On objective function, regularizer and prediction error of a learning algorithm for dealing with multiplicative weight noise

John SUM[1,2], Chi-sing LEUNG[2], Kevin HO[3]

[1] Department of Information Management,

Chung Shan Medical University, Taichung, Taiwan

[2] Department of Electronic Engineering,

City University of Hong Kong, Hong Kong

[3]Department of Computer Science and Communication Engineering

Providence University, Sha-Lu, Taiwan ROC.

Abstract

In this paper, an objective function for training a functional link network to tolerate multiplicative weight noise is presented. This objective function is similar to that of other regularizer-based objective function, which is composed of the mean square training error term and a regularizer term. Based on this objective function, a simple learning algorithm of attaining a fault tolerant functional link network can be obtained. In sequel, the mean prediction error of the trained network in response to multiplicative weight noise effect is analyzed. A formula resembling the Akakie Information Criteria is obatined. Simulatied experiments on two artificial datasets and one real-world application are made to verify the theoretical results.

Keywords: Fault tolerance, Mean prediction errors, Multiplicative noise, functional link network.

1 Introduction

In neural networks, network faults can be exhibited in many different forms, such as node fault and weight fault. One kind of weight faults is called multiplicative weight noise, which is due to the finite precision representation of a number, [1], [2]. For instance, in a digital implementation using a low precision floating point format (say a 16-bit half-float) to represent a trained weight, the magnitude of its truncation error is proportional to that of the trained weight [3]. In this regard, the truncation error caused by a finite precision representation is in essence an inherent source of multiplicative weight noise [4], [5], [6], [7], [8]. In contrast to additive weight noise in which the noise factor is independent of the weight magnitude, the resultant effect of a multiplicative weight noise could be so huge if the corresponding trained weight is of large magnitude. Without properly handling its effect, drastic performance degradation can be resulted in the performance[3]. For more than a decade, many studies on multiplicative weight noise have thus been working in order to alleviate such effect. Although many of them have succeeded in improving the fault tolerant abililty of a neural network, not much theoretical works on the objective function for attaining a weight noise tolerant funtional link network and its corresponding prediction error have been developed.

For the Madaline model, Stevenson *et al* [9] and Piche [10] gave comprehensive analysis on the effect multiplicative weight noise. For the multilayer perceptron model, Choi *et at* [11] applied the statistical approach to derive various *output sensitivity measures*. Townsend *et al* [12] derived the output sensitivity of an radial basis function (RBF) network suffered from perturbations in centers and output weights.

As the output sensitivity is only an indirect view point to understand the effect of multiplicative weight noise, the actual effect on the performance cannot be identified easily from the output sensitivity. A more practical approach is to study the effect on the error sensitivity measure. Catala *et al* [13] proposed a fault tolerance model and studied the performance degradation of an RBF network if its RBF centers, widths and weights are corrupted by multiplicative noise. Bernier *et al* extended the Choi's results [11] and derived the *error sensitivity measure* for multilayer perceptrons [7] and RBF networks [8]. Similarly, Fontenla-Romero *et al*[14] derived the *error sensitivity measure* for functional

link nets.

While many works have been carried out for the effect of multiplicative weight noise, various training methods aiming at reducing the effect of multiplicative noise have also been developed. Since the effect of a multiplicative weight noise is proportional to the magnitude of a trained weight, one intuitive approach is to control the magnitude of the network weights during training. There are several some heuristics. For example, Cavalieri *et al* [15] have proposed a modified backpropagation learning for multilayer perceptrons. In their algorithm, whenever the magnitude of a weight has reached a predefined upper limit, the weight will not be updated unless the update can bring its magnitude down. On the other hand, considering that the noise effect can largely be reduced at the output node if all the weight values are equal, Simon [16] suggested a distributed fault tolerance learning approach, in which the training error is minimized subjected to an equality constraint on weight magnitude. Extended from their previous work in [13], Parra and Catala [17] demonstrated how a fault tolerant RBF network can be obtained by using a weight decay regularizer [18], [19].

