{\mathbf{w}}_\beta\right)^2 \right\rangle_{\mathcal{D}_t,\beta} \\ &= \left\langle y^2 \right\rangle_{\mathcal{D}_t} - 2(1-p)\left\langle y\Phi^T(\mathbf{x})\mathbf{w}\right\rangle_{\mathcal{D}_t} \\ &\quad + (1-p)\mathbf{w}^T\left((1-p)\mathbf{H}_\phi + p\mathbf{G}\right)\mathbf{w} \qquad (27) \\ \bar{\mathcal{E}}(\mathcal{D}_f)_\beta &= \left\langle y'^2 \right\rangle_{\mathcal{D}_f} - 2(1-p)\left\langle y'\Phi^T(\mathbf{x}')\mathbf{w}\right\rangle_{\mathcal{D}_f} \\ &\quad + (1-p)\mathbf{w}^T\left((1-p)\mathbf{H}'_\phi + p\mathbf{G}'\right)\mathbf{w}. \qquad (28)\end{aligned}$$

According to the similar derivations of (16) and (20), the second term $-2(1-p)\left\langle y'\Phi^T(\mathbf{x}')\mathbf{w}\right\rangle_{\mathcal{D}_f}$ in (28) is

$$-2(1-p)\mathbf{w}_o^T\mathbf{H}'_\phi\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o \qquad (29)$$

and the second term $-2(1-p)\left\langle y\Phi^T(\mathbf{x})\mathbf{w}\right\rangle_{\mathcal{D}_t}$ in (27) is

$$\begin{aligned}&-2(1-p)\left(\mathbf{w}_o^T\mathbf{H}_\phi\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o\right. \\ &\left.+\frac{S_e}{N}\mathrm{Tr}\left\{\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\right\}\right). \qquad (30)\end{aligned}$$

Therefore, the MPE can be estimated by

$$\bar{\mathcal{E}}(\mathcal{D}_f)_\beta \approx \bar{\mathcal{E}}(\mathcal{D}_t)_\beta + 2\frac{S_e}{N}\mathrm{Tr}\left\{(1-p)\mathbf{H}_\phi(\mathbf{H}_\phi + \lambda\mathbf{I})^{-1}\right\}. \qquad (31)$$

From (13), we can obtain

$$\begin{aligned}\bar{\mathcal{E}}(\mathcal{D}_f)_\beta &= (1-p)\,\mathcal{E}(\mathcal{D}_t) + \frac{1}{N}\sum_{j=1}^{N} p\,y_j^2 \\ &\quad + 2\frac{S_e}{N}\mathrm{Tr}\left\{(1-p)\,\mathbf{H}_\phi\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\right\} \\ &\quad + (p^2 - p)\,\mathbf{w}^T\left(\mathbf{H}_\phi - \mathbf{G}\right)\mathbf{w}. \qquad (32)\end{aligned}$$

From (32), we can directly estimate the generalization ability based on the training error of fault-free network, the trained weights, and the training set.

C. Complexity of MPE Formulae

When we use the test set method (test set and generating a number of faulty networks) to measure the generalization error, the complexity is $O(M \times N' \times L)$ where M is the number of RBF nodes, N' is the size of the test set, L is the number of faulty networks, and $L >> M$ and $N' > M$. Note that in some situations, data are very scarce and we may not have a test set.

In our MPE formulae, the complexity of the direct calculation is $O(M \times N + M^3)$, where N is the size of the training set, and $N > M$. Since the matrix $\mathbf{H}_\phi$ is fixed for all λ and it can be per-diagonalized, the term $\mathrm{Tr}\{\mathbf{H}_\phi\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\}$ can be computed in an efficient way, given by $\frac{S_e}{N}\sum_i^M \frac{d_i}{d_i+\lambda}$, where d_i is the eigenvalue of $\mathbf{H}_\phi$. Hence, the complexity is equal to $O(M \times N + M^2)$ only.[1] Since the number M of nodes is usually less than the number N of training samples, the complexity is equal to $O(M \times N)$.

[1]Although the computational cost of $\mathrm{Tr}\left\{\mathbf{H}_\phi\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\right\}$ is reduced to $O(M)$, the matrix $\mathbf{G}$, in (29) and (36), becomes non-diagonal in the eigen domain of $\mathbf{H}_\phi$. Hence, the complexity is equal to $O(M \times N + M^2)$.

V. Selection of Weight Decay Parameter

The two MPE formulae, (25) and (32), can help us to perform the model selection of faulty RBF networks, without using the test set and generating faulty networks. For a fixed architecture,[2] we should minimize the generalization error with respect to λ. Since the MPE values, $\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b}$ and $\bar{\mathcal{E}}(\mathcal{D}_f)_\beta$, must be evaluated with the weight vector and the weight vector is also a function of λ, there is no a simple exact close form expression for the optimal value of λ.

In general, for the classical fault-free case, there are two approaches [19], [20], [39], [43] to search an optimal value of λ. The first approach [39], [43] is to try several values of λ. For each value of λ, we construct an RBF network based on the training set and then use a formula to predict the generalization error. Since a large range of λ produces the similar MPE value, taking logarithm on λ can help us to reduce the computational load. In the second approach [19], [20], we estimate the gradient of the prediction error with respect to λ. Based on the approximated gradient, we can search the appropriate value of λ.

In this paper, we follow the second approach. We will discuss a systemic way to estimate the gradient for faulty cases. We will derive two expressions to estimate the gradients of MPEs for faulty networks. One is of the multiplicative weight noise case. Another is of the open node fault case. One should notice that the result in [19] and [20] is for handling the fault-free case only. Here, we investigate the MPEs for faulty networks. Besides, the mathematical derivation of the gradient of MPEs in our apporach is different from the derivation in [19] and [20].

A. Multiplicative Weight Noise: Gradient of MPE With Respect to λ

The test error under multiplicative weight noise is given by

$$\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b} = J_1 + J_2 + J_3 + J_4 \qquad (33)$$

where $J_1 = \left\langle y'^2\right\rangle_{\mathcal{D}_f}$, $J_2 = -2\left\langle y'\Phi^T(\mathbf{x}')\mathbf{w}\right\rangle_{\mathcal{D}_f}$, $J_3 = \mathbf{w}^T\mathbf{H}'_\phi\mathbf{w}$, and $J_4 = \mathbf{w}^T\mathbf{G}'_\phi\mathbf{w}$. Our task is to evaluate $\frac{\partial\bar{\mathcal{E}}(\mathcal{D}_f)_\mathbf{b}}{\partial\lambda}$. Since J_1 is not a function of λ or $\mathbf{w}$, $\frac{\partial J_1}{\partial\lambda} = 0$.

