A Fault-Tolerant Regularizer for RBF Networks

Chi-Sing Leung, *Member, IEEE*, and John Pui-Fai Sum, *Senior Member, IEEE*

***Abstract*—In classical training methods for node open fault, we need to consider many potential faulty networks. When the multinode fault situation is considered, the space of potential faulty networks is very large. Hence, the objective function and the corresponding learning algorithm would be computationally complicated. This paper uses the Kullback–Leibler divergence to define an objective function for improving the fault tolerance of radial basis function (RBF) networks. With the assumption that there is a Gaussian distributed noise term in the output data, a regularizer in the objective function is identified. Finally, the corresponding learning algorithm is developed. In our approach, the objective function and the learning algorithm are computationally simple. Compared with some conventional approaches, including weight-decay-based regularizers, our approach has a better fault-tolerant ability. Besides, our empirical study shows that our approach can improve the generalization ability of a fault-free RBF network.**

***Index Terms*—Kullback–Leibler divergence, node open fault, regularization.**

NOMENCLATURE

$\mathbf{x}$	K-dimensional input pattern.
y	1-D output of the system.
$\{\mathbf{x}_i, y_i\}$	The ith input–output training pair.
K	Input dimension.
N	Number of training samples.
$\mathcal{D}$	Training set.
$f(\mathbf{x})$	Unknown system.
e	Measurement noise.
σ^2	Variance of e.
$\mathcal{P}_{\mathcal{D}}(\mathbf{x})$	Density function of input.
$\mathcal{P}_{\mathcal{D}}(y \mid \mathbf{x})$	Conditional density function of y given $\mathbf{x}$.
$\mathcal{P}_{\mathcal{D}}(y, \mathbf{x})$	Joint density function of $\mathbf{x}$ and y.
$\hat{f}(\mathbf{x}, \mathbf{w})$	RBF network function.
w_j	The jth RBF weight.
M	Number of RBF nodes.
$\mathbf{w}$	RBF weight vector.
$\phi_j(\mathbf{x})$	The jth kernel function.
$\mathbf{c}_j$	The jth RBF center.
Δ	Width of RBFs.
$\Phi(\mathbf{x})$	The collection of RBF values.
$\tilde{w}_j = b_j w_j$	The jth weight of a faulty network.
b_j	The fault factor of the jth RBF node.
$\mathbf{b}$	Fault vector.
p	Fault rate.
$\tilde{\mathbf{w}}$	Weight vector of a faulty network.
$E(\mathbf{w})$	MSE of a fault-free network.
$E(\mathbf{w}, \mathbf{b}')$	MSE of a faulty network with fault vector $\mathbf{b}'$
$\mathcal{S}_b$	Collection of fault vectors.
$\mathcal{S}_{b^1}$	Collection of fault vectors with one fault.
$\mathcal{P}(y \mid \mathbf{x}, \mathbf{b}; \mathbf{w})$	Conditional density function of a faulty network.
$\mathcal{P}(y, \mathbf{x} \mid \mathbf{b}; \mathbf{w})$	Joint density function of the input–output of the faulty RBF network.
$\bar{D}$	Average Kullback–Leibler divergence over all possible fault vectors.
$\mathcal{E}(\mathbf{w}, p)$	The objective function of our robust method.
$\mathbf{H}_\phi$	Autocorrelation matrix of RBFs.
$\mathbf{G}$	The diagonal of $\mathbf{H}_\phi$.
$\mathbf{R}$	Regularizer matrix of our robust method.
$\hat{\mathbf{w}}$	Optimal weight vector of our robust method.

Manuscript received July 14, 2006; revised June 22, 2007; accepted July 2, 2007. This work was supported by the City University of Hong Kong under Grant 7002108.

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

J. P.-F. Sum was with the Department of Electronic Engineering, the City University of Hong Kong, Kowloon Tong, Hong Kong. He is now with the Institute of Electronic Commerce, National Chung Hsing University, Taichung 402, Taiwan.

Digital Object Identifier 10.1109/TNN.2007.912320

I. INTRODUCTION

IT WAS commonly assumed that neural networks have a built-in ability against node failures. In fact, many literatures [1]–[4] showed that if special care is not taken during training, the fault situation could lead to a drastic performance degradation. Hence, obtaining a fault-tolerant neural network is of paramount importance. Among many fault models [5], an important fault model is the multinode open fault [5]–[8]. In this fault model, some hidden nodes are disconnected to the output layer. Several algorithms for handling this fault model have been developed. They can be roughly classified into three categories.

In the first category, the trick is to add some heuristics during training. Injecting random node fault [4], [9] during training is a typical example. In [6], Zhou proposed the powerful T3 algorithm to train a network. The algorithm identifies the break point from the fault curve. Afterwards, injecting random node fault is used to train the network. However, when the size of networks is large, the space of potential faulty networks is very large, and then extensive amount of training time is required. Since the output of a network is very sensitive to large weights,

another technique is to limit the weight magnitude. Limiting weight magnitude can be achieved by adding a weight decay regularizer [4] or hard bounding the weight magnitude [10]. One deficiency of limiting weight magnitude is that the theoretical support of the underlying objective function is not so clear. For example, in [4], the way to set the weight decay constant is not discussed even though the fault statistics is available.

The second category considers the replication technique in which hidden nodes are replicated from a trained network [5], [8]. However, in this approach, we need to use an additional source.

In the third category, the idea is to formulate the training process as solving unconstraint or constraint optimization problems. Neti *et al.* [11] and Deodhare *et al.* [12] defined the training objective as a minimax problem. The beauty of these approaches is that the nice objective functions are defined. However, their drawback is that solving a minmax problem is very complicated. Simon and Sherief [13] and Phatak and Tchernev [14] formulated the learning problem as *an unconstrained optimization problem in which the objective function consists of two terms*. The first term is the mean square error (MSE) of a fault-free network. The second term is the sum of MSEs of faulty networks. Zhou *et al.* [7] defined a similar objective function and developed the corresponding learning algorithm. They also empirically showed that the proposed objective function can improve generalization and fault tolerance.

The aforementioned formulations are effective to handle single-node fault. However, they are computationally complicated when the multinode open fault situation is considered. For example, in a radial basis function (RBF) network [15]–[19] with M nodes, for multinode fault, the number of potentially faulty networks is $N_{\text{networks}} = \sum_{i=1}^{n} C_n^M$, where n is the maximum number of faulty nodes. Also, in [7], the theoretical guideline to set the weighting factors of those MSE terms was not addressed.

Many regularization techniques [20] for improving generalization, such as weight decay, can improve fault tolerance [4]. Based on the Vapnik-Cervonenkis (VC) dimension, Phatak [21] qualitatively explained why adding redundancy can improve fault tolerance and generalization. It means that the training methods that improve fault tolerance can also lead to a better generalization and vice versa [6], [7] [13], [22].

In regularization techniques [20], [23]–[25] for improving the generalization ability, these training algorithms are usually computationally simple. In view of this, it is interesting to develop a regularization term for handling multinode open fault and to develop the corresponding computational friendly training algorithm.

This paper uses the RBF network model as an example to develop a regularizer for multinode open fault. Assuming that there is a Gaussian distributed noise term in the output data, we use the Kullback–Leibler divergence to develop an objective function for fault tolerance. Afterwards, the corresponding regularizer is identified in the objective function. With the proposed objective function, we develop a simple optimal training rule that minimizes the Kullback–Leibler divergence. The organization of this paper is as follows. In Section II, we review the concept of RBF networks and fault tolerance. In Section III, the objective function for multinode open fault and the corresponding regularizer are defined. Afterwards, the corresponding training algorithm is derived. Section IV presents our simulation results. Conclusion is presented in Section V. The Appendix describes the background of some conventional regularizers for improving the generalization.

II. Background

A. Data Model

Throughout this paper, we are given a training data set

$$\mathcal{D} = \{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \Re^K, y_i \in \Re, i = 1, \ldots, N\}$$

where $\mathbf{x}_i$ and y_i are the input and output samples of an unknown system, respectively. We assume that the data set $\mathcal{D}$ is generated by a stochastic system [26], [27], given by

$$y_i = f(\mathbf{x}_i) + e_i \tag{1}$$

where $f(\cdot)$ is the unknown system, and e_i's are the random measurement noise. The noise e_i's are independent zero-mean Gaussian random variables with variance equal to σ_e^2. The input–output relationship is specified by the conditional probability density function $\mathcal{P}_{\mathcal{D}}(y \,|\, \mathbf{x})$. Let $\mathcal{P}_{\mathcal{D}}(\mathbf{x})$ be the density function of the input $\mathbf{x}$. The joint probability density of the input–output is given by

$$\mathcal{P}_{\mathcal{D}}(y, \mathbf{x}) = \mathcal{P}_{\mathcal{D}}(\mathbf{x}) \mathcal{P}_{\mathcal{D}}(y \,|\, \mathbf{x}). \tag{2}$$

Now, our problem is to construct a model for approximating the mapping $f(\cdot)$.

B. RBF Model

In the RBF approach, the mapping $f(\cdot)$ is approximated by an RBF network, given by

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) \tag{3}$$

where $\mathbf{x} \in \Re^K$ is the input, $\mathbf{w} = [w_1, \ldots, w_M]^T$ is the RBF weight vector, and $\phi_j(\cdot)$ is the jth kernel function, given by

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{\Delta}\right). \tag{4}$$

Vectors $\mathbf{c}_j$'s are the RBF centers. The parameter $\Delta(> 0)$ controls the width of the kernel function. In the vector–matrix notation, (3) can be written as

$$\hat{f}(\mathbf{x}, \mathbf{w}) = \Phi^T(\mathbf{x}) \mathbf{w} \tag{5}$$

where $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_M(\mathbf{x})]^T$. Our learning task is to find a weight vector that best fits the observations.

C. Multinode Open Fault

A faulty RBF network can be described by a weight multiplicative model, given by

$$\tilde{w}_j = b_j w_j \tag{6}$$

for $j = 1, \ldots, M$. The fault factor b_j describes whether the jth node operates properly or not. When $b_j = 0$, the jth node is out of work. Otherwise, the jth node operates properly. Define $\mathbf{b} = [b_1, \ldots, b_M]^T$ as the fault vector. In vector–matrix notation, (6) can be rewritten as

$$\tilde{\mathbf{w}} = \mathbf{b} \otimes \mathbf{w} \tag{7}$$

where $\otimes$ is the elementwise multiplication operator.