From the training error sensitivity point of view [11], Bernier *et al* in [4] derived a mean square error sensitivity term for a neural network that is corrupted by weight perturbation. Later, they proposed to use this sensitivity term as an explicit regularizer [5] [7] for training a multilayer perceptron to tolerate multiplicative weight noise. In their formulation, the objective function is defined as

$$\text{Mean Square Training Error} + \gamma \text{Mean Square Error Sensitivity}.$$

However, the actual selection rule of γ in the regularization term has not been addressed theoretically. Bernier *et al* only proposed to use a validation set to determine its value.

Following the same approach as work done by Bernier *et al* in [5] [7] and Parra & Catala in [17], an objective function and the learning algrithm for training a functional link network to tolerate muliplicative weight noise will be derived. Then, the property of the corresponding regularizer is discussed and the mean prediction error is analyzed. The contributions in the paper are listed as follows :

- An objective function and the corresponding learning algorithm are derived.

- It is shown that under certain speical condition, the regularizer can be reduced to the weight decay term.
- A formula resembling Akakie Information Criteria has been derived to estimate the mean prediction error.

In the next section, the background knowledge about the network model, the weight noise model and the regularization technique will be presented. The objective function for training a functional link network [20] to tolerate multiplicative weight noise is derived in Section 3. By minimizing the objective function, a training algorithm is also presented. Based on the objective function being derived, a formula for estimating the mean prediction error of the trained network is obtained in Section 4. Simulations results verifying the theoretical works are elucidated in Section 5. In these simulations, we consider the functional link network as a radial basis function (RBF) network [21], [22] Finally, the conclusion is presented in Section 6.

2 Background

2.1 Data Model

Throughout the paper, we are given a training dataset $\mathcal{D}_t$,

$$\mathcal{D}_t = \left\{(\mathbf{x}_j, y_j) : \mathbf{x}_j \in \Re^K, y_j \in \Re, j = 1, \cdots, N.\right\} \ ,$$

where $\mathbf{x}_j$ and y_j are the input and output samples of an unknown system, respectively. We assume that the dataset $\mathcal{D}_t$ is generated by a stochastic system[23][21], given by

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

where $f(\cdot)$ is the unknown system mapping; and e_j's are the random measurement noise. The noise e_i's are identical independent zero-mean Gaussian random variables with variance equal to S_e.

2.2 Network Model

In the functional link net approach [24], [25], we would like to approximate the mapping $f(\cdot)$ by a weighted sum of basis functions, given by

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}), \tag{2}$$

where $\mathbf{w} = [w_1, \cdots, w_M]^T$ is the weight vector, and $\phi_j(\cdot)$'s are the pre-defined basis functions (mappings from the K-dimensional space to real number). In the vector-matrix notation, (2) can be written as

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

where $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \cdots, \phi_M(\mathbf{x})]^T$. Our learning task is to find out a weight vector that best fits the observations. That means, we would like to find a weight vector that minimizes the following objective function:

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

Throughout the paper, we assume the Gram matrix

$$\mathbf{G} = \int \Phi(\mathbf{x}) \Phi^T(\mathbf{x}) \mathcal{P}(\mathbf{x}) d\mathbf{x} \approx \frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j) \Phi^T(\mathbf{x}_j) \tag{5}$$

is not near singular, where $\mathcal{P}(\mathbf{x})$ is the density function of $\mathbf{x}$.

2.3 Multiplicative Weight Noise

An implementation of a weight vector $\mathbf{w}$ is denoted by $\tilde{\mathbf{w}}$. In multiplicative weight noise, each implemented weight deviates from its nominal value by a random percent, i.e.,

$$\tilde{w}_i = w_i + b_i\, w_i \;\; \forall \;\; i = 1, 2, \cdots, M \,, \tag{6}$$

where b_i's are identical independent mean zero random variables with variance S_b. Denote $\mathbf{b} = [b_1, \cdots, b_M]^T$. The density function of b_i's are symmetrical.

2.4 Regularization

Adding regularizer [18], [26], [27], [28] is a common technique in neural network learning. A regularized objective function is usually defined as

$$\mathcal{J}(\mathbf{x}, \mathbf{R}) = \frac{1}{N} \sum_{j=1}^{N} (y_j - \Phi^T(\mathbf{x})\, \mathbf{w})^2 + \gamma \mathbf{w}^T \mathbf{R} \mathbf{w}, \tag{7}$$

where $\gamma \mathbf{w}^T \mathbf{R} \mathbf{w}$ is the regularizer term, γ is the weighting factor, and $\mathbf{R}$, the so-called regularization matrix, is positive definite. In the standard weight decay [18], [27], the matrix $\mathbf{R}$ is an identity matrix.