From (21)

$$J_2 = -2\mathbf{w}_o^T\,\mathbf{H}'_\phi\,\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\,\mathbf{H}_\phi\,\mathbf{w}_o\,. \qquad (34)$$

Similar to Section IV, we assume that $\mathbf{H}'_\phi \approx \mathbf{H}_\phi$ and $\mathbf{G}'_\phi \approx \mathbf{G}_\phi$. Now, J_2 becomes

$$J_2 = -2\mathbf{w}_o^T\,\mathbf{H}_\phi\,\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\,\mathbf{H}_\phi\,\mathbf{w}_o\,. \qquad (35)$$

Although J_2 now is a function of λ, in the current form of J_2 the gradient with respect to λ is not analytic computable. Since $\mathbf{H}_\phi$ is positive or semi-positive definite, we can consider the SVD decomposition on $\mathbf{H}_\phi$ [24]

$$\mathbf{H}_\phi = \mathbf{P}\mathbf{D}\mathbf{P}^T \qquad (36)$$

[2]The number of nodes and the RBF width parameter Δ are fixed.

where $\mathbf{P}$ is an orthonormal matrix, $\mathbf{P}\mathbf{P}^T = \mathbf{I}$, and $\mathbf{D}$ is a diagonal matrix whose elements $\{d_1, \ldots, d_M\}$ are the eigenvalues of $\mathbf{H}_\phi$. Define

$$\check{\mathbf{w}}_o = \mathbf{P}^T \mathbf{w}_o \,. \tag{37}$$

From (36) and (37), $J_2 = -2\sum_{i=1}^{M} \frac{\check{w}_{oi}^2 d_i^2}{d_i+\lambda}$. So, we have

$$\frac{\partial J_2}{\partial \lambda} = 2\sum_{i=1}^{M} \frac{\check{w}_{oi}^2 d_i^2}{(d_i+\lambda)^2} \,. \tag{38}$$

Similarly, we have

$$\begin{aligned} J_3 \;&=\; \mathbf{w}_o^T \mathbf{H}_\phi \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1} \mathbf{H}_\phi \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1} \mathbf{H}_\phi \mathbf{w}_o \\ &\quad + \frac{S_e}{N} \operatorname{Tr}\left(\mathbf{H}_\phi \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1} \mathbf{H}_\phi \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1}\right). \end{aligned} \tag{39}$$

From (36) and (37), $J_3 = \sum_{i=1}^{M} \left(\frac{\check{w}_{oi}^2 d_i^3}{(d_i+\lambda)^2} + \frac{S_e}{N} \frac{d_i^2}{(d_i+\lambda)^2} \right)$. The gradient of J_3 is given by

$$\frac{\partial J_3}{\partial \lambda} = -2\sum_{i=1}^{M} \left(\frac{\check{w}_{oi}^2 d_i^3}{(d_i+\lambda)^3} + \frac{S_e}{N} \frac{d_i^2}{(d_i+\lambda)^3} \right). \tag{40}$$

Similarly, for J_4, we have

$$\begin{aligned} J_4 \;&=\; \mathbf{w}_o^T \mathbf{H}_\phi \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1} \mathbf{G} \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1} \mathbf{H}_\phi \mathbf{w}_o \\ &\quad + \frac{S_e}{N} \operatorname{Tr}\left(\mathbf{H}_\phi \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1} \mathbf{G} \left(\mathbf{H}_\phi + \lambda \mathbf{I}\right)^{-1}\right). \end{aligned} \tag{41}$$

In (41), we can diagonalize $\mathbf{H}_\phi$ but we will introduce a non-diagonal matrix $\mathbf{P}^T\mathbf{G}\mathbf{P}$. Hence, we need an approximation on $\mathbf{G}$. Recall that $\mathbf{G} = \mathbf{diag}(\mathbf{H}_\phi)$ and $g_{ii} = \langle \phi_i^2(x) \rangle_{\mathcal{D}_f}$. If the RBF centers are distributed according to the input patterns, we can approximate $\mathbf{G}$ by $\mathbf{G} \approx \bar{g}\mathbf{I}$, where $\bar{g} = \frac{1}{M}\sum_{i=1}^{M} \langle \phi_i^2(x) \rangle_{\mathcal{D}_f}$. Although this approximation may not always strictly hold, our simulation results (in the next section) show that with this approximation we still can locate a good value of λ to minimize the generalization error of faulty networks. With the approximation, the gradient of J_4 is

$$\frac{\partial J_4}{\partial \lambda} = -2\sum_{i=1}^{M} \left(\frac{\check{w}_{oi}^2 d_i^2 \bar{g}}{(d_i+\lambda)^3} + \frac{S_e}{N} \frac{d_i \bar{g}}{(d_i+\lambda)^3} \right). \tag{42}$$

To sum up, the gradient is given by

$$\begin{aligned} \frac{\partial \bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}}}{\partial \lambda} \;&=\; 2\sum_{i=1}^{M} \left(\frac{\check{w}_{oi}^2 d_i^2}{(d_i+\lambda)^2} - \frac{\check{w}_{oi}^2 d_i^3 + \check{w}_{oi}^2 d_i^2 \bar{g}}{(d_i+\lambda)^3} \right. \\ &\quad \left. - \frac{S_e}{N} \frac{d_i^2 + d_i \bar{g}}{(d_i+\lambda)^3} \right). \end{aligned} \tag{43}$$

B. Open Weight Fault: Gradient of MPE With Respect to λ

Following the method used in the weight noise case, the gradient of MPE open fault is given by

$$\begin{aligned} \frac{\partial \bar{\mathcal{E}}(\mathcal{D}_f)_\beta}{\partial \lambda} \;&=\; 2(1-p)\sum_{i=1}^{M} \left(\frac{\check{w}_{oi}^2 d_i^2}{(d_i+\lambda)^2} \right. \\ &\quad - \frac{(1-p)\check{w}_{oi}^2 d_i^3 + p\check{w}_{oi}^2 d_i^2 \bar{g}}{(d_i+\lambda)^3} \\ &\quad \left. - \frac{S_e}{N} \frac{(1-p) d_i^2 + p d_i \bar{g}}{(d_i+\lambda)^3} \right). \end{aligned} \tag{44}$$

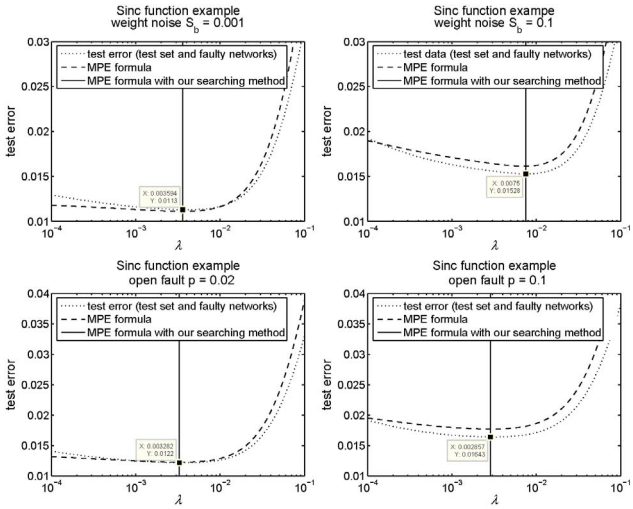

Fig. 2. MPE value and the true test error for the sinc function example. The vertical solid line indicates the optimized λ based on MPE formula with our searching method.

C. Searching λ

We can develop a gradient-based searching method to minimize the MPE value based on (43) and (44). We do not know $\mathbf{w}_o$, as well as $\check{\mathbf{w}}_o$, in advance. Hence, during the estimation, we put the current estimate of weight vector $\mathbf{w}$, as well as $\check{\mathbf{w}} = \mathbf{P}^T \mathbf{w}$, in (43) and (44). Our searching method is based on the famous bold-driver technique [4], [45].[3] The updating λ is given by

$$\lambda^{new} = \lambda - \eta \frac{\partial \mathrm{MPE}}{\partial \lambda}. \tag{45}$$

where η is an adjustable learning rate. The parameter η is varied according to whether or not an update on λ improves the MPE value. There are two rules for adjusting the parameter, given as follows.

1) If an update on λ, based on (45), produces a network with its MPE value, based on (25) or (32), above the previous MPE value, the change to the λ is rejected and η is multiplied by a factor ε less than one.
2) If an update on λ decreases the MPE value, the changes to λ is accepted and η is multiplied with a factor γ greater than 1.

In our experiments, the initial value of λ is set to 0.1, the initial value of η is set to 0.01, and $\varepsilon = 1.04$ and $\gamma = 0.7$.