For a fault-free network, the average error is given by

$$E(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} \|y_i - \Phi^T(\mathbf{x})\mathbf{w}\|^2. \tag{8}$$

For a particular fault vector $\mathbf{b}$, the average error is given by

$$E(\mathbf{w}, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} \|y_i - \Phi^T(\mathbf{x})\tilde{\mathbf{w}}\|^2. \tag{9}$$

In [7], [13], and [14], the following objective function is used:

$$J(\mathbf{w}) = \alpha E(\mathbf{w}) + \beta \frac{1}{h} \sum_{\mathbf{b}}{}' \in \mathcal{S}_b E(\mathbf{w}, \mathbf{b}') \tag{10}$$

where $\mathcal{S}_b$ is the set of all possible fault vectors considered, and h is the number of fault vectors in $\mathcal{S}_b$. The parameters α and β are the weighting factors. With (10), we can develop training algorithms based on the backpropagation method or genetic algorithm. In [7], it is shown that the training algorithm, based on (10), can improve fault tolerance as well as generalization. However, when the number of faulty nodes is greater than one, the number of potential faulty networks is very large. Also, the rule to set the two weighting factors was not theoretically discussed [7], [14].

III. Robust Learning for Multinode Open Fault

This section uses the Kullback–Leibler divergence to develop an objective function of RBF networks for multinode open fault. In the derivation, we assume that there is a Gaussian distributed noise term in the output data. This assumption is commonly used in the literature [26]. A regularization term for multinode open fault in RBF networks is then identified in the proposing objective function. Finally, based on the objective function, we develop a simple optimal training rule that minimizes the Kullback–Leibler divergence between the data set model and the network model.

A. Kullback–Leibler Divergence

Assume that the fault factors b_j's are identical, independent binary random variables with $\mathbf{Prob}(b_j = 0) = p$ and $\mathbf{Prob}(b_j = 1) = 1 - p$. With the fault model (7), we can assume that the data set $\mathcal{D}$ is generated by the following stochastic model:

$$y_i = \hat{f}(\mathbf{x}_i, \tilde{\mathbf{w}}) + e_i = \Phi^T(\mathbf{x})\tilde{\mathbf{w}} + e_i. \tag{11}$$

Based on the stochastic model in the neural networks [26], the input–output relationship of the faulty RBF network is specified by the conditional probability density function $\mathcal{P}(y \,|\, \mathbf{x}, \mathbf{b}; \mathbf{w})$. The conditional density function is Gaussian. The mean and variance of y are equal to $\Phi^T(\mathbf{x})\tilde{\mathbf{w}}$ and σ_e^2, respectively. The joint probability density function of the input–output is then given by

$$\mathcal{P}(y, \mathbf{x} \,|\, \mathbf{b}; \mathbf{w}) = \mathcal{P}_{\mathcal{D}}(\mathbf{x}) \mathcal{P}(y \,|\, \mathbf{x}, \mathbf{b}; \mathbf{w}). \tag{12}$$

To measure the discrepancy between a faulty RBF network and the data set, we use the Kullback–Leibler divergence [28], given by

$$\int \int \mathcal{P}_{\mathcal{D}}(y, \mathbf{x}) \log \frac{\mathcal{P}_{\mathcal{D}}(y, \mathbf{x})}{\mathcal{P}(y, \mathbf{x} \,|\, \mathbf{b}; \mathbf{w})} d\mathbf{x}\, dy. \tag{13}$$

As $\mathbf{b}$ is a random fault vector, we need to consider the average Kullback–Leibler divergence over all possible fault vectors, given by

$$\bar{D} = \left\langle \int \int \mathcal{P}_{\mathcal{D}}(y, \mathbf{x}) \log \frac{\mathcal{P}_{\mathcal{D}}(y, \mathbf{x})}{\mathcal{P}(y, \mathbf{x} \,|\, \mathbf{b}; \mathbf{w})} d\mathbf{x}\, dy \right\rangle_{\mathcal{S}_b} \tag{14}$$

where $\langle \cdot \rangle$ is the expectation operator, and $\mathcal{S}_b$ denotes the set of all possible fault vectors.

As $\mathcal{P}_{\mathcal{D}}(y, \mathbf{x})$ is independent of $\mathbf{b}$ and $\mathbf{w}$

$$\bar{D} = \text{const.} - \left\langle \int \int \mathcal{P}_{\mathcal{D}}(y, \mathbf{x}) \log \mathcal{P}(y, \mathbf{x} \,|\, \mathbf{b}; \mathbf{w}) d\mathbf{x}\, dy \right\rangle_{\mathcal{S}_b}. \tag{15}$$

From (12), (15) can be rewritten as

$$\bar{D} = \text{const.} - \left\langle \int \int \mathcal{P}_{\mathcal{D}}(y, \mathbf{x}) \log \mathcal{P}(y \,|\, \mathbf{x}, \mathbf{b}; \mathbf{w})\, d\mathbf{x}\, dy \right\rangle_{\mathcal{S}_b}. \tag{16}$$

From (11), $\mathcal{P}(y \,|\, \mathbf{x}, \mathbf{b}; \mathbf{w})$ is given by

$$\mathcal{P}(y \,|\, \mathbf{x}, \mathbf{b}; \mathbf{w}) = \frac{1}{\sqrt{2\pi\sigma_e^2}} \exp\left(-\frac{(y - \Phi^T(\mathbf{x})\,\tilde{\mathbf{w}})^2}{2\sigma_e^2} \right). \tag{17}$$

Since the logarithm of $\mathcal{P}(y \,|\, \mathbf{x}, \mathbf{b}; \mathbf{w})$ is

$$\log \mathcal{P}(y \,|\, \mathbf{x}, \mathbf{b}; \mathbf{w}) = -\frac{1}{2} \log 2\pi\sigma_e^2 - \frac{(y - \Phi^T(\mathbf{x})\,\tilde{\mathbf{w}})^2}{2\sigma_e^2} \tag{18}$$

we have

$$\bar{D} = \text{const.} + \left\langle \int \int \mathcal{P}_{\mathcal{D}}(y, \mathbf{x})(y - \Phi^T(\mathbf{x})\,\tilde{\mathbf{w}})^2\, d\mathbf{x}\, dy \right\rangle_{\mathcal{S}_b}. \tag{19}$$

From (19), our task is to find the expression for $\langle \int \int \mathcal{P}_{\mathcal{D}}(y, \mathbf{x})(y - \Phi^T(\mathbf{x})\,\tilde{\mathbf{w}})^2\, d\mathbf{x} dy \rangle_{\mathcal{S}_b}$. Afterwards, we minimize (19) with respect to the weight vector.

B. Objective Function

In (19), the second term is positive. Hence, minimizing the average Kullback–Leibler divergence $\bar{D}$ is equivalent to minimizing this second term with respect to the weight vector. The second term in (19) can be rewritten as

$$\mathcal{E}(\mathbf{w},p) = \int\int \mathcal{P}_{\mathcal{D}}(y,\mathbf{x}) \left\{ y^2 - 2y \left\langle \sum_{j=1}^{M} b_j w_j \phi_j(\mathbf{x}) \right\rangle_{\mathcal{S}_b} + \left\langle \left(\sum_{j=1}^{M} b_j w_j \phi_j(\mathbf{x}) \right)^2 \right\rangle_{\mathcal{S}_b} \right\} d\mathbf{x}\, dy. \tag{20}$$

Since $\langle b_j \rangle = \langle b_j^2 \rangle = 1-p$ and $\langle b_j b_{j'} \rangle = (1-p)^2$ for $j \neq j'$, we have

$$\left\langle \sum_{j=1}^{M} b_j w_j \phi_j(\mathbf{x}) \right\rangle_{\mathcal{S}_b} = (1-p) \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}) \tag{21}$$

$$\left\langle \left(\sum_{j=1}^{M} b_j w_j \phi_j(\mathbf{x}) \right)^2 \right\rangle_{\mathcal{S}_b} = \mathbf{w}^T ((1-p)\mathbf{G} + (1-p)^2 (\mathbf{H}_\phi - \mathbf{G}))\mathbf{w} \tag{22}$$

where

$$\mathbf{H}_\phi = \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i) \Phi^T(\mathbf{x}_i) \tag{23}$$

and

$$\mathbf{G} = \mathbf{diag}(\mathbf{H}_\phi). \tag{24}$$

If the number N of samples is large enough, we have $\int\int \mathcal{P}_{\mathcal{D}}(y,\mathbf{x}) y^2\, d\mathbf{x}\, dy = (1)/(N) \sum_{i=1}^{N} y_i^2$, $\int\int \mathcal{P}_{\mathcal{D}}(y,\mathbf{x}) y \phi_j(\mathbf{x})\, d\mathbf{x}\, dy = (1)/(N) \sum_{i=1}^{N} y_i\, \phi_j(\mathbf{x}_i)$, and $\int\int \mathcal{P}_{\mathcal{D}}(y,\mathbf{x}) \phi_j(\mathbf{x}) \phi_{j'}(\mathbf{x})\, d\mathbf{x}\, dy = (1)/(N) \sum_{i=1}^{N} \phi_j(\mathbf{x}) \phi_{j'}(\mathbf{x}_i)$. Now, the objective function $\mathcal{E}(\mathbf{w},p)$ can be rewritten as

$$\mathcal{E}(\mathbf{w},p) = \frac{1}{N} \sum_{i=1}^{N} y_i^2 - 2(1-p) \frac{1}{N} \sum_{k=1}^{N} y_i \Phi^T(\mathbf{x})\, \mathbf{w} + (1-p)\mathbf{w}^T \{(1-p)\mathbf{H}_\phi + p\mathbf{G}\}\mathbf{w}. \tag{25}$$

C. Regularizer for Multinode Open Fault

Since the term $(1)/(N) \sum_{i=1}^{N} y_i^2$ in (25) is independent of $\mathbf{w}$, we can rescale this term by a constant $(1-p)$. Thus, we have

$$\mathcal{E}(\mathbf{w},p) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \Phi^T(\mathbf{x}_i)\, \mathbf{w})^2 + \mathbf{w}^T \mathbf{R} \mathbf{w} \tag{26}$$

where

$$\mathbf{R} = p(\mathbf{G} - \mathbf{H}_\phi). \tag{27}$$

In (26), the second term is similar to the conventional regularization term in regularization techniques [29], [23] [15], [30], [31]. Hence, we could define the *multinode open fault regularizer* as

$$\mathbf{w}^T \mathbf{R} \mathbf{w} \tag{28}$$

where $\mathbf{R}$ is the so-called regularization matrix.