In [5] and [7], Bernier *et al* use the regularization technique to improve the performance of a multilayer perceptron affected the multiplicative weight noise. In their formulation, they defined that the regularization term is related to the error sensitivity of weights (Equation (9) in [5]). In case of functional link net, the explicit regularization term is given by $\gamma \mathbf{w}^T \mathbf{R}_{exp} \mathbf{w}$. The matrix $\mathbf{R}_{exp}$ is defined as

$$\mathbf{R}_{exp} = \frac{S_b}{N} \begin{bmatrix} \sum_{j=1}^{N} \phi_1^2(\mathbf{x}_j) & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{N} \phi_2^2(\mathbf{x}_j) & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \sum_{j=1}^{N} \phi_M^2(\mathbf{x}_j) \end{bmatrix}. \tag{8}$$

In [5] and [7], the physical meaning of their objective function was not discussed. Besides, the actual selection rule on γ is not addressed. They only proposed to use a validation set to determine the value of the weighting factor γ.

Suppose the functional-link basis functions are RBF basis functions[21], [22] with the same width. If the input $\mathbf{x}$ is uniformly distributed, i.e. $\mathcal{P}(\mathbf{x})$ is a constant factor depended on the size of the space of $\mathbf{x}$. The RBF width is small in compared with the input space and the centers are located not at the boundary of the input space.

$$\begin{aligned} \frac{1}{N} \sum_{j}^{N} \phi_i^2(\mathbf{x}_j) &\approx \int \phi_i^2(\mathbf{x}) \mathcal{P}(\mathbf{x}) d\mathbf{x} \\ &= \rho, \end{aligned}$$

which is a constant factor. Thus, the explicit regularizer will become a factor depended on the noise variance S_b and ρ, i.e.

$$\mathbf{R}_{exp} = S_b \, \rho \, \mathbf{I} \, . \tag{9}$$

Which is a conventional weight decay regularizer.

3 Fault Tolerance for Multiplicative weight Noise

In this section, we will first derive an objection function for minimizing the training error over weight noise. Our result shows that the proposed objective function consists of two terms. One is the conventional training error. Another one is the explicit regularizer proposed in [7]. That means, the explicit regularizer is used for minimizing training

error. Besides, we will also explain that why the conventional weight decay regularizer can improve the fault tolerant ability verse multiplicative weight noise. Afterwards, a fault tolerant training algorithm for multiplicative weight noise based on the proposed will be introduced. Lastly, we will discuss the way to estimate the prediction error of a network trained by the proposed algorithm.

3.1 Training error

The training error of an implementation $\tilde{\mathbf{w}}$ is given by

$$\mathcal{J}(\tilde{\mathbf{w}}) = \frac{1}{N}\sum_{j=1}^{N}(y_j - \Phi^T(\mathbf{x}_j)\,\tilde{\mathbf{w}})^2\,. \tag{10}$$

From (6), (10) becomes

$$\begin{aligned}\mathcal{J}(\mathbf{w},\mathbf{b}) &= \frac{1}{N}\sum_{j=1}^{N}\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)(w_i+b_i)\right)^2 \\ &= \frac{1}{N}\sum_{j=1}^{N}\left[\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)w_i\right)^2 + 2\left(\sum_{i=1}^{M}\phi_i(\mathbf{x}_j)b_i w_i\right)\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)w_i\right)\right. \\ &\quad \left. + \sum_{i=1}^{M}\sum_{i'=1}^{M}\phi_i(\mathbf{x}_j)\phi_{i'}(\mathbf{x}_j)b_i b_{i'} w_i w_{i'}\right]\,. \end{aligned} \tag{11}$$

Since b_is are identical independent zero mean random variables with symmetric density, the expectation value of $\mathcal{J}(\mathbf{w},\mathbf{b})$ is equal to

$$\bar{\mathcal{J}}(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N}\left[\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)w_i\right)^2 + \sum_{i=1}^{M}S_b\phi_i^2(\mathbf{x}_j)w_i^2\right]\,. \tag{12}$$

From (8), (12) can be rewritten in a matrix vector form:

$$\bar{\mathcal{J}}(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N}(y_j - \Phi^T(\mathbf{x})\,\mathbf{w})^2 + \mathbf{w}^T\mathbf{R}_{exp}\mathbf{w}. \tag{13}$$

$\bar{\mathcal{J}}(\mathbf{w})$ is the expected training error over weight noise. Compared with (8) adding the Bernier's explicit regularizer [7] with $\gamma = 1$ is to minimizing the expected training error over weight noise.