D. Complexity

We can reduce the searching complexity by per-diagonalizing $\mathbf{H}_\phi$. Afterwards, we perform the searching of λ in the eigen domain of $\mathbf{H}_\phi$. It is well known that there are many $O(M^3)$ methods to diagonalize $\mathbf{H}_\phi$. The computational cost to prepare other pre-computations, such as transforming $\mathbf{G}$ and $\sum_{j=1}^{N} \Phi(\mathbf{x}_j) y_j$ into the eigen domain, is $O(M^2 + M \times N)$.

With the above pre-computation, the complexity within a searching iteration can be greatly reduced. Given a new λ, the

[3]This bold-driver method is originally used for training multilayered perceptrons and is used in MathLab Toolbox. In fact, it can adopted to search parameters in many optimization problems.

TABLE I
EFFICIENCY OF USING THE MPE FOR THE SINC FUNCTION EXAMPLE

	Our MPE Result		Test Set Method	
	Searched λ (MPE Value)	True Test Error	Optimal λ Range	True Test Error
$S_b = 0.001$ weight noise	0.003593 (0.01110)	0.01130	0.003548 to 0.004467	0.01130
$S_b = 0.01$ weight noise	0.004037 (0.01158)	0.01168	0.003981 to 0.004467	0.01167
$S_b = 0.1$ weight noise	0.007500 (0.01613)	0.01528	0.007079 to 0.007943	0.01528
$p = 0.02$ open fault	0.003282 (0.01228)	0.01224	0.003162 to 0.004467	0.01224
$p = 0.1$ open fault	0.002857 (0.01773)	0.01648	0.002818 to 0.003162	0.01648

Our method can accurately locate the appropriate value of λ for minimizing the true test error of faulty networks.

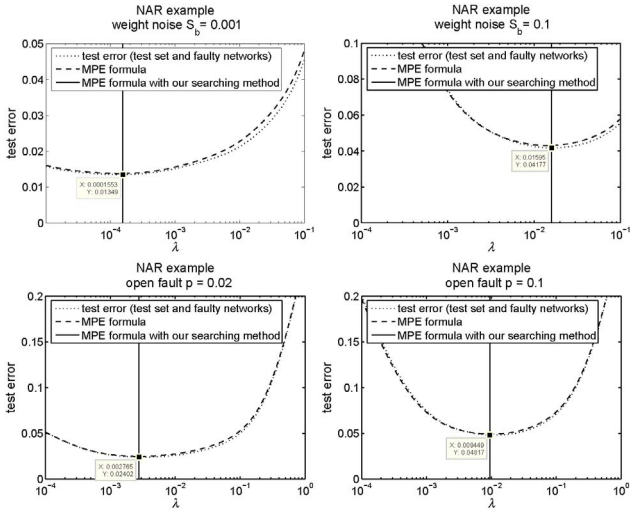

Fig. 3. MPE value and the true test error for the NAR time series prediction. The vertical solid line indicates the optimized λ based on MPE formula with our searching method.

complexity to get a new weight vector in the eigen domain is $O(M)$. Based on this eigen domain weight vector, the complexity to get the gradient is $O(M)$. The complexity to get the MPE value is $O(N \times M + M^2)$. Since the number N of training samples should be greater than the number M of RBF nodes, the overall complexity is $O(N \times M)$.

VI. SIMULATIONS

In this section, our theoretical results are experimentally verified by some examples: the sinc function [13], nonlinear autoregressive time series (NAR) [13], Mackey Glass chaotic time series (MGTS) [30], and Abalone life prediction [43].

A. Selection on Weight Decay Parameter

This section demonstrates how the MPE formulae, (25) and (32), and the method in Section V can help us to select an appropriate value of λ. The value of the decay parameter is selected based on our searching method presented in Section V.

As a comparison, for each setting, we train a number of networks under different λ values. The weight-decay parameter changes from 0.00001 to 1 in the logarithm scale. For each trained RBF network, we randomly generate 10 000 faulty networks for each fault level. Afterwards, we measure the true test error based on test set and generated faulty networks. Notice that the scope of this section is to demonstrate the effectiveness of our MPE formulae and searching method. We will discuss further applications of our MPE results for selecting other settings in RBF networks in Sections VI-B–VI-D.

TABLE II
EFFICIENCY OF USING THE MPE FOR THE NAR TIME SERIES PREDICTION

	Our MPE Result		Test Set Method	
	Searched λ (MPE Value)	True Test Error	Optimal λ Range	True Test Error
$S_b = 0.001$ weight noise	0.0001553 (0.01382)	0.01349	0.0001259 to 0.0001585	0.01349
$S_b = 0.01$ weight noise	0.001733 (0.02065)	0.02018	0.001585 to 0.001778	0.02018
$S_b = 0.1$ weight noise	0.01595 (0.04314)	0.04177	0.01585 to 0.01778	0.04177
$p = 0.02$ open fault	0.002765 (0.02483)	0.02402	0.003162 to 0.003348	0.02388
$p = 0.1$ open fault	0.09449 (0.04909)	0.04817	0.01122 to 0.01259	0.04805

MPE can accurately locate the appropriate value of λ for minimizing the true test error of faulty networks.

For the weight noise case, we try three values, $\{S_b = 0.001, 0.01, 0.1\}$. The case of $S_b = 0.001$ represents the nearly-fault-free situation. The other two values represent a middle fault level and a high-fault level, respectively. For the open weight fault case, we try two values, $\{p = 0.02, 0.1\}$. These two values represent a middle fault level and a high-fault level, respectively. For other fault levels, we also obtain consistent results.

1) *Regression:* The sinc function is a common benchmark example. The output is generated by $y = \mathrm{sinc}(x) + e$, where the noise term e is a mean zero Gaussian noise with variance $\sigma_e^2 = 0.01$. A training dataset (100 samples) is generated. The input data x are uniformly taken in from -5 to 5. Besides, a test set, containing 1000 samples, is generated. In our experiment, the network model has 37 nodes. There are 37 centers selected as $\{-5, -4.75, \ldots, 4.75, 5\}$ because the input x is uniformly taken in from -5 to 5. In addition, the parameter Δ is set to 0.1. The above setting on the number of RBF nodes and the parameter Δ produces satisfactory performance.[4]

The simulation results are presented in Fig. 2 and Table I. Our searching results are indicated by vertical lines in the figure. From the figure and table, although there are small differences between the true test error and MPE value, the two curves are quite similar in shape. Our MPE results can help us to select an appropriate value of λ for minimizing the true test error of faulty networks.

For example, for the weight noise case with $S_b = 0.1$, based on the test set method, the optimal value of λ is around $0.007079 - 0.007943$. In this range of λ, the true test error

[4]Of course, we can further optimize the setting based on our theoretical MPE results of faulty networks. We will use some examples to illustrate this further optimization in Sections VI-B and VI-C.

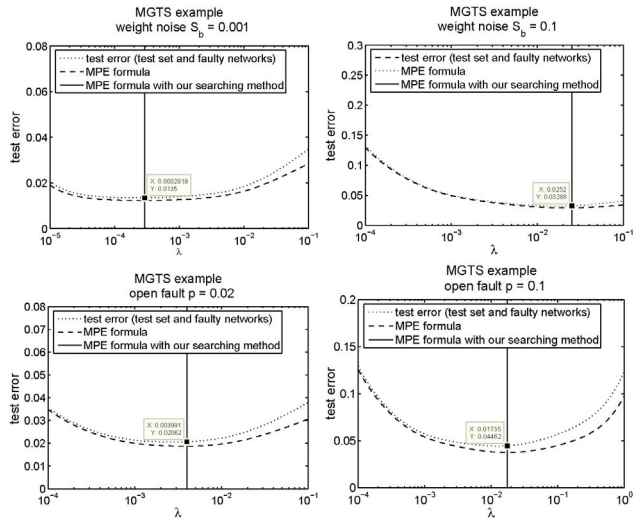

Fig. 4. MPE value and the true test error for the chaotic time series prediction. The vertical solid line indicates the optimized λ based on MPE formula with our searching method.

is around 0.01528. With our MPE formula and searching method on λ, the searched λ is around 0.007500, and the corresponding true test error is also around 0.01528. For the node fault cases, we also obtain the consistent result.