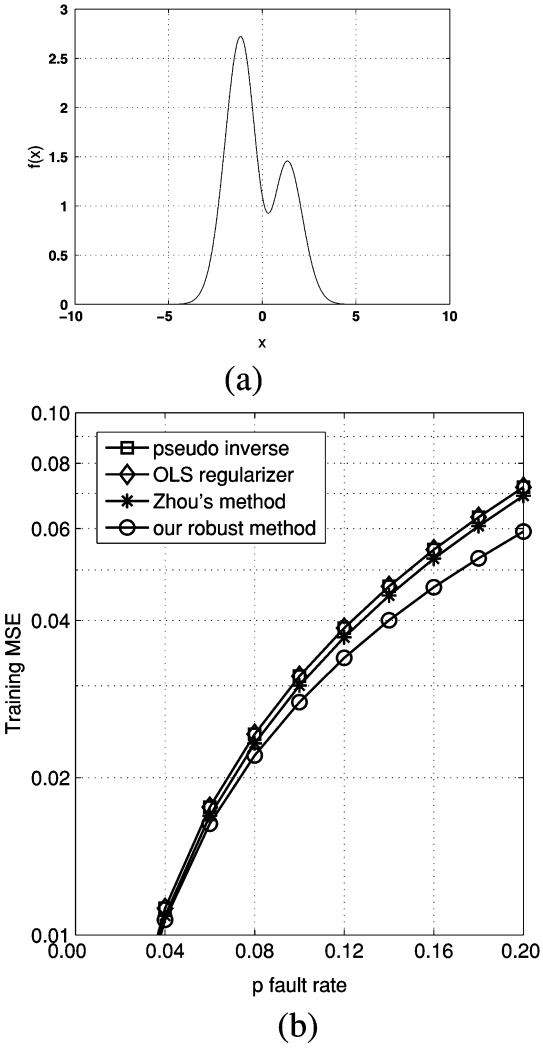

Fig. 1. (a) Hermite function. (b) Training set MSE of faulty networks for the Hermite function approximation.

D. Our Robust Algorithm

Taking the derivative of (26) with respect to $\mathbf{w}$, we have

$$\frac{\partial \mathcal{E}(\mathbf{w},p)}{\partial \mathbf{w}} = (\mathbf{H}_\phi + \mathbf{R})\, \mathbf{w} - \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i) y_i. \tag{29}$$

Hence, the optimal RBF weight vector for minimizing the Kullback–Leibler divergence is given by

$$\hat{\mathbf{w}} = (\mathbf{H}_\phi + \mathbf{R})^{-1} \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i) y_i. \tag{30}$$

E. Property of Multinode Open Fault Regularizer

Improving the generalization ability of a neural network has been studying for more than two decades. The common tech-

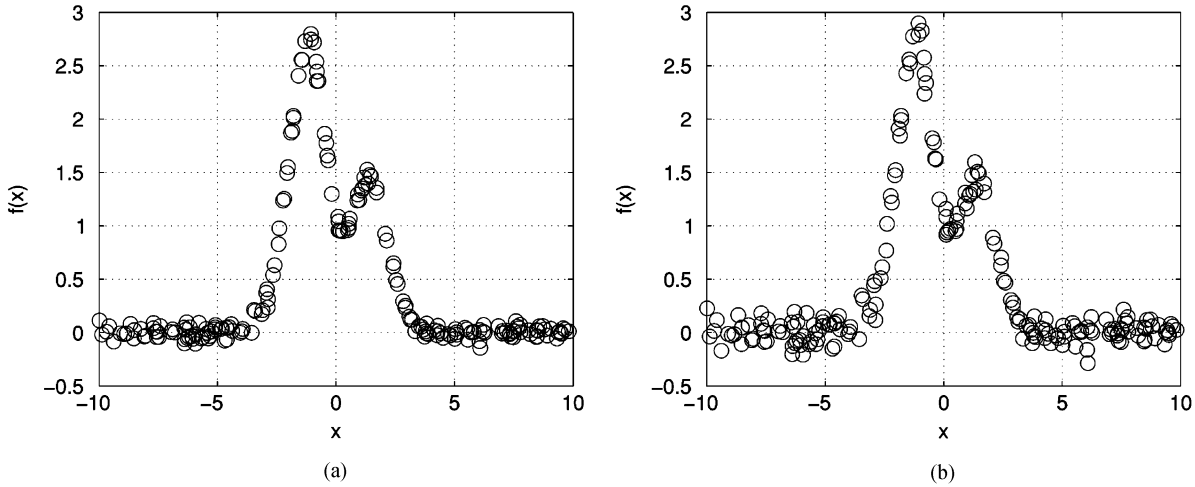

Fig. 2. Training data for the noisy Hermite function approximation: (a) $\sigma_e^2 = 0.0025$ and (b) $\sigma_e^2 = 0.01$.

nique is to add a regularization term [23], [31], such as weight decay [20] and local regularizer [32], [33], into the objective function. The objective function for multinode open fault also has a similar form, so it is interesting to know whether there is a regularizer that optimizes generalization that can also be used to optimize fault tolerance which is in terms of Kullback–Leibler divergence.

By inspecting the objective function (26), we can see that the optimal regularizer for fault tolerance is equal to $p\mathbf{w}^T(\mathbf{G} - \mathbf{H}_\phi)\mathbf{w}$. The corresponding regularization matrix is given by $\mathbf{R} = p(\mathbf{G} - \mathbf{H}_\phi)$. Since the diagonal elements of the matrix $\mathbf{R}$ are equal to zero and the matrix $\mathbf{R}$ is symmetric, the summation of all eigenvalues of $\mathbf{R}$ must be zero and all its eigenvalues must be real. **This means that for multinode open fault tolerance, the regularization matrix should contain both positive and negative eigenvalues**. Now, our question is "Do regularizers for improving generalization have the similar property?" From the Appendix, the regularizer matrices of the common regularizers, such as the Tipping's regularizer and the Chen's regularizer [32], [33], are positive definite or semipositive definite. Hence, in the sense of Kullback–Leibler divergence, the Tipping's regularizer and the Chen's regularizer cannot achieve the optimal fault tolerance for multinode open fault.

IV. Simulations

In our simulations, two rough guidelines are used for selecting the number of RBF nodes. We vary the fault rate from 0.02 to 0.2. To ensure that there are some faulty nodes in faulty networks for small fault rates, we should select enough RBF nodes. Hence, one simple guideline is that the number of RBF nodes should not be much lower than $(1)/(0.02)$. On the other hand, if the number of nodes is too large, we may have a serious overfitting problem. Therefore, the number of RBF nodes should be lower than the number of training samples.

We consider four data sets: the Hermite function [23], the sinc function [32], [34], [35], the nonlinear time series [32], and the astrophysical data [36], [37].

In the Hermite and sinc function examples, we use 37 nodes. We uniformly distribute 37 centers in the input domain. For the nonlinear time series and astrophysical data examples, the input dimension is greater than one and then it is very difficult for us to select RBF centers. Hence, we use the Chen's local regularized assisted orthogonal least squares (LROLS) [32] to select important RBF centers from the training samples. For the nonlinear time–series example, the number of selected nodes is 21. For the astrophysical data example, the number of selected nodes is 53 nodes.

A. Fault Tolerance

Three other techniques are also considered in the simulation. They are the pseudoinverse, Chen's OLS regularizer [27], [32], and Zhou's method [7]. The pseudoinverse is a reference which tests the performance of faulty networks when special care is not considered. In the pseudoinverse, the weight vector is given by $\hat{\mathbf{w}} = (\mathbf{H}_\phi)^{-1}(1)/(N)\sum_{i=1}^{N} y_i\Phi(\mathbf{x}_i)$. The Chen's local OLS regularizer [32] is a typical version of weight decay regularizers. It is an extension of the Bayesian framework on RBF networks [15] and is able to automatically estimate the hyper-parameters (weight decay constants). The Zhou's method [7] considers the following objective function:

$$J(\mathbf{w}) = \alpha E(\mathbf{w}) + \beta\frac{1}{M}\sum_{\mathbf{b}}^{z} \in \mathcal{S}_z E(\mathbf{w}, \mathbf{b}^z) \qquad (31)$$

where $E(\mathbf{w})$ is the MSE of the fault-free network, $E(\mathbf{w}, \mathbf{b}^z)$ is the MSE of a faulty network with fault vector $\mathbf{b}^z$, and $\mathcal{S}_b$ is the set of all possible fault vectors with single-node open fault. The values of α and β are given by $\alpha = (1-p)^M$ and $\beta = (1-p)^{M-1}p$, where α is the probability that all RBF nodes work properly, and β is the probability that an RBF node is faulty.

1) Hermite Function: We consider the Hermite function, shown in Fig. 1(a), given by

$$f(x) = 1.1(1 - x + 2x^2)\exp(-x^2/2) \qquad (32)$$

where $x \in [-10, 10]$. The RBF network model has 37 RBF nodes. The 37 centers are selected as $\{-9, -8.5, -8, \cdots, 8, 8.5, 9\}$. The parameter Δ is set to 0.49. After training, we randomly generate 100 000 faulty networks for each fault rate.