As mentioned in the last section, the regularizer $\mathbf{R}_{exp}$ reduces to the conventional weight decay regularizer if the functional network is defined as an RBF network and the RBF

width is small in compared with the input space. In sequel, the expected training error over weight noise will become

$$\frac{1}{N}\sum_{j=1}^{N}(y_j - \Phi^T(\mathbf{x})\,\mathbf{w})^2 + S_b\rho\mathbf{w}^T\mathbf{w}.$$

Which is an objective function that has been used to train an RBF network tolerable to multiplicative weight noise [13] [17].

3.2 Learning algorithm

Taking derivative (13) with respect to $\mathbf{w}$, we have

$$\frac{\partial\bar{\mathcal{J}}(\mathbf{w})}{\partial\mathbf{w}} = \frac{1}{N}\sum_{j=1}^{N}\Big(\Phi(\mathbf{x}_j)\Phi^T(\mathbf{x}_j)\,\mathbf{w} - \Phi(\mathbf{x}_j)y_j\Big) + \mathbf{R}_{exp}\,\mathbf{w}, \tag{14}$$

$$= (\mathbf{G} + \mathbf{R}_{exp})\,\mathbf{w} - \frac{1}{N}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)y_j\,. \tag{15}$$

Setting (15) to zero, the optimal weight vector for multiplicative weight noise (for minimizing the training error) is given by

$$\hat{\mathbf{w}} = (\mathbf{G} + \mathbf{R}_{exp})^{-1}\,\frac{1}{N}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)\,y_j. \tag{16}$$

4 Prediction error

Once an optimal weight vector $\hat{\mathbf{w}}$ has been attained, one would like to know its performance for the future data. Using similar approach as in [18], [29], [26], we are able to derive a formula for the case that multiplicative weight noise is introduced.

4.1 Data and network models

During the derive, we assume that the modeling error is very small. That means, we assume that the system output is given by,

$$y = \Phi^T(\mathbf{x})\mathbf{w}_o + e. \tag{17}$$

where $\mathbf{w}_0$ is the true system weight vector, and e is the random measurement noise with Gaussian distribution and variance equal to S_e.

For our learning algorithm, the trained weight vector is given by

$$\hat{\mathbf{w}} = (\mathbf{G} + \mathbf{R}_{exp})^{-1} \frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j)\, y_j.$$

Recall that when the trained weight is affected by weight noise, the implementation value is given by

$$\tilde{w}_i = \hat{w}_i + b_i\, w_i \;\; \forall \;\; i = 1, 2, \cdots, M\,, \tag{18}$$

where b_i's are identical independent mean zero random variables with variance S_b.

The difference between the true weight vector and the trained weight vector is

$$\begin{aligned}
\Delta\mathbf{w} &= \hat{\mathbf{w}} - \mathbf{w}_o \\
&= (\mathbf{G} + \mathbf{R}_{exp})^{-1} \frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j) y_j - \mathbf{w}_o \\
&= (\mathbf{G} + \mathbf{R}_{exp})^{-1} \left(\frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j) y_j - (\mathbf{G} + \mathbf{R}_{exp}) \mathbf{w}_o \right) \\
&= (\mathbf{G} + \mathbf{R}_{exp})^{-1} \left(\frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j) y_j - \frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j) \Phi^T(\mathbf{x}_j) \mathbf{w}_o + \mathbf{R}_{exp} \mathbf{w}_o \right) \\
&= (\mathbf{G} + \mathbf{R}_{exp})^{-1} \left(\frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j) e_j - \mathbf{R}_{exp} \mathbf{w}_o \right)
\end{aligned} \tag{19}$$