2) *Nonlinear Autoregression Time Series Prediction:* This part considers the following NAR time series, given by

$$\begin{aligned} y(i) &= \left(0.8 - 0.5\exp(-y^2(i-1))\right) y(i-1) \\ &\quad - \left(0.3 + 0.9\exp(-y^2(i-1))\right) y(i-2) \\ &\quad + 0.1\sin(\pi y(i-1)) + e(i) \end{aligned} \tag{46}$$

where $e(i)$ is a mean zero Gaussian random variable that drives the series. Its variance is equal to 0.01. One thousand samples were generated given $y(0) = y(-1) = 0$. The first 500 data points were used for training and the other 500 samples were used for testing. Our RBF model is used to predict $y(t)$ based on the past observations, $y(t-1)$ and $y(t-2)$. The prediction is given by

$$\hat{y}(t) = \hat{f}(\mathbf{x}(t), \mathbf{w}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}(t)) \tag{47}$$

where $\mathbf{x}(t) = [y(t-1), y(t-2)]^T$. For this 2-D input case, the Chen's LROLS method [13] is applied to select important RBF centers (basis functions) from the training samples. The number of selected RBF nodes is 21. In addition, the parameter Δ, i.e., the width of RBFs, is set to 0.81. This setting on the number of RBF nodes and the parameter Δ produces satisfactory performance.

The results from our MPE method and the true test error (from the test set and faulty networks) are depicted in Fig. 3 and Table II. Similar to the previous example, the figure and table show that the MPE formulae and our searching method can locate the an appropriate value of λ for minimizing the true test error of faulty networks. Although there some differences between the two ranges of λ, the difference between the corresponding two true test errors is very small. In this case, the degradation of using MPE to achieve an optimal true test error is very small.

TABLE III
EFFICIENCY OF USING THE MPE FOR THE CHAOTIC TIME SERIES PREDICTION

	Our MPE Result		Test Set Method	
	Searched λ (MPE Value)	True Test Error	Optimal λ Range	True Test Error
$S_b = 0.001$ weight noise	0.0002919 (0.01229)	0.01350	0.0002239 to 0.0002512	0.01349
$S_b = 0.01$ weight noise	0.002013 (0.01578)	0.01719	0.001585 to 0.001778	0.01716
$S_b = 0.1$ weight noise	0.02520 (0.02946)	0.03288	0.01585 to 0.01778	0.03242
$p = 0.02$ open fault	0.003991 (0.01870)	0.02061	0.002512 to 0.002818	0.02049
$p = 0.1$ open fault	0.01735 (0.03750)	0.04461	0.01122 to 0.01259	0.04429

Our can accurately locate the appropriate value of λ for minimizing the true test error of faulty networks

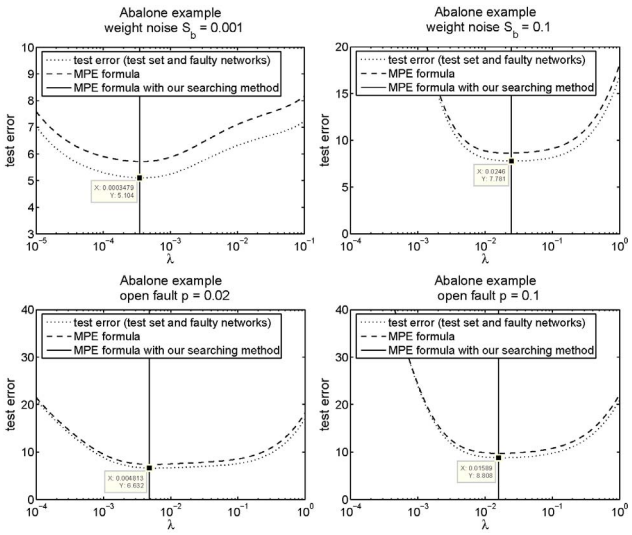

Fig. 5. MPE value and the true test error for the Abalone dataset. The vertical solid line indicates the optimized λ based on MPE formula with our searching method.

For the node fault case, with $p = 0.1$, based on the test set method, the optimal value of λ is around $0.01122-0.01259$. In this range of λ the true test error is around 0.04805. With our MPE result, the optimal value of λ is around 0.009449, and the corresponding true test error is 0.04817. The degradation of using MPE to achieve an optimal true test error is less than 0.24%.

3) *Chaotic Time Series Prediction:* The general form of Mackey Glass chaotic equation is expressed as

$$y(t) = \frac{ay(t-\tau)}{b + y^h(t-\tau)} - cy(t) + e(t) \tag{48}$$

where the parameters, a, b, c, h, and τ, are set to $a = 0.2$, $b = 1$, $c = 0.9$, and $h = 10$, respectively. The parameter $e(i)$ is a mean zero Gaussian random variable that drives the series. Its variance is 0.01. One thousand samples were generated. The first 500 data points, were used for training and the other 500 samples were used for testing.

TABLE IV
EFFICIENCY OF USING THE MPE FOR THE ABALONE DATA SET

	Our MPE Result		Test Set Method	
	Searched λ (MPE Value)	True Test Error	Optimal λ Range	True Test Error
$S_b = 0.001$ weight noise	0.0003479 (5.7140)	5.1044	0.0003548 to 0.0003981	5.1033
$S_b = 0.01$ weight noise	0.002524 (6.9674)	6.2914	0.003548 to 0.003981	6.2583
$S_b = 0.1$ weight noise	0.02460 (8.640)	7.7808	0.02511 to 0.02818	7.7805
$p = 0.02$ open fault	0.004813 (7.4144)	6.6324	0.005623 to 0.006310	6.6180
$p = 0.1$ open fault	0.01589 (8.8081)	8.8081	0.01778 to 0.01995	8.8022

MPE can accurately locate the appropriate value of λ for minimizing the true test error of faulty networks.

Our RBF model is used to predict $y(t)$ based on the past observations, $\{y(t-4), y(t-3), y(t-2), y(t-1)\}$. For this 4-D input case, the Chen's LROLS method is also applied to select important RBF centers (basis functions). The number of selected RBF nodes is 20. In addition, The parameter Δ is set to 0.81.

The results from our MPE method and the true test error (from the test set and faulty networks) are depicted in Fig. 4 and Table III. Similar to the previous examples, the figure and table show that the MPE formulae can locate the an appropriate value of λ for minimizing the true test error of faulty networks.

We take a close look on how the decay parameter affects the test error for a nearly fault-free case. For the weight noise case with $S_b = 0.001$, this situation is similar to the fault-free case. From our MPE results, we should use a very small λ, such as $\lambda = 0.00029$ and the test error is around 0.0135. Now, if we do not use any regularization, the test error is greater than 0.02.

Besides, we also take a close look on how the fault level affects the optimized value of decay parameter. For the weight noise case with $S_b = 0.001$, the optimal value of λ is around 0.00029. When we increase the fault level to $S_b = 0.1$, the optimal value of λ should be around 0.02. From the figure, if we choose an incorrect value of λ, the degradation on the performance is very large. For the weight noise case with $S_b = 0.1$, based on our method, when λ is correctly set, the test set error is around 0.033. Now, if we use a small value of λ (saying less than 0.0001), the test error will become very large (around 0.125). This example demonstrates that the fault situation could lead to a very poor performance when the weight-decay is not correctly used.