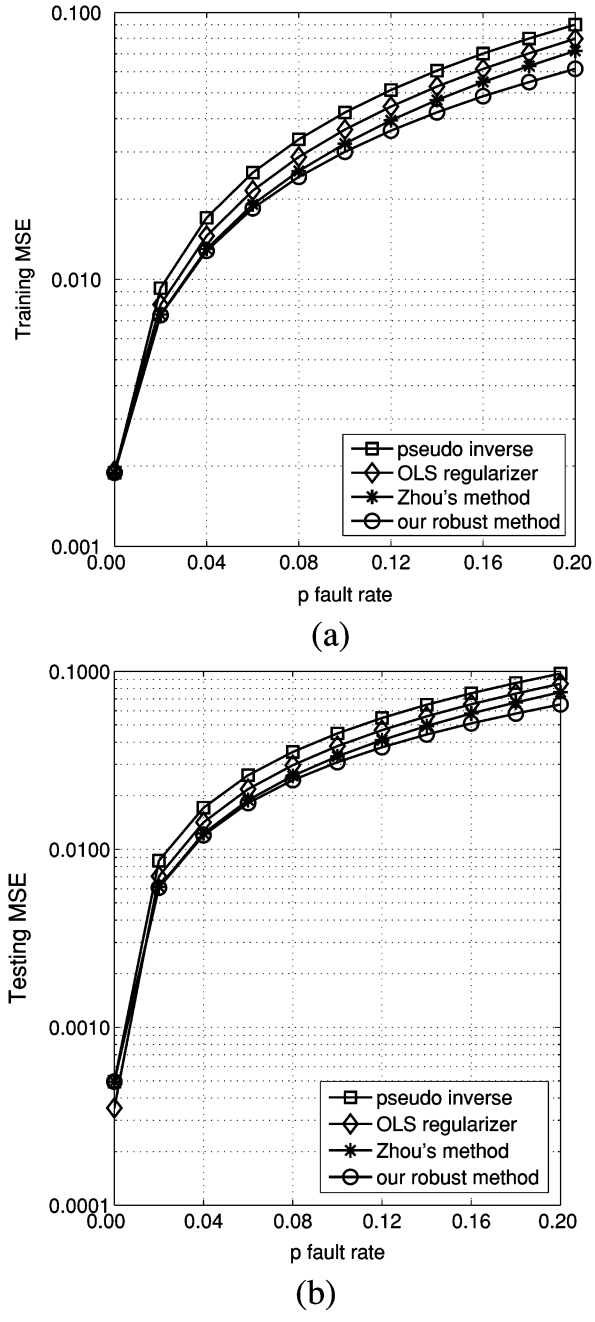

Fig. 3. MSEs of faulty networks for the noisy Hermite function approximation, where $\sigma_e^2 = 0.0025$. (a) Training error. (b) Testing error.

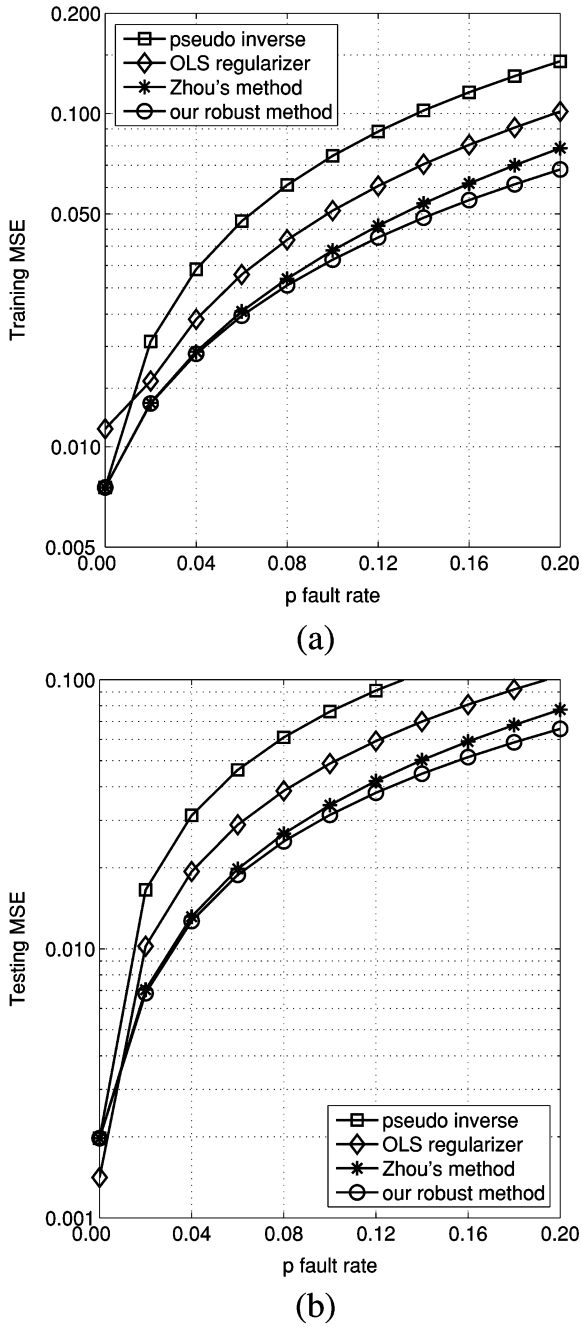

Fig. 4. MSEs of faulty networks for the noisy Hermite function approximation, where $\sigma_e^2 = 0.01$. (a) Training error. (b) Testing error.

Noise-Free Training Data: A training set (200 random samples) is generated based on (32). As there is no output noise in the data, we do not consider the testing samples. The average MSEs for various fault rates are shown in Fig. 1(b). The pseudoinverse and OLS regularizer have the similar performance. This is because the target function is a very smooth function and overtraining by the pseudoinverse is very small. Hence, the OLS regularizer and pseudoinverse produce similar results and have similar performance. The Zhou's method and our robust method can improve fault tolerance. Compared with the Zhou's method, our approach has a much better performance for high fault rates.

Noisy Training Data: In real situation, noise may be added during the data collection process. Therefore, the common practice [29], [26], [27], [34] usually assumes that the measurement output y is contaminated by the measurement noise. That means that the training output is generated by

$$y = 1.1(1 - x + 2x^2)\exp(-x^2/2) + e \tag{33}$$

where the noise term e is a mean zero Gaussian noise with variance σ_e^2. Our goal is to test whether the trained networks can capture the underlying model $1.1(1 - x + 2x^2)\exp(-x^2/2)$. We consider two noise levels ($\sigma_e^2 =$

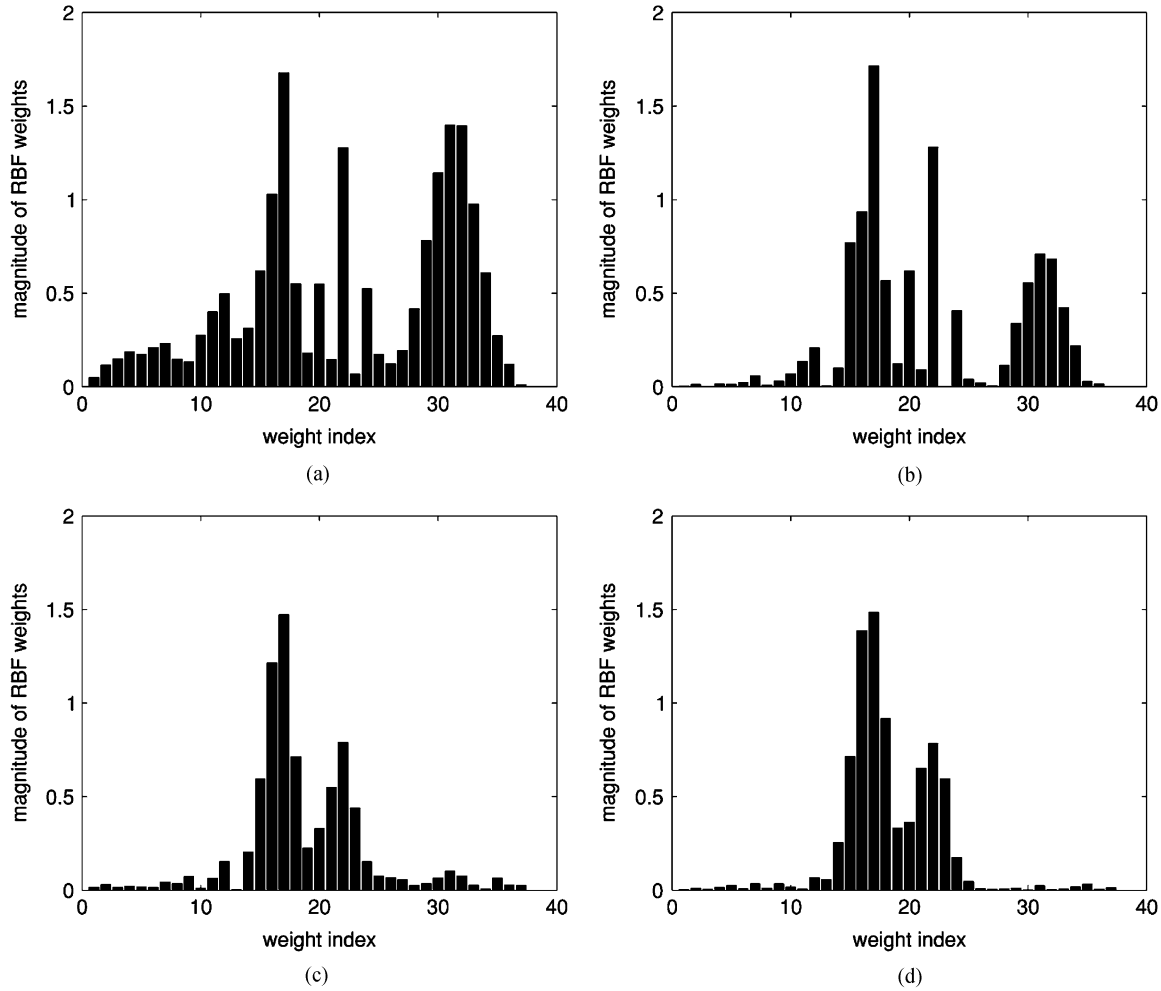

Fig. 5. Magnitude of the trained weights for the noisy Hermite function approximation, where $\sigma_e^2 = 0.01$.

0.0025 and 0.01). For each noise level, a training data set $\mathcal{D}$ (200 samples), shown in Fig. 2, is generated. Also, a noise-free testing data set (200 samples) is generated based on (32).

The average training and testing MSEs are shown in Figs. 3 and 4. For $p = 0$, the OLS regularizer has a larger training error but can achieve a better testing error. This confirms that the OLS regularizer can improve generalization. For the noisy training data, the pseudoinverse gives out a very poor performance, especially, for high output noise level (Fig. 4). This result could be explained by investigating the magnitude of RBF weights (Fig. 5). For noisy conditions, since the model trained by the pseudoinverse is not regularized, the RBF network is overtrained and then the weight magnitude is relatively large. Especially, for the RBF nodes whose centers are around -9 to -5 and 5 to 9, the magnitude of weights obtained from pseudoinverse is much greater than that of weights obtained from our robust method. Hence, if node fault appears in these RBF nodes, the degradation on the training and testing MSEs is very large. As shown in Fig. 5, the OLS regularizer and Zhou's method can suppress the weight magnitude and then improve fault tolerance (shown in Figs. 3 and 4). However, our robust approach gives out the best MSE performance.