4.2 Training error

From the definition, the mean training error (MTE) of a fault-free network is given by

$$\begin{aligned}
\text{MTE} &= \frac{1}{N} \sum_{j=1}^{N} (y_j - \Phi^T(\mathbf{x}_j)\, \hat{\mathbf{w}})^2 \\
&= \frac{1}{N} \sum_{j=1}^{N} \left(\Phi^T(\mathbf{x}_j) \mathbf{w}_o + e_j - \Phi^T(\mathbf{x}_j)\, \hat{\mathbf{w}} \right)^2 \\
&= \frac{1}{N} \sum_{j=1}^{N} e_j^2 - \frac{2}{N} \sum_{j=1}^{N} e_j \Phi^T(\mathbf{x}_j)\, \Delta\mathbf{w} + \Delta\mathbf{w}^T\, \mathbf{G}\, \Delta\mathbf{w}
\end{aligned} \tag{20}$$

Substituting (19) into (20) and then taking the expectation on e_j's, we have

$$\begin{aligned}
\langle \text{MTE} \rangle &= S_e - 2 \frac{S_e}{N} \mathbf{Tr} \left\{ \mathbf{G}\, (\mathbf{G} + \mathbf{R}_{exp})^{-1} \right\} \\
&\quad + \frac{S_e}{N} \mathbf{Tr} \left\{ \mathbf{G}\, (\mathbf{G} + \mathbf{R}_{exp})^{-1}\, \mathbf{G}\, (\mathbf{G} + \mathbf{R}_{exp})^{-1} \right\} \\
&\quad + \mathbf{w}_o^T \mathbf{R}_{exp}\, (\mathbf{G} + \mathbf{R}_{exp})^{-1}\, \mathbf{G}\, (\mathbf{G} + \mathbf{R}_{exp})^{-1}\, \mathbf{R}_{exp} \mathbf{w}_o
\end{aligned} \tag{21}$$

where **Tr** is the trace operator, i.e., the sum of diagonal element of a matrix. *Equation (21) tells us the training error for a fault-free network trained by our proposed algorithm.*

4.3 Prediction error

Given an new sample $\mathbf{x}$, it is interesting to estimate the performance of the trained network affected by multiplicative weight noise. Denote $\mathcal{P}(\mathbf{x})$ and $\mathcal{P}(e)$ be the density function of the input $\mathbf{x}$ and the measurement noise e, respectively. The mean prediction error (MPE) of a faulty network is given by

$$\text{MPE} = \int\int (y - \Phi^T(\mathbf{x})\tilde{\mathbf{w}})^2 \mathcal{P}(\mathbf{x})\, \mathcal{P}(e)\, d\mathbf{x}\, de\,. \tag{22}$$

Considering the average over weight noise, we have

$$\text{MPE} = \int\int (y - \Phi^T(\mathbf{x})\hat{\mathbf{w}})^2 \mathcal{P}(\mathbf{x})\, \mathcal{P}(e)\, d\mathbf{x}\, de + \hat{\mathbf{w}}^T \mathbf{R}_{exp} \hat{\mathbf{w}}\,. \tag{23}$$

From (17), (23) becomes

$$\begin{aligned} \text{MPE} &= \int\int (\Phi^T(\mathbf{x})\mathbf{w}_o + e - \Phi^T(\mathbf{x})\hat{\mathbf{w}})^2 \mathcal{P}(\mathbf{x})\, \mathcal{P}(e)\, d\mathbf{x}\, de + \hat{\mathbf{w}}^T \mathbf{R}_{exp} \hat{\mathbf{w}} \\ &= \int\int (e - \Phi^T(\mathbf{x})\Delta\mathbf{w})^2 \mathcal{P}(\mathbf{x})\, \mathcal{P}(e)\, d\mathbf{x}\, de + \hat{\mathbf{w}}^T \mathbf{R}_{exp} \hat{\mathbf{w}}\,. \\ &= S_e + \Delta\mathbf{w}^T \mathbf{G} \Delta\mathbf{w} + \hat{\mathbf{w}}^T \mathbf{R}_{exp} \hat{\mathbf{w}}\,. \end{aligned} \tag{24}$$