4) *Abalone Dataset:* To further verify our theoretic results, a multidimensional real dataset called the Abalone dataset is employed. The dataset is used to predict the age of Abalone, which includes nine physical measurements, i.e., sex, length, diameter, height, whole weight, shucked weight, viscera weight, shell weight, and rings. The age of Abalone can be determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope. It can

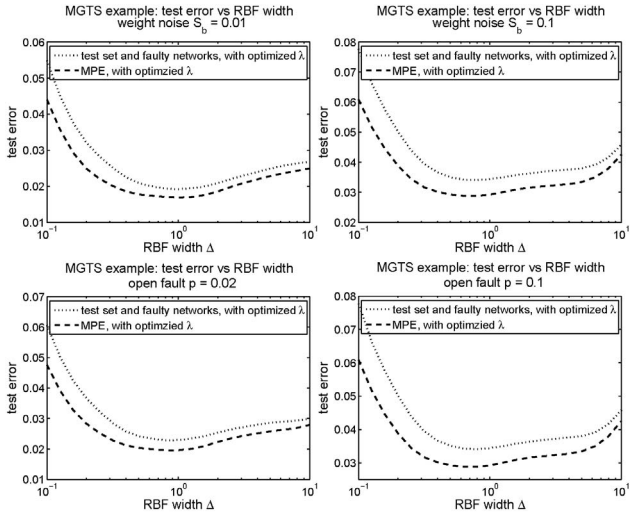

Fig. 6. Test error with optimized λ vs. RBF width Δ. We randomly choose 50 data point from the training set as the RBF node centers. Twenty difference values of λ from 0.1 to 10 in the logarithm scale are considered.

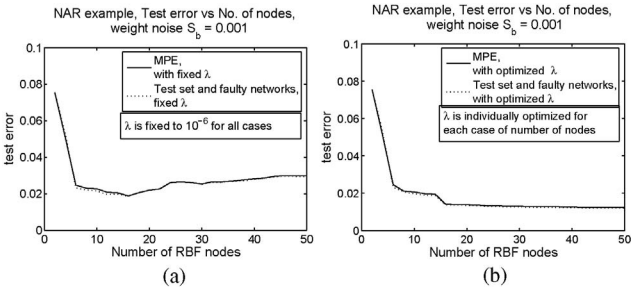

Fig. 7. Test error vs. number of nodes. Comparison between a fixed small λ and optimized λ. (a) Fixed very small λ. (b) Optimized λ.

be seen that the whole process is boring and time-consuming. The number of rings is the value to be predicted as either a continuous value or a classification problem. Abalone dataset has 4177 samples with nine variables. In our simulation, the second to eighth variables are taken as the input vector $\mathbf{x}$ and the ninth variable is taken as the output value y. The formal two thousand samples are used for training, and the remaining samples are used for measuring the predict error. In addition, for the selection of important RBFs, the Chen's LROLS method is employed to select the most important 20 important RBFs. Notice that the dynamic range of the output in this example is from 1 to 29. Hence, the magnitude of the test error (square error) is greater than that of the previous three examples.

The results from MPE (obtained from training error of a fault-free network) and the true test error (from the test set) are depicted in Fig. 5 and Table IV. Similar to the previous examples, the figure and table show that the MPE formulae can locate an appropriate value of λ for minimizing the true test error of faulty networks.

5) *Remark: Difference Between MPE Value and True Test Error:* From Figs. 2–5, although there are small differences between the true test errors and MPE values, the two curves, obtained from the true test errors and MPE vales, are quite similar in shape.

In the classical results on generalization errors [1], [31], [32] for the fault-free case, there are also some differences between the true test error and the estimated test error. In our MPE results for the fault case, the differences come from the approximation and assumptions in the analysis. In the practical situation, there is no way to know the test set $\mathbf{H}'$ and $\mathbf{G}'$. Hence, we use the training set $\mathbf{H}$ and $\mathbf{G}$ instead.

In the neural network community, we assume that the data are generated by $y_j = f(\mathbf{x}_j) + e_j$, where $f(\cdot)$ is the unknown system mapping, and e_j is the zero-mean Gaussian measurement noise with variance equal to S_e. In practical situations, we cannot know the variance S_e and we need to use some methods for its estimation.

The neural network approach is an incomplete modeling approach because we do not known the exact mathematical model of the unknown system. In the sinc, NAR, and MGST examples, the training and test data are not generated based on the RBF model. Also, in the Abalone example, there is no exact mathematical model to describe it. We use an RBF model to approximate the unknown system and use the model to analyze the generalization error. Of course, this will introduce a small non-computable error term in the analysis of the generalization error.

B. Selection on the RBF Width

The MPE formulae allows us not only to predict the performance of a trained network but also to select the model from various settings. In RBF networks, one turning parameter is the RBF width Δ. In this section, we illustrate how our MPE results can help us to select an appropriate value of Δ. Following the classical approaches in the selection of parameters for fault–free networks [31], [32], [39], [43], [46], we try different values of Δ. Afterwards, we use the MPE formulae to estimate the test error of faulty networks.

To illustrate the idea, the MGTS example mentioned in the previous section is used. We randomly choose 50 data points from the training set as the RBF node centers and 20 difference values of σ from 0.1 to 10 in the logarithm scale. For each value of Δ, we use our searching method to optimize the decay parameter λ, and then use the MPE formulae to estimate the test error of faulty networks. The results are depicted in Fig. 6. Since the decay parameters are optimized, for different values of Δ the variation on the test errors is not very large. Also, over a wide range of Δ, the test errors of faulty networks are quite similar. Although there are small differences between the true test errors and MPE values, both methods can locate the similar range of optimal Δ. Thus, the simulation confirms the applicability of our MPE results for the selection of RBF width. For other faulty levels and examples, we obtained similar results (not shown here).

C. Selection on the Number of RBF Nodes

In the RBF approach, another turning parameter is the number of nodes used. In this section, we illustrate how our MPE method can help us to select an appropriate number of nodes. Once again, following the classical approaches in the selection of parameters for fault-free networks [31], [32], [39], [43], we train a number of RBF networks with different number of nodes. To illustrate the idea, the NAR and Abalone examples mentioned in the previous section are used. Other settings are similar to the Section VI-A.

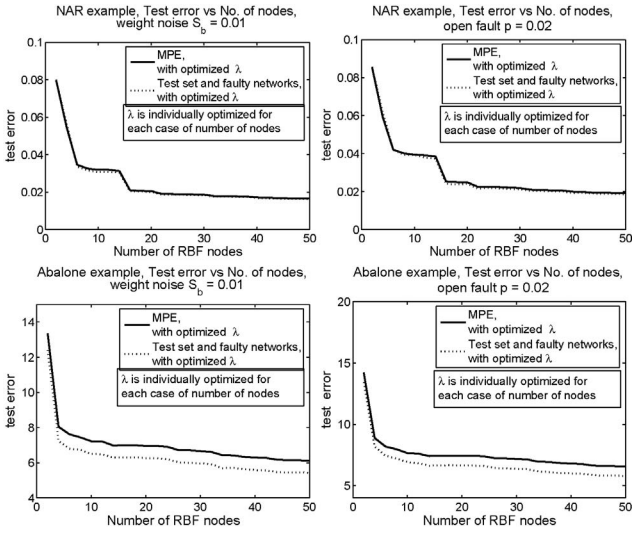

Fig. 8. Test error vs. number of nodes for the NAR and Abalone examples. The parameter λ is individually optimized for each case of number of nodes.