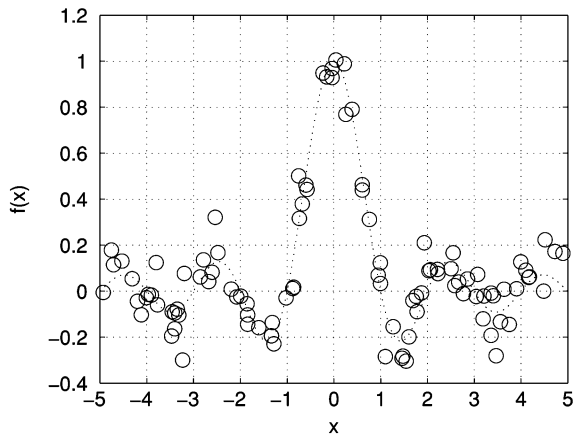

Fig. 6. Training data for the function.

2) Sinc Function: The function is a common benchmark example [32], [34], [35]. The output is generated by

$$y = \mathrm{sinc}(x) + e \tag{34}$$

where the noise term e is a mean zero Gaussian noise with variance $\sigma_e^2 = 0.01$. A training data set (100 samples), shown in

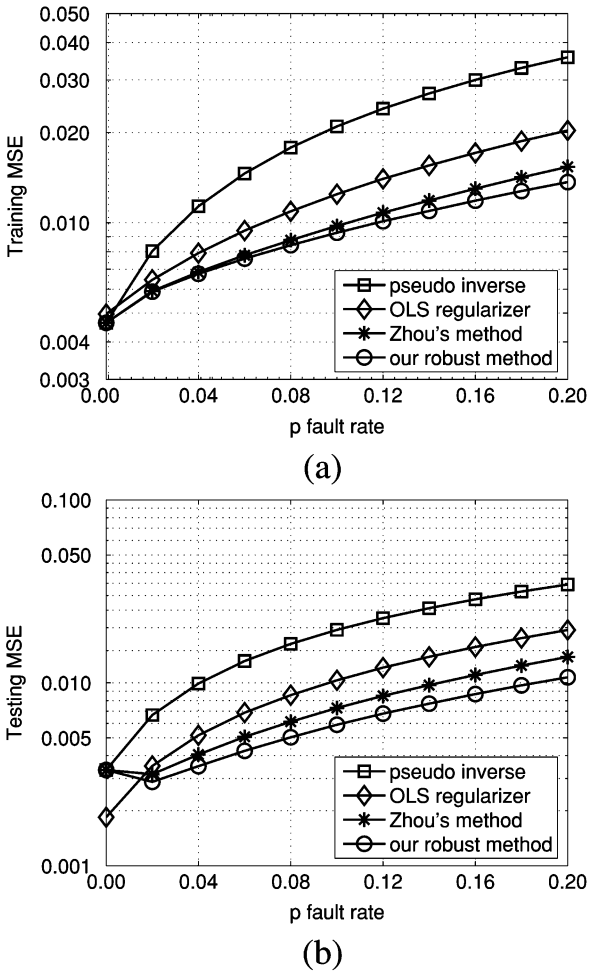

Fig. 7. MSEs of faulty networks for the noisy function approximation, where $\sigma_e^2 = 0.01$. (a) Training error. (b) Testing error.

Fig. 6, is generated. Also, a noise-free testing data set (100 samples) is generated. The RBF network model has 37 RBF nodes. The 37 centers are selected as $\{-4.5, -4.25, \cdots, 4.25, 4.5\}$. The parameter Δ is set to 0.1. For each fault rate, we randomly generate 60 000 faulty networks. The training and testing MSEs are depicted in Fig. 7. The results are quite similar to those of the noisy Hermite function example. That is, for the fault-free case ($p = 0$), the OLS regularizer can improve the generalization. For faulty networks ($p > 0$), among those four algorithms, our algorithm gives out the best fault tolerance.

3) Nonlinear Time–Series Example: We consider the following nonlinear autoregressive time series [32], given by

$$\begin{aligned} y(i) = &(0.8 - 0.5\exp(-y^2(i-1)))\, y(i-1) \\ &- (0.3 + 0.9\exp(-y^2(i-1)))y(i-2) \\ &+ 0.1\sin(\pi y(i-1)) + e(i) \end{aligned} \tag{35}$$

where $e(i)$ is a mean zero Gaussian random variable that drives the series. Its variance is equal to 0.09. 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 $y(i)$ based on the past observations $y(i-1)$ and $y(i-2)$. The prediction is given by

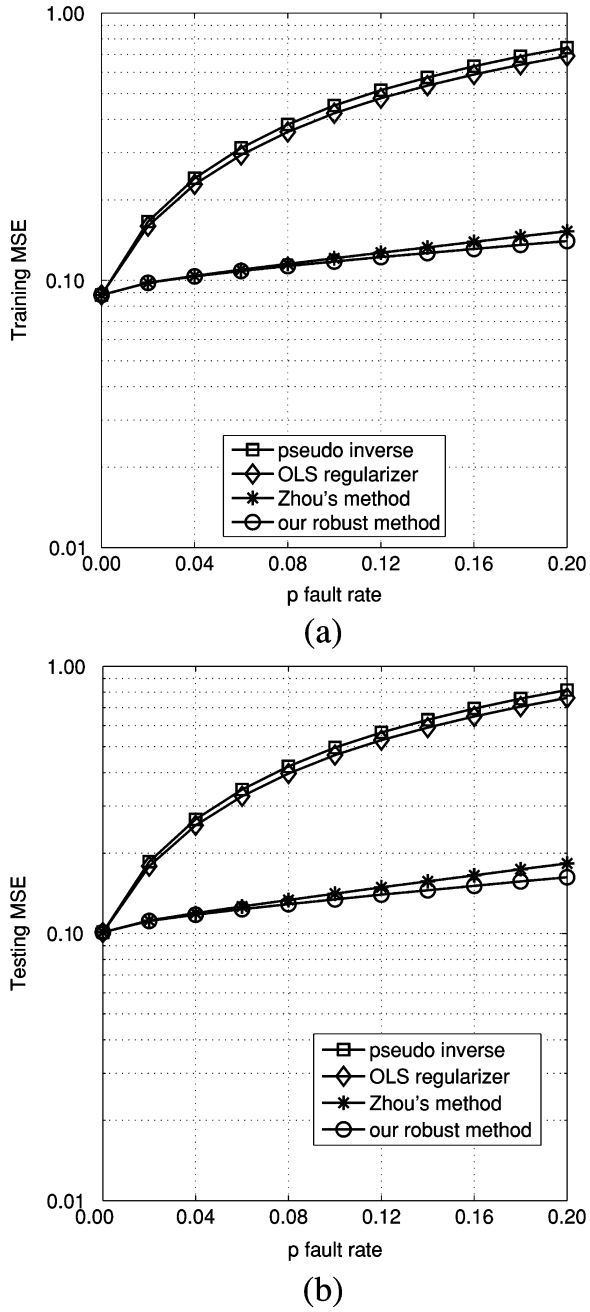

Fig. 8. MSEs of faulty networks for the NAR prediction. (a) Training error. (b) Testing error.

$$\hat{y}(i) = \hat{f}(\mathbf{x}(i), \mathbf{w}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}(i)) \tag{36}$$

where $\mathbf{x}(i) = [y(i-1), y(i-2)]^T$.

For this 2-D input case, the Chen's LROLS is applied to select important RBF centers (basis functions). The initial structure of the model consists of 500 basis functions

$$\hat{f}(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{500} w_j \phi_j(\mathbf{x}) \tag{37}$$

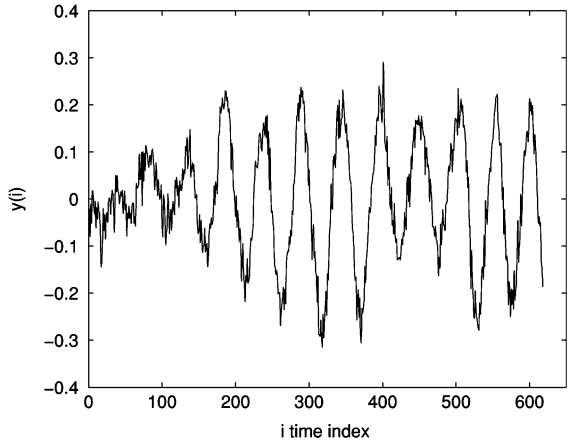

Fig. 9. Astrophysical data.

where $\phi_j(\mathbf{x}) = \exp(-(||\mathbf{x}-\mathbf{c}_i||^2)/(\Delta))$. The centers are assigned as $\mathbf{c}_1 = [y(0), y(1)]^T, \ldots, \mathbf{c}_{500} = [y(499), y(500)]^T$, and the width of the centers Δ is set to 0.81. Chen in [32] proposed a method to rank the importance of RBF centers. After the ranking process, we use the change of the regularized error reduction ratio (RERR) [32] as the termination condition for the selection process. If the change of RERR is smaller than 0.0001, we stop the selection. The number of selected RBF nodes is 21. Let $\hat{f}_{\text{Chen}}$ be the RBF network attained

$$\hat{f}_{\text{Chen}}(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{21} w_{\pi_j}^{\text{CHEN}} \phi_{\pi_i}(\mathbf{x}) \tag{38}$$

where $\pi_1, \ldots, \pi_{21}$ are the indices of the RBF nodes being selected.

After the selection process, four RBF networks are trained with the four algorithms: pseudoinverse, Chen's OLS regularizer, Zhou's method, and our robust method. We then randomly generate 60 000 faulty networks for each fault rate. The performances of the trained RBF networks are depicted in Fig. 8. Similar to previous examples, the pseudoinverse gives out a very poor performance. The OLS regularizer can improve the performance a bit. The Zhou's method and our robust method can greatly improve fault tolerance. Among those four methods, our robust method gives out the best fault tolerance.

4) Astrophysical Data: We consider the time–series prediction of a real data set. The series is the time variation of the intensity of the white dwarf star PG1159-035 during March 1989 [36], [37]. The data samples are noisy and nonlinear in nature.[1] Part one of this data set, shown in Fig. 9, is selected. There are 618 data samples.

Our task is to train RBF networks to predict the current value $y(i)$ based on five past values $y(i-1), \ldots, y(i-5)$. That means that the RBF model has five inputs and one output. Hence, there are 613 input–output pairs. The first 309 pairs are the training data and the remaining pairs are the testing data. In this example, we also use the LROLS algorithm to select important RBF centers from the training set. After selection, we have 53 important RBF nodes. RBF networks are then trained with the four algorithms. For each case, we randomly generate 60 000 faulty networks for each fault rate. The performance of the trained RBF networks is depicted in Fig. 10. The results are also similar to those of the previous examples. Among those methods, our robust method gives out the best performance.