Substituting (19) into (24) and taking the average over measurement noise e_j's, we have the real MPE, denoted as $\langle \text{MPE} \rangle$, given by

$$\begin{aligned} \langle \text{MTE} \rangle &= S_e + \frac{S_e}{N} \mathbf{Tr}\left\{ \mathbf{G} \left(\mathbf{G} + \mathbf{R}_{exp}\right)^{-1} \mathbf{G} \left(\mathbf{G} + \mathbf{R}_{exp}\right)^{-1} \right\} \\ &\quad + \mathbf{w}_o^T \mathbf{R}_{exp} \left(\mathbf{G} + \mathbf{R}_{exp}\right)^{-1} \mathbf{G} \left(\mathbf{G} + \mathbf{R}_{exp}\right)^{-1} \mathbf{R}_{exp} \mathbf{w}_o + \hat{\mathbf{w}}^T \mathbf{R}_{exp} \hat{\mathbf{w}}\,. \end{aligned} \tag{25}$$

Equation (25) tells us the error for a unseen sample for a faulty network trained by our proposed algorithm.

From (21) and (25), we have the following important relationship between the MPE of a network affected by weight noise and MTE of the trained network, given by

$$\langle MPE \rangle = \langle MTE \rangle + 2\frac{S_e}{N} \mathbf{Tr}\left\{ \mathbf{G} \left(\mathbf{G} + \mathbf{R}_{exp}\right)^{-1} \right\} + \hat{\mathbf{w}}^T \mathbf{R}_{exp} \hat{\mathbf{w}}, \tag{26}$$

If we assume that prediction error and training do not deviate much from their expected values, the following equation can be used as an estimation of the prediction error of a faulty network trained by the proposed algorithm:

$$\text{Prediction Error} \approx \text{Training Error} + 2\frac{S_e}{N}\mathbf{Tr}\left\{\mathbf{G}\left(\mathbf{G}+\mathbf{R}_{exp}\right)^{-1}\right\} + \hat{\mathbf{w}}^T\mathbf{R}_{exp}\hat{\mathbf{w}}, \quad (27)$$

Whenever N is large enough, this equation gives a good estimation on the prediction error of a model as one will see in the later section. For fault-free case ($S_b = 0$, i.e., $\mathbf{R}_{exp} = \mathbf{0}$),

$$\langle MPE\rangle = \langle MTE\rangle + 2S_e\frac{M}{N}. \quad (28)$$

This is the famous Akakie Information Criteria (AIC) [30]. As noted by Moody [18] and others working on information criteria, the factor S_e is usually an unknown but it can be approximated by setting [18]

$$S_e \approx \frac{N}{N-M_{eff}}\text{Training Error}$$

where M_{eff} is called the effective number of parameters defined by

$$M_{eff} = \mathbf{Tr}\left\{\mathbf{G}\left(\mathbf{G}+\mathbf{R}_{exp}\right)^{-1}\right\}.$$

Equation (26) and Equation (27) can then be rewritten as

$$\langle MPE\rangle = \left(\frac{N+M_{eff}}{N-M_{eff}}\right)\langle MTE\rangle + +\hat{\mathbf{w}}^T\mathbf{R}_{exp}\hat{\mathbf{w}} \quad (29)$$

and

$$\text{Prediction Error} \approx \left(\frac{N+M_{eff}}{N-M_{eff}}\right) \times \text{Training Error} + \hat{\mathbf{w}}^T\mathbf{R}_{exp}\hat{\mathbf{w}}. \quad (30)$$

Given S_b and a set of training data $\mathcal{D}_t$, the tolerant weight vector $\hat{\mathbf{w}}$ can be obtained. Afterwards, it is able to get the *Training Error* from the data set. From the regularization matrix $\mathbf{R}_{exp}$ and the Gram matrix $\mathbf{G}$, we can obtain the effective number M_{eff}. Putting all these factors in Equation (29) (or Equation (30)), the performance of the network in regard to weight noise effect can be estimated.

5 Simulations

To demonstrate the performance of the derived algorithm and the corresponding mean prediction error equation, we select a special functional network, the radial basis function

(RBF) network, and apply it to the following problems : (1) Hermite function approximation problem, (2) Nonlinear time series prediction problem and (3) Astrophysical time series prediction problem. For the first two problems, the datasets are generated artificially by computer programs. While the dataset used in the third problem is downloaded from an Internet website. It is a real-world dataset.