In Fig. 7(a), we show the situation when the decay parameter is fixed to a small value. The two curves, generated by the test set and MPE methods, have similar shapes. This result confirms the applicability of the MPE formulae and our searching method for the selecting number of nodes. In Fig. 7(a), the networks are nearly not regularized. Hence, we observe that the two curves are in "V" shape. When the network size is too small, the trained network may not be able to solve the problem well. On the other hand, when the network size is too large, the trained network usually has a poor generalization ability. From the figure, even if the decay parameter is not optimized, our MPE formulae can be used for estimating the test error and selecting the number of nodes.

When an optimized value of λ is used for each case of number of nodes, the curves of the test errors are in L shape, as shown in Fig. 7(b). It is because the networks are regularized and the number of effective free parameters is restricted. From Fig. 7(b), for this weight noise level, around 20 nodes are sufficient to handle the NAR problem because further increasing the number of nodes does not greatly improve the performance. Fig. 8 shows more results on the test errors with optimized λ's for the NAR example and Abalone examples. From the figure, the two curves, generated from the test set method with faulty networks and our MPE methods, are quite similar in shape. For the NAR example, with $S_b = 0.01$ and $p = 0.02$, around 20–30 nodes are sufficient to handle the NAR problem because further increasing the number of nodes does not greatly improve the performance. For the Abalone example, although there are some differences between the estimated MPE values and the measured test errors, the two curves, generated from the test set method with faulty networks and our MPE methods, are quite similar in shape. Hence, we can use the MPE curves to select the number of nodes.

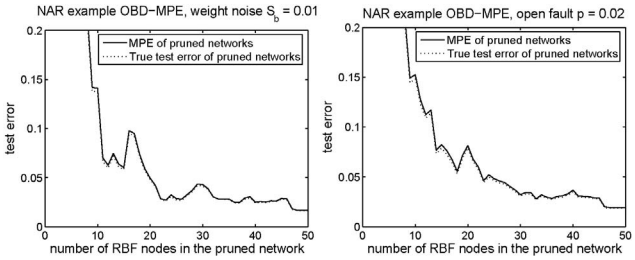

Fig. 9. Test error vs. number of nodes for the OBD-MPE approach.

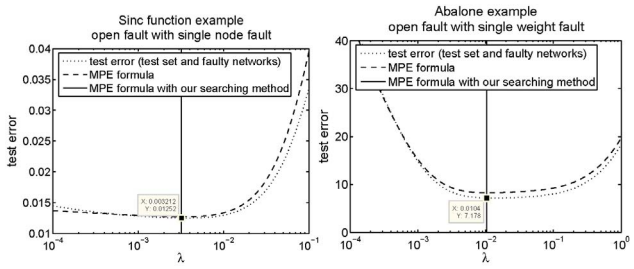

Fig. 10. MPE value and the true test error for the sinc function example and the Abalone example for the single weight fault case. The vertical solid line indicates the optimized λ based on our searching method and MPE formulae.

TABLE V
PERFORMANCE OF VARIOUS APPROACHES FOR THE SINGLE WEIGHT CASE

	Weight-Decay	Pseudo-Inverse	Fault Regularizer
Test Error (sinc)	0.01252	0.01651	0.013565
Mean of $\|w_i\|'s$ (sinc)	0.1111	0.1943	0.135
Test Error (Abalone)	7.1783	10403	7.3045
Mean of $\|w_i\|'s$ (Abalone)	1.7925	139.37	2.6538

D. Pruning on a Trained RBF Network

In Section VI-C, we train a number of RBF networks with different number of nodes. Afterwards, we can select a suitable model based on our MPE results. However, the drawback is that we need to construct several RBF networks.

In this section, we will demonstrate another approach to construct an RBF network based on the idea of optimal brain damage (OBD) [25] and our MPE results. In this OBD-MPE approach, we first train a large RBF network with the weight-decay method. The optimal decay parameter is selected from our approach. Afterwards, we use the OBD concept to rank RBF nodes. Lastly, we one by one delete unimportant RBF nodes from the ranking list and use our MPE result to estimate the test errors of the pruned networks. Based on the estimated test errors, we can select an appropriate network from those pruned networks.

To illustrate the idea, the NAR example is considered. The RBF network contains 50 RBF nodes. Other settings are the same as those in Section VI-A. Fig. 9 shows the result for the cases of multiplicative weight noise (with $S_b = 0.01$) and

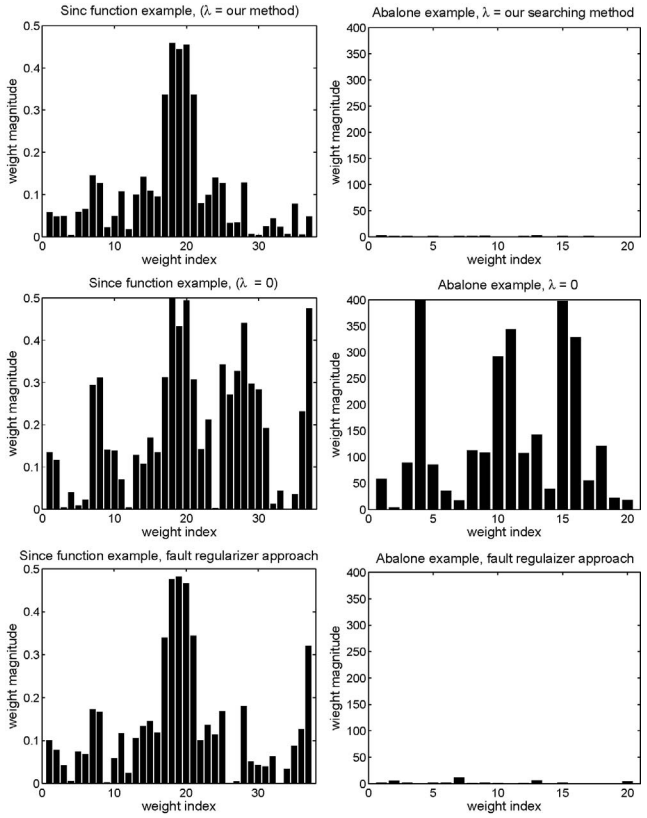

Fig. 11. Weight magnitude of the trained networks.

open weight fault (with $p = 0.02$).[5] The figure shows how the test errors increases as the number of nodes decreases. The shape of the two curves (the MPE and test set methods) are similar.

For the weight noise case with $S_b = 0.01$, there is a significant increase in the test error when the number of nodes in the pruned network is less than 23. That means, from the OBD-MPE approach, we can use 23 nodes to solve this problem and the test error is around 0.026. In Section VI-C (Fig. 8), when we use around 23 nodes, the test error is around 0.019. For the open fault case with $p = 0.02$, there is a significant increase in the test error when the number of nodes in the pruned network is less than 33. That means, from the OBD-MPE approach, we can use 33 nodes to solve this problem and the test error is around 0.027. In Section VI-C (Fig. 8), when we use around 33 nodes, the test error is around 0.02.

In term of test error, the Section VI-C method is better. It is because the weight vector of the OBD-MPE approach is initially optimized with respect to all initial nodes. Deleting some unimportant nodes will lead to the situation that the unpruned weights will not best fit for the unpruned nodes. On the other hand, in the method in Section VI-C, for each case of number of RBF nodes, the weight vector is optimized but multiple networks with different number of nodes are trained. In the OBD-MPE approach, we need to train one network only. To sum up, the pros and cons of the OBD-MPE approach are,

[5]For other fault levels, not shown in the paper, we also obtain the similar consistent result.

that it does not need to train multiple networks with different number of nodes but the test error is not optimized.