[1]It can be downloaded from http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html

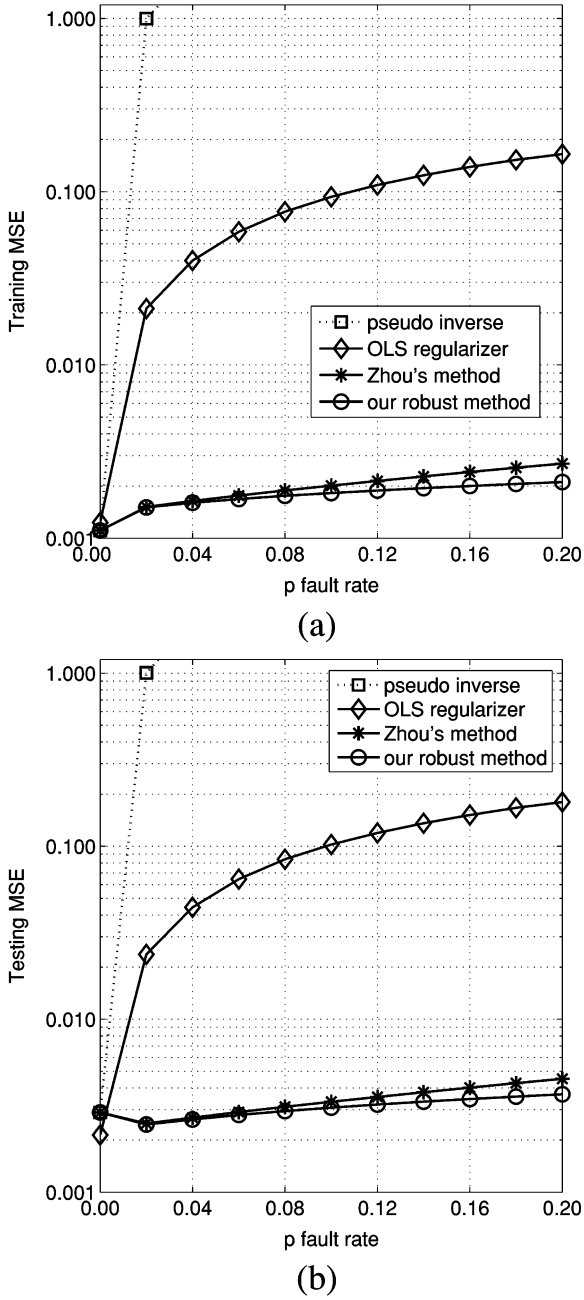

Fig. 10. MSEs of faulty networks for the astrophysical data prediction. (a) Training error. (b) Testing error.

B. Further Enhancement: Triple Modular Redundancy

To further enhance fault tolerance, we investigate the replication of RBF nodes [5], [8]. We adopt the idea of triple modular redundancy (TMR). The examples considered are the NAR prediction and the astrophysical data prediction. The setting of the RBF networks used here is the same as that used in Sections IV-A3 and IV-A4. After training, for each trained RBF network, we duplicate two more identically trained RBF

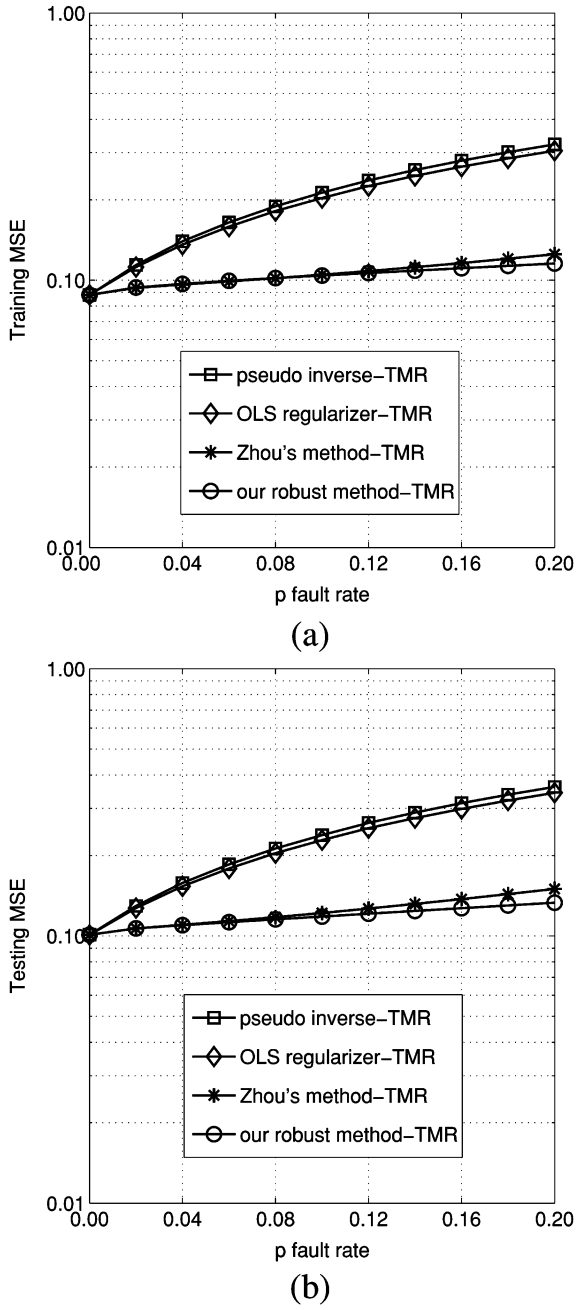

Fig. 11. MSEs of faulty networks with TMR for the NAR prediction. (a) Training error. (b) Testing error.

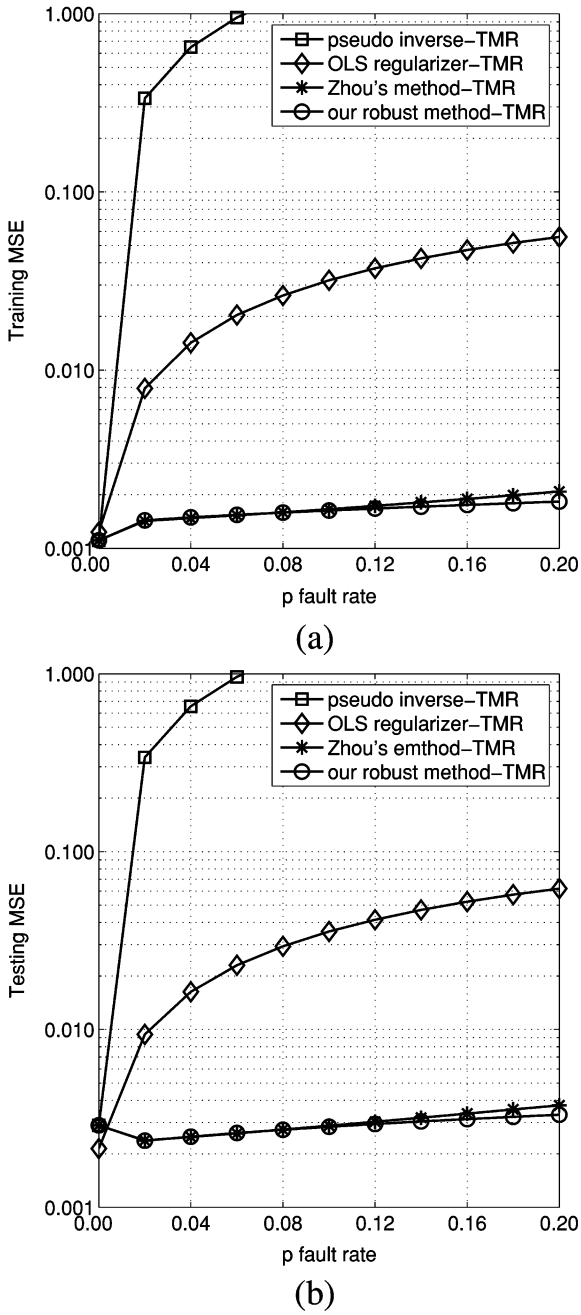

Fig. 12. MSEs of faulty networks with TMR for the astrophysical data prediction. (a) Training error. (b) Testing error.

networks. The three network outputs are added together with the output divided by three. The performances are depicted in Figs. 11 and 12, respectively. Compared with the non-TMR approach (shown in Figs. 8 and 10), the TMR approach can further improve fault tolerance. Our robust method with TMR gives out the best performance.

C. Incorrect Training Fault Rates

With our robust learning, we can find the optimal RBF weight (with respect to the Kullback–Leibler divergence) if we know the exact fault rate, i.e., fault statistics. In some practical situations, there may be some differences between the training fault rate and the exact fault rate. Hence, it is interesting to study the degradation due to the deviation.

The examples we considered here are the noisy function with $\sigma_e = 0.01$ and the astrophysical data prediction. The setting of the RBF model used here is the same as that used in Sections IV-A2 and IV–A4. In the simulation, for our robust method and the Zhou's method, we use two training fault rates $p' = 0.02$ and 0.1 to train RBF networks. The MSEs of the RBF networks trained with training fault rates are then measured on various true fault rates. For each RBF network, we randomly generate 60 000 faulty networks for each true fault rate. The MSE performances are depicted in Figs. 13 and 14.