E. Performance on the Single Weight Fault Case

For the open weight fault, our analysis can handle multi-weight fault. One special case of the open weight fault is the single weight fault, where one and only one of M weights is opened. To illustrate the performance on the single weight fault case, we consider the sinc function example and the Abalone example. In the single weight fault case, the fault rate p is set to $\frac{1}{M}$. The sinc function example and Abalone example use 37 and 20 RBF nodes, respectively. The simulation results are presented in Fig. 10. Our searching results are indicated by vertical lines in the figure. From the figure, our MPE result can help us to select an appropriate value of λ for minimizing the true test error for the single node fault case.

To further investigate the performance, we compare the weight-decay approach with the fault regularizer approach [26] and the pseudo inverse approach (with $\lambda = 0$). The performance and the magnitude of those approaches are summarized in Table V and Fig. 11. From the table and figure, the performance of the pseudo inverse approach is much poorer than that of our MPE weight-decay approach and the fault regularizer approach because the weight magnitude of the pseudo inverse approach is greater than that of the other two approaches. For the fault regularizer approach, its performance is poorer than that of our MPE weight-decay approach. It is because the fault regularizer approach [26] is to optimize the training error while the weight-decay approach with our MPE weight-decay approach is to optimize the test error.

VII. Conclusion

In this paper, the error analysis on the two faulty RBF networks was presented. The MPE formulae with weight-decay for weight noise and node fault were developed. Also, a searching method for decay parameter was developed. Simulation results show that our MPE results can help us to select an appropriate value of decay parameter for minimizing the true test error of faulty networks. We also used a number of examples to demonstrate how to extend our MPE results of faulty networks for selecting RBF settings. Since the generalization error plays an important role in neural networks, it is interesting to further explore other applications or theoretical issues of the MPE result for faulty networks. Although our discussion focused on RBF networks, one can follow our derivation to handle multilayer networks with other activations, such as sigmoid and hyperbolic tangent. Of course, in such an extended case, we should use some linearizable techniques to linearize the network function of a multilayer network.

Appendix A

Let $\Upsilon = \left\langle y\Phi^T(\mathbf{x})\mathbf{w}\right\rangle_{\mathcal{D}_t}$. From (14), we have

$$\Upsilon = \left\langle \sum_{i=1}^{N} y_i \Phi^T(\mathbf{x}_i)\left(\mathbf{H}_\phi + \lambda\mathbf{I}\right)^{-1}\frac{1}{N}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)y_j \right\rangle_{\mathcal{D}_t}. \quad (49)$$

Substituting $y_i = \mathbf{w}_o^T\Phi(\mathbf{x}_i) + e_i$

$$\begin{aligned}\Upsilon = \frac{1}{N}\Bigg\langle &\sum_{i=1}^{N}\mathbf{w}_o^T\Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_i)(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\cdot\\ &\sum_{j=1}^{N}\Phi(\mathbf{x}_j)\Phi^T(\mathbf{x}_j)\mathbf{w}_o\\ &+\sum_{i=1}^{N}\mathbf{w}_o^T\Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_i)(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)e_j\\ &+\sum_{i=1}^{N}e_i\Phi^T(\mathbf{x}_i)(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)\Phi^T(\mathbf{x}_j)\mathbf{w}_o\\ &+\sum_{i=1}^{N}e_i\Phi^T(\mathbf{x}_i)(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)e_j\Bigg\rangle_{\mathcal{D}_t}. \end{aligned}\quad (50)$$

Since $\mathbf{H}_\phi = \frac{1}{N}\sum_{i=1}^{N}\Phi(\mathbf{x}_i)\Phi^T(\mathbf{x}_i)$, we have

$$\begin{aligned}\Upsilon =&\mathbf{w}_o^T\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o\\ &+\left\langle\frac{1}{N}\sum_{i=1}^{N}e_i\Phi^T(x_i)\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\sum_{j=1}^{N}e_j\Phi(\mathbf{x}_j)\right\rangle_{\mathcal{D}_t}. \end{aligned}\quad (51)$$

As e_i's are independent

$$\begin{aligned}\Upsilon =\ &\mathbf{w}_o^T\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o\\ &+\frac{Se}{N}\mathbf{Tr}\left\{\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\right\}. \end{aligned}\quad (52)$$

References

[1] A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Behavioral fault model for neural networks," in *Proc. Int. Conf. Comput. Eng. Technol.*, Jan. 2009, pp. 71–75.

[2] A. Ahmadi, M. H. Sargolzaie, S. M. Fakhraie, C. Lucas, and S. Vakili, *A Low-Cost Fault-Tolerant Approach for Hardware Implementation of Artificial Neural Networks*, vol. 2. Los Alamitos, CA: IEEE Comput. Soc., Jan. 2009, pp. 93–97.

[3] S. Amari, N. Murata, K. R. Muller, M. Finke, and H. H. Yang, "Asymptotic statistical theory of overtraining and cross-validation," *IEEE Trans. Neural Netw.*, vol. 8, no. 5, pp. 996–985, Sep. 1997.

[4] R. Battiti, "Accelerated backpropagation learning: Two optimization methods," *Complex Syst.*, vol. 3, pp. 331–342, Aug. 1989.

[5] J. L. Bernier, J. Ortega, M. M. Rodriguez, I. Rojas, and A. Prieto, "An accurate measure for multilayer perceptron tolerance to weight deviations," *Neural Process. Lett.*, vol. 10, no. 2, pp. 121–130, Oct. 1999.

[6] J. L. Bernier, J. Ortega, I. Rojas, E. Ros, and A. Prieto, "Obtaining fault tolerant multilayer perceptrons using an explicit regularization," *Neural Process. Lett.*, vol. 12, no. 2, pp. 107–113, Oct. 2000.

[7] J. L. Bernier, J. Ortega, E. Ros, I. Rojas, and A. Prieto, "A quantitative study of fault tolerance, noise immunity, and generalization ability of MLPs," *Neural Comput.*, vol. 12, no. 12, pp. 2941–2964, Dec. 2000.

[8] G. Bolt, "Fault tolerant multilayer perceptron networks," Ph.D. dissertation, Dept. Comput. Sci., Univ. of York, York, U.K., Jul. 1992.

[9] J. Burr, "Digital neural network implementations," in *Neural Networks, Concepts, Applications, and Implementations*, Vol III. Englewood Cliffs, NJ: Prentice-Hall, 1995, pp. 237–285.

[10] S. Cavalieri and O. Mirabella, "A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks," *Neural Netw.*, vol. 12, no. 1, pp. 91–106, Jan. 1999.

[11] P. Chandra and Y. Singh, "Fault tolerance of feedforward artificial neural networks: A framework of study," in *Proc. Int. Joint Conf. Neural Netw. 2003*, vol. 1. Jul. 2003, pp. 489–494.

[12] P. Chandra and Y. Singh, "Feedforward sigmoidal networks: Equicontinuity and fault-tolerance properties," *IEEE Trans. Neural Netw.*, vol. 15, no. 6, pp. 1350–1366, Nov. 2004.

[13] S. Chen, X. Hong, C. Harris, and P. Sharkey, "Sparse modeling using orthogonal forward regression with press statistic and regularization," *IEEE Trans. Syst., Man, Cybern. B*, vol. 34, no. 2, pp. 898–911, Mar. 2004.
[14] C. T. Chiu, K. Mehrotra, C. K. Mohan, and S. Ranka, "Modifying training algorithms for improved fault tolerance," in *Proc. Int. Conf. Neural Netw.*, vol. 4. Jun. 1994, pp. 333–338.
[15] F. M. Dias and A. Antunes, "When response variability increases neural network robustness to synaptic noise," *Neural Comput.*, vol. 18, no. 6, pp. 1349–1379, Jun. 2006.
[16] M. D. Emmerson and R. I. Damper, "Determining and improving the fault tolerance of multilayer perceptrons in a pattern-recognition application," *IEEE Trans. Neural Netw.*, vol. 4, no. 5, pp. 788–793, Sep. 1993.
[17] V. V. Fedorov, *Theory of Optimal Experiments*. New York: Academic, 1972.
[18] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Comput.*, vol. 4, no. 1, pp. 1–58, Jan. 1992.
[19] P. Guo, "Studies of model selection and regularization for generalization in neural networks with applications," Ph.D. dissertation, Chin. Univ. Hong Kong, Hong Kong, 2002.
[20] P. Guo, M. Lyu, and C. Chen, "Regularization parameter estimation for feedforward neural networks," *IEEE Trans. Syst., Man, Cybern. B*, vol. 33, no. 1, pp. 35–44, Jan. 2003.
[21] T. Haruhiko, M. Masahiko, K. Hidehiko, and H. Terumine, "Enhancing both generalization and fault tolerance of multilayer neural networks," in *Proc. Int. Joint Conf. Neural Netw. 2007*, Aug. 2007, pp. 1429–1433.
[22] S. Himavathi, D. Anitha, and A. Muthuramalingam, "Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 880–888, May 2007.
[23] A. Krogh and J. A. Hertz, "A simple weight-decay can improve generalization," in *Proc. NIPS4 Conf.*, 1992, pp. 950–957.
[24] J. Larsen, "Design of neural network filters," Ph.D. dissertation, Tech. Univ. Denmark, Lyngby, Denmark, Jul. 1993.
[25] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. NIPS2 Conf.*, 1990, pp. 598–605.
[26] C. S. Leung and J. Sum, "A fault-tolerant regularizer for rbf networks," *IEEE Trans. Neural Netw.*, vol. 19, no. 3, pp. 493–507, May 2008.
[27] C. S. Leung, G. H. Young, J. Sum, and W. K. Kan, "On the regularization of forgetting recursive least square," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1482–1486, Nov. 1999.
[28] C. S. Leung, A. Tsoi, and L. Chan, "Two regularizers for recursive least square algorithms in feedforward multilayered neural networks," *IEEE Trans. Neural Netw.*, vol. 12, no. 6, pp. 1314–1332, Nov. 2001.
[29] B. Liu and T. Kaneko, "Error analysis of digital filter realized with floating-point arithmetic," *Proc. IEEE*, vol. 57, no. 10, pp. 1735–1747, Oct. 1969.
[30] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, Jul. 1977.
[31] J. E. Moody, "Note on generalization, regularization, and architecture selection in nonlinear learning systems," in *Proc. 1st IEEE-SP Workshop Neural Netw. Signal Process.*, Sep. 1991, pp. 1–10.
[32] J. E. Moody, S. J. Hanson, and R. P. Lippmann, "The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems," in *Proc. NIPS4 Conf.*, 1992, pp. 847–854.
[33] N. Murata, S. Yoshizawa, and S. Amari, "Network information criterion determining the number of hidden units for an artificial neural network model," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 865–872, Nov. 1994.
[34] A. F. Murray and P. J. Edwards, "Enhanced MLP performance and fault tolerance from synaptic weight noise during training," *IEEE Trans. Neural Netw.*, vol. 5, no. 5, pp. 792–802, Sep. 1994.
[35] M. J. L. Orr, "Regularization in the selection of radial basis function centers," *Neural Comput.*, vol. 7, no. 3, pp. 606–623, May 1995.
[36] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," *IEEE Comput.*, vol. 25, no. 5, pp. 76–79, May 1992.
[37] D. S. Phatak and I. Koren, "Complete and partial fault tolerance of feedforward neural nets," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 446–456, Mar. 1995.
[38] S. W. Piche, "The selection of weight accuracies for madalines," *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 432–445, Mar. 1995.
[39] T. S. Rögnvaldsson, "A simple trick for estimating the weight-decay parameter," in *Neural Networks: Tricks of the Trade*. London, U.K.: Springer-Verlag, 1998, pp. 71–92.
[40] C. Sequin and R. Clay, "Fault tolerance in artificial neural networks," in *Proc. Int. Joint Conf. Neural Netw. 1990*, Jan. 1990, pp. 703–708.
[41] N. C. Steele and J. H. Tabor, "On parity problems and the functional-link artificial neural network," *Neural Comput. Applicat.*, vol. 2, no. 4, pp. 205–208, Dec. 1994.
[42] M. Stevenson, R. Winter, and B. Widrow, "Sensitivity of feedfoward neural networks to weight errors," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 71–80, Jan. 1990.
[43] M. Sugiyama and H. Ogawa, "Optimal design of regularization term and regularization parameter by subspace information criterion," *Neural Netw.*, vol. 15, no. 3, pp. 349–361, Apr. 2002.
[44] E. Tchernev, R. Mulvaney, and D. Phatak, "Investigating the fault tolerance of neural networks," *Neural Comput.*, vol. 17, no. 7, pp. 1646–1664, Jul. 2005.
[45] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the backpropagation method," *Biol. Cybern.*, vol. 59, no. 4-5, pp. 257–263, Dec. 1988.
[46] L. Wu and J. E. Moody, "A smoothing regularizer for feedforward and recurrent neural networks," *Neural Comput.*, vol. 8, no. 3, pp. 461–489, Apr. 1996.
[47] Z. H. Zhou and S. F. Chen, "Evolving fault-tolerant neural networks," *Neural Comput. Applicat.*, vol. 11, nos. 3–4, pp. 156–160, Jun. 2003.
[48] Z. H. Zhou, S. F. Chen, and Z. Q. Chen, "Improving tolerance of neural networks against multinode open fault," in *Proc. Int. Joint Conf. Neural Netw.*, vol. 3. Jul. 2001, pp. 1687–1692.

Chi-Sing Leung (M'04) received the B.S. degree in electronics, the M.Phil. degree in information engineering, and the Ph.D. degree in computer science, all from the Chinese University of Hong Kong, Shatin, Hong Kong, in 1989, 1991, and 1995, respectively.

He is currently an Associate Professor with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong. His current research interests include neural computing, data mining, and computer graphics.

Dr. Leung received the 2005 IEEE Transactions on Multimedia Prize Paper Award for his paper titled, "The Plenoptic Illumination Function" published in 2002. He was a member of Organizing Committee of ICONIP2006. He is a Guest Editor of the *Neurocomputing* and *Neural Computing and Applications*. He is the Program Chair of ICONIP2009. He is also a Governing Board Member of the Asian Pacific Neural Network Assembly.

Hong-Jiang Wang received the B.Eng. degree from the Department of Electronic and Electrical Engineering, Ningbo University, Ningbo, China, in 2000 and the Ph.D. degree from the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China, in 2007.

His current research interests include neural computing and wireless communication networks.

John Sum (SM'05) received the B.Eng. degree in electronic engineering from the Hong Kong Polytechnic University, Kowloon, Hong Kong, in 1992, and the M.Phil. and Ph.D. degrees in computer science engineering from the Chinese University of Hong Kong, Shatin, Hong Kong, in 1995 and 1998, respectively.

He spent 6 years teaching in several universities, including the Kong Baptist University, Kowloon, Hong Kong, the Open University of Hong Kong, Hong Kong, and the Hong Kong Polytechnic University, Kowloon. In 2005, he moved to Taiwan and began teaching in Chung Shan Medical University, Taiwan. Currently, he is an Assistant Professor with the Institute of Electronic Commerce, the National Chung Hsing University, Taichung, Taiwan. His current research interests include neural computation, mobile sensor networks, and scale-free network.

Dr. Sum is an Associate Editor of the *International Journal of Computers and Applications*.