For the training error, the degradation of our robust method due to using an incorrect training fault rate is not so large. For

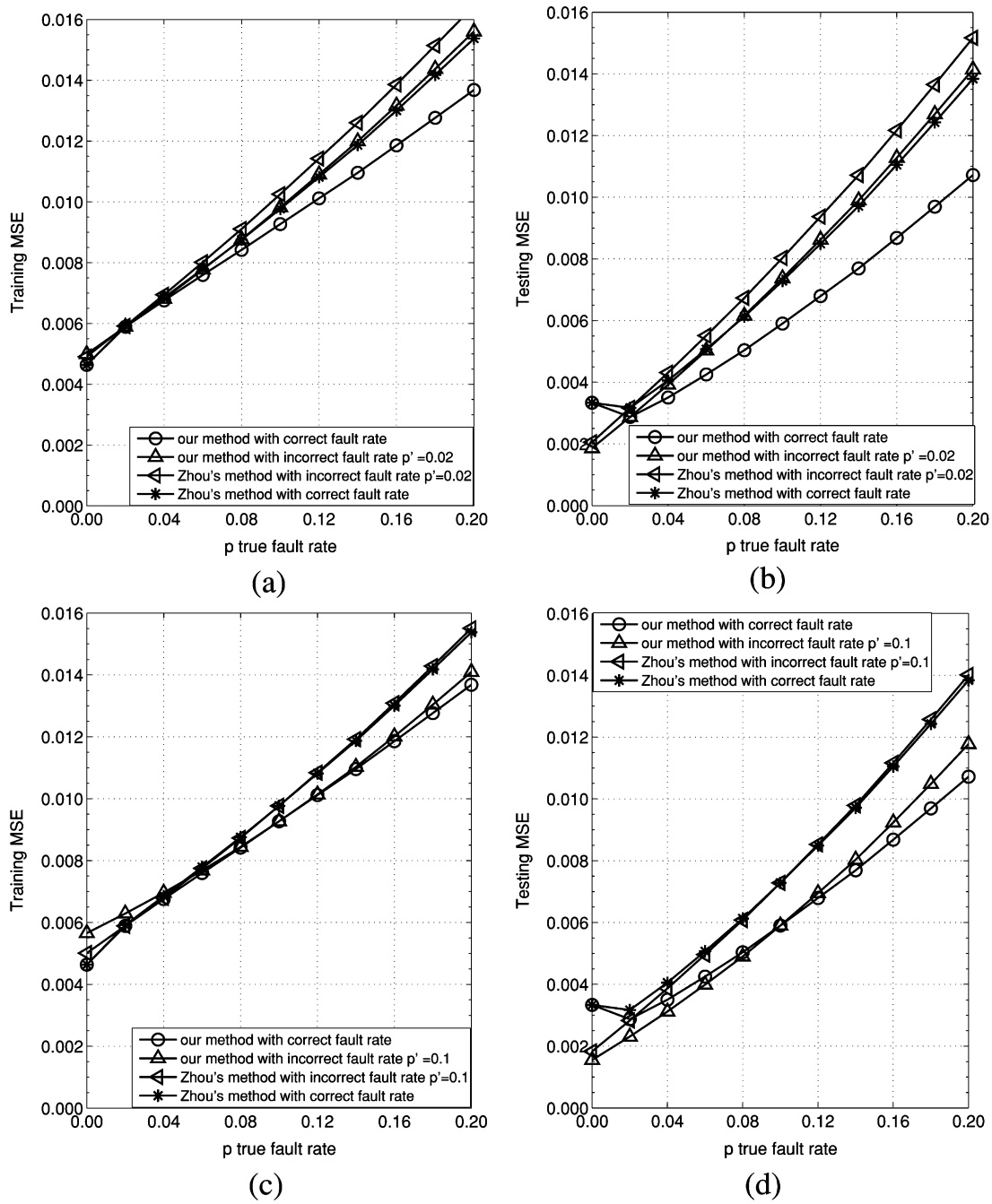

Fig. 13. MSEs of faulty networks with incorrect guess fault rate for the noisy function example. (a) and (b) MSEs for the guess fault rate $p' = 0.02$. (c) and (d) MSEs for the guess fault rate $p' = 0.1$.

a small training fault rate $p' = 0.02$, the degradation becomes larger as the true fault rate increases. For a large training fault rate $p' = 0.1$, the degradation is large when the true fault rate is small. The interesting result is that even though we use a small training fault rate we still obtain certain improvement on fault tolerance for large true fault rates. For example, when the incorrect fault rate p' is equal to 0.02, for large true fault rates, the performance of our approach is still much better than that of the pseudoinverse and OLS regularizers. Compared with the Zhou's method with incorrect fault rates, our robust method with incorrect fault rates gives out a better MSE performance.

For the testing error, the result is quite similar to that of the training error. However, there is an interesting phenomenon. That is, as the true fault increases, the testing error first decreases and then increases. That means that the proposed regularizer has a certain generalization ability for fault-free networks or faulty networks with very small true fault rates. When the true fault rate is small $(p < 0.02)$, the effect of the multinode open fault is not so large. As the proposed method can control the weight magnitude, it can improve the generalization. However, as the true rate further increase, the effect of the multinode open fault becomes larger and the testing error starts to increase.

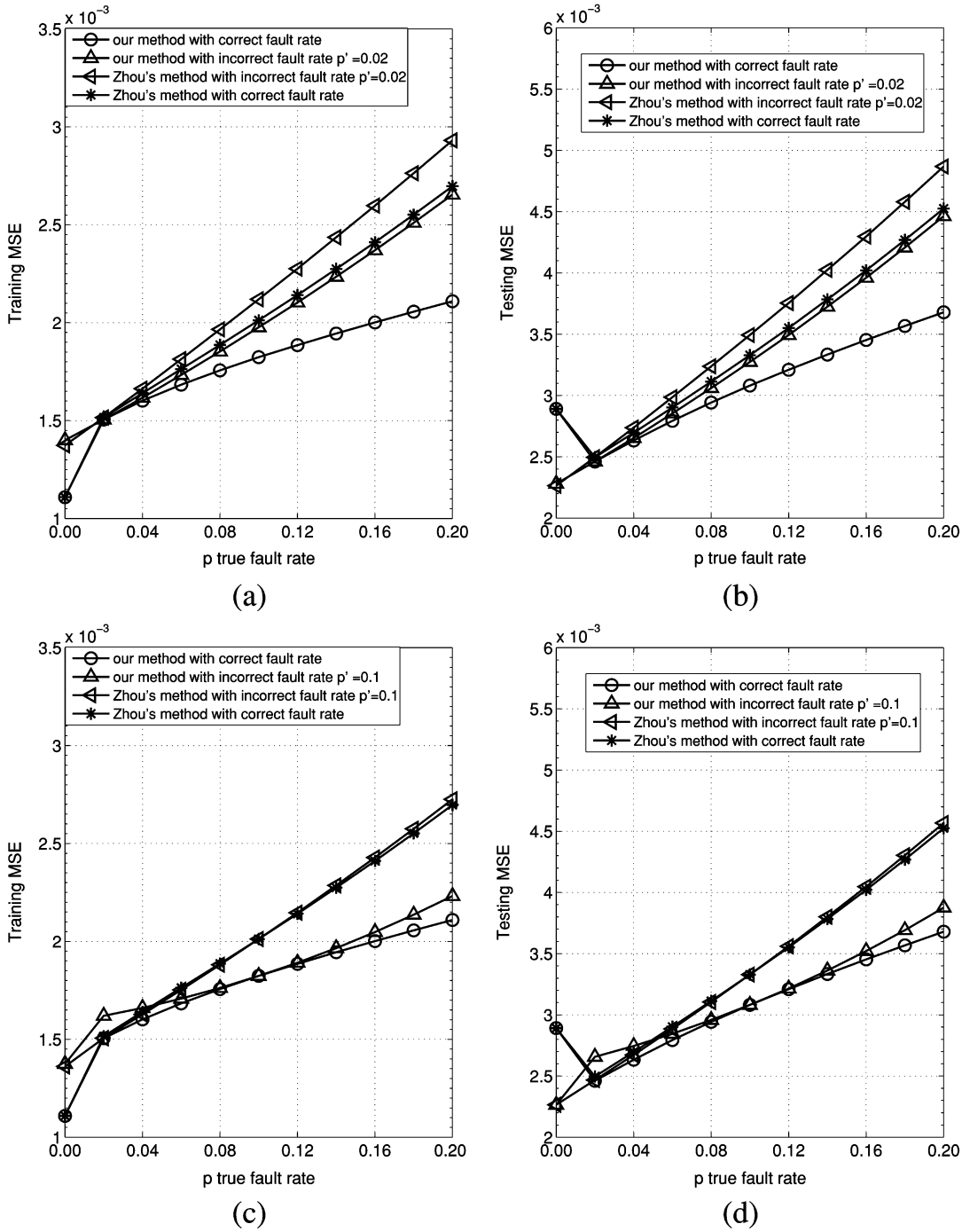

Fig. 14. MSEs of faulty networks with incorrect guess fault rate for the astrophysical data prediction. (a) and (b) MSEs for the guess fault rate $p' = 0.02$. (c) and (d) MSEs for the guess fault rate $p' = 0.1$.

D. Generalization Ability

From Section IV-C, we observe that the proposed robust method could improve certain generalization ability. This section investigates the generalization ability of the proposed robust method under fault-free condition. We use the proposed robust method with various training fault rates to train a number of RBF networks. Afterwards, we test the generalization ability of the trained RBF networks under "fault-free condition." The examples we considered here are the noisy function with $\sigma_e = 0.01$ and the astrophysical data prediction.

The results are depicted in Fig. 15, where we can observe that with an appropriate value of p our algorithm can also improve generalization. For the function example, for some training fault rates, our method is even better than the OLS regularizer. This does not mean that the OLS regularizer is not good for generalization. This is because the OLS regularizer can automatically select the appropriate regularization parameters while our robust method cannot. This means that the limitation of our algorithm for improving the generalization is that we need to have an additional validation set for selecting appropriate training value of p.

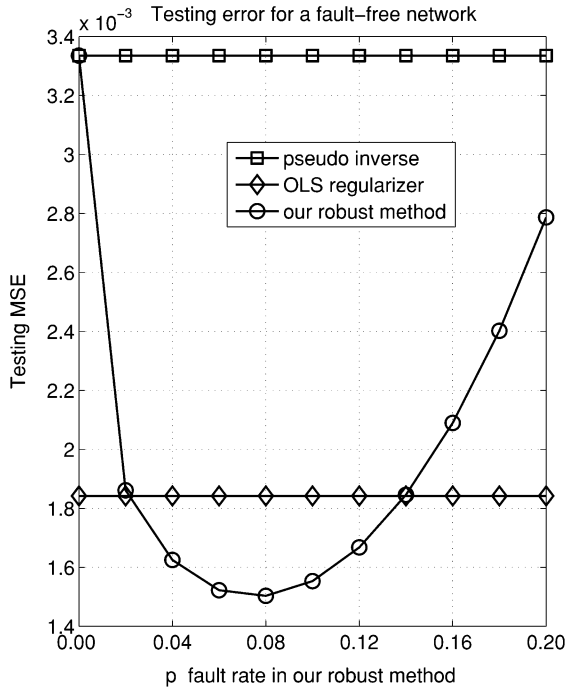

Fig. 15. Generalization ability of various algorithms under the fault-free condition. Networks are trained with the training fault rate p. (a) Noisy function. (b) Astrophysical data prediction.

V. Conclusion

This paper addresses the fault tolerance of RBF networks where all hidden nodes have the same fault rate and their fault probabilities are independent. Assuming that there is a Gaussian distributed noise in the output data, we have derived an objective function for robustly training an RBF network based on the Kullback–Leibler divergence. We also find that for a fault-tolerance regularizer some eigenvalues of the regularization matrix should be negative. For the Tipping's regularizer and the OLS regularizer, the regularization matrices are positive or semipositive definite. Hence, they cannot efficiently handle the multinode open fault. Various simulation studies confirm that in terms of fault tolerance our approach is better than other methods being tested. Also, the proposed robust method can improve fault tolerance even when an incorrect training fault rate is used. Besides, we test the generalization ability of the proposed method.

Although our discussion focuses on the RBF networks, one can follow our derivation to handle networks with other activations, such as sigmoid and hyperbolic tangent. Of course, in such extended cases, the objective function is similar to that of the RBF networks but the learning algorithm may not be similar. Hence, one future direction is to generalize our approach to handle an node open fault or a weight open fault for multilayer networks with other activation functions.

Our robust method is designed for open node fault. It may not be able to handle other fault models, such as "stuck-at-maximum" and "stuck-at-minimum." Hence, another future direction is to generalize our method to handle other fault models.

Appendix

The Tipping local regularizer [33] is given by

$$\mathbf{w}^T \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \lambda_M \end{bmatrix} \mathbf{w} \tag{39}$$

where $\lambda_j > 0$. Each weight is controlled by an independent hyperparameter that can be obtained by the evidence maximization [30]. In accordance with the reestimation technique shown in [33], the parameter λ_j satisfies the condition

$$\lambda_j = \langle w_j^2 \rangle^{-1}$$

where the expectation is taken over the posterior probability of w_j's given training data set and the regularizer. Obviously, all eigenvalues of the regularization matrix are nonnegative. Therefore, it is able to penalize the weight magnitude but it is unlikely to train an RBF network with optimal tolerance for multinode open fault.

Inspired by Tipping's local regularization, Chen in [32] extended the idea of OLS by introducing a regularizer term with M hyperparameters. The objective function to be minimized is given by

$$\sum_{i=1}^{N} \|y_i - \Phi^T(\mathbf{x})\,\mathbf{w}\|^2 + \mathbf{w}^T \mathbf{A}^T \mathbf{Z}\, \mathbf{A}\, \mathbf{w}. \tag{40}$$

The matrix $\mathbf{A}$ is an upper triangular matrix satisfying the following condition:

$$\begin{bmatrix} \Phi^T(\mathbf{x}_1) \\ \Phi^T(\mathbf{x}_2) \\ \cdots \\ \Phi^T(\mathbf{x}_N) \end{bmatrix} = \mathbf{Q} \underbrace{\begin{bmatrix} 1 & a_{12} & \cdots & a_{1M} \\ 0 & 1 & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}_{\mathbf{A}} \tag{41}$$

where $\mathbf{Q} \in \Re^{N\times M}$ and $\mathbf{A} \in R^{M\times M}$ are obtained by a modified Gram–Schmitt procedure called the OLS method. It should be noticed that column vectors $\mathbf{q}_j$'s form an orthogonal basis. The matrix $\mathbf{Z}$ is a diagonal matrix

$$\mathbf{Z} = \mathbf{diag}\{z_1, z_2, \ldots, z_M\}. \tag{42}$$

Elements $z_1, z_2, \ldots, z_M$ in $\mathbf{Z}$ are the hyperparameters satisfying the following equalities:

$$z_j = \frac{\gamma_j \sum_{i=1}^{N}(y_i - \Phi^T(\mathbf{x}_i)\mathbf{w})^2}{\left(N - \sum_{j=1}^{M}\gamma_j\right)[\mathbf{A}\,\mathbf{w}]_j^2} \tag{43}$$

$$\gamma_j = \frac{\mathbf{q}_j^T\mathbf{q}_j}{z_j + \mathbf{q}_j^T\mathbf{q}_j} \tag{44}$$

where $[\mathbf{Aw}]_j$ is the jth element of the vector $\mathbf{Aw}$. It should be noticed that $1 \geq \gamma_j \geq 0$ and $M < N$. Hence, all eigenvalues of the matrix $\mathbf{A}^T\mathbf{ZA}$ are nonnegative. Similar to the Tipping's local regularizer, the Chen's regularizer is able to penalize the weight magnitude but it is unlikely to train an RBF network to have the optimal fault tolerance.

REFERENCES

[1] G. R. Bolt, "Fault models for artificial neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Singapore, 1991, pp. 1373–1378.

[2] G. R. Bolt, J. Austin, and G. Morgan, "Fault tolerant multi-layer perceptron networks," Tech. Rep. YCS-92-180, 1992.

[3] A. F. Murray and P. J. Edwards, "Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training," *IEEE Trans. Neural Netw.*, vol. 5, no. 5, pp. 792–802, Sep. 1994.

[4] C. T. Chiu, K. Mehrotra, C. K. Mohan, and S. Ranka, "Modifying training algorithms for improved fault tolerance," in *Proc. Int. Conf. Neural Netw.*, 1994, vol. 4, pp. 333–338.

[5] 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.

[6] Z. H. Zhou, S. F. Chen, and Z. Q. Chen, "Improving tolerance of neural networks against multi-node open fault," in *Proc. Int. Joint Conf. Neural Netw.*, 2001, vol. 3, pp. 1687–1692.

[7] Z. H. Zhou and S. F. Chen, "Evolving fault-tolerant neural networks," *Neural Comput. Appl.*, vol. 11, pp. 156–160, 2003.

[8] 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.

[9] C. H. Sequin and R. D. Clay, "Fault tolerance in feedforward artificial neural networks," *Neural Netw.*, vol. 4, pp. 111–141, 1990.

[10] S. Cavalieri and O. Mirabella, "A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks," *Neural Netw.*, vol. 12, pp. 91–106, 1999.

[11] C. Neti, M. H. Schneider, and E. D. Young, "Maximally fault tolerance neural networks," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 14–23, Jan. 1992.

[12] D. Deodhare D, M. Vidyasagar, and S. S. Keerth, "Sythesis of fault-tolerant feedforward neural networks using minimax optimization," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 891–900, Sep. 1998.

[13] D. Simon and H. El-Sherief, "Fault-tolerance training for optimal interpolative nets," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1531–1535, Nov. 1995.

[14] D. S. Phatak and E. Tchernev, "Synthesis of fault tolerant neural networks," *Proc. Int. Joint Conf. Neural Netw.*, pp. 1475–1480, 2002.

[15] O. J. L. Mark, "Regularization in the selection of radial basis function centers," *Neural Comput.*, vol. 7, pp. 606–623, 1995.

[16] X. Hong, "A fast identification algorithm for box-cox transformation based radial basis function neural network," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 1064–1069, Jul. 2006.

[17] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[18] C. Constantinopoulos and A. Likas, "An incremental training method for the probabilistic RBF network," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 966–974, Jul. 2006.

[19] B. S. Lin, F. C. Chong, and F. Lai, "Higher-order-statistics-based radial basis function networks for signal enhancement," *IEEE Trans. Neural Netw.*, vol. 18, no. 3, pp. 823–832, May 2007.

[20] J. E. Moddy, "Note on generalization, regularization, and architecture selection in nonlinear learning systems," in *Proc. 1st IEEE-SP Workshop Neural Netw. Signal Process.*, 1991, pp. 1–11.

[21] D. S. Phatak, "Relationship between fault tolerance, generalization and the Vapnik-Cervonenkis (VC) dimension of feedforward ANNs," *Proc. Int. Joint Conf. Neural Netw.*, vol. 1, pp. 705–709, 1999.

[22] N. Kamiura, Y. Taniguchi, T. Isokawa, and N. Matsui, "An improvement in weight-fault tolerance of feedforward neural networks," in *Proc. Appl. Telecommun. Symp.*, 2001, pp. 359–364.

[23] C. S. Leung, A. C. Tsoi, and L. W. 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.

[24] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Comput.*, vol. 7, no. 1, pp. 108–116, 1995.

[25] Y. Xu, K. W. Wong, and C. S. Leung, "Generalized RLS approach to the training of neural networks," *IEEE Trans. Neural Netw.*, vol. 17, no. 1, pp. 19–34, Jan. 2005.

[26] S. I. 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.

[27] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, "Sparse modelling using orthogonal forward regression with press statistic and regularization," *IEEE Trans. Syst. Man. Cybern. B, Cybern.*, vol. 34, no. 2, pp. 898–911, Apr. 2004.

[28] S. Kullback, *Information Theory and Statistics*. New York: Wiley, 1959.

[29] J. Moody, "A smoothing regularizer for feedforward and recurrent neural networks," *Neural Comput.*, vol. 8, pp. 461–489, 1996.

[30] D. J. C. Mackay, "A practical Bayesian framework for backprop networks," *Neural Comput.*, vol. 4, pp. 448–472, 1992.

[31] 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.

[32] S. Chen, "Local regularization assisted orthogonal least squares regression," *Neurocomputing*, pp. 559–585, 2006.

[33] M. E. Tipping, "The relevance vector machine," in *Advances in Neural Information Processing Systems 12*. Cambridge, MA: MIT Press, 2000, pp. 652–658.

[34] V. Cherkassky and Y. Ma, "Multiple model regression estimation," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 785–798, Jul. 2005.

[35] V. Vapnik, S. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems 9*. Cambridge, MA: MIT Press, 1997, pp. 281–287.

[36] S. Singh, "Noise impact on time-series forecasting using an intelligent pattern matching technique," *Pattern Recognit.*, vol. 32, pp. 1389–1398, 1999.

[37] E. W. M. Lee, C. P. Lim, R. K. K. Yuen, and S. M. Lo, "A hybrid neural network model for noisy data regression," *IEEE Trans. Syst. Man Cybern. B., Cybern.*, vol. 34, no. 2, pp. 951–960, Apr. 2004.

Chi-Sing Leung (M'04) received the B.Sc. 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, Hong Kong, in 1989, 1991, and 1995, respectively.

Currently, he is an Associate Professor at the Department of Electronic Engineering, City University of Hong Kong. In 2007, he gave a one-hour lecture, "Is there anything comparable to spherical harmonics but simpler?" at the Game Developers Conference in San Francisco, CA. His 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.

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

From 1998 to 2004, he had been teaching at 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, he moved to Taiwan and begun teaching at the Chung Shan Medical University. Since August 2007, he has been 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*.