5.1 Hermite function approximation

The Hermite function is a nonlinear function is defined as follows :

$$f(x) = 1.1(1 - x + 2x^2)\exp(-x^2/2) + e, \tag{31}$$

where $x \in [-10, 10]$ and $e \sim \mathcal{N}(0, S_e)$ is a mean zero Gaussian noise of variance S_e. The shape of the noise free function is shown in Figure 1. Here in the figure, the output noise variance S_e is 0.01. One interesting property of Hermite function that makes it suitable

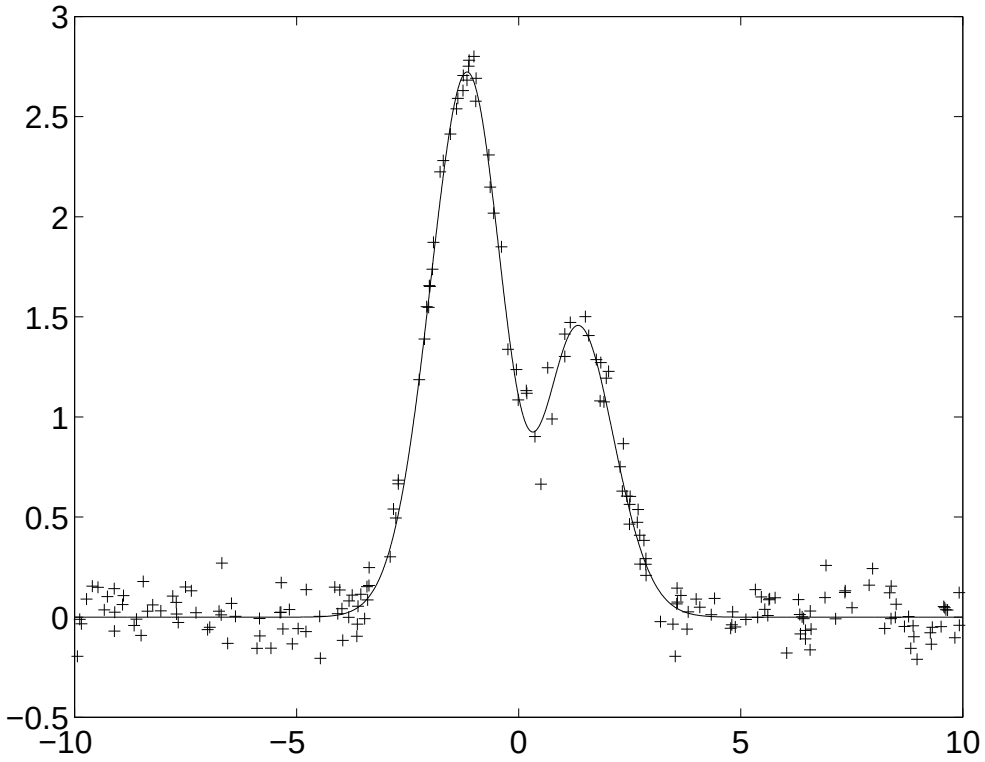


Fig. 1. Hermite function. The solid line corresponds to the noisy free function while the '+' dots corresponds to 200 samples from which the output noise with variance $S_e = 0.01$.

to be used as an illustrative example in our simulation is that it fluctuates mainly in the

middle portion around the origin, i.e. $x \in [-5, 5]$. For $|x| > 5$, the value of $f(x)$ is almost zero. Fit well the middle portion will lead to ripples at the flat region, i.e. over fit in the flat regions. Fit well the flat regions might lead to under fit the middle portion.

We assume a RBF network, denoted by $\hat{f}(x)$, consisting of M basis functions are used for all the experiments.

$$\hat{f}(x) = \Phi(x)^T \mathbf{w}. \tag{32}$$

The function $\Phi(x) = [\phi(x, c_1, \sigma), \phi(x, c_2, \sigma), \cdots, \phi(x, c_M, \sigma)]^T$ is defined as a Gaussian function of the form,

$$\phi(x, c, \sigma) = \exp\left(-\frac{(x-c)^2}{\sigma}\right).$$

σ controls the width of the radial basis function. In all the experiments, the centers c_is are defined in the following locations : $\{-9, -8.5, -8, \cdots, 8, 8.5, 9\}$.

5.1.1 Selection of σ

Since adding a regularizer can implicitly improve the generalization ability of a model, we need to select an appropriate value of σ such that the performance of the model is not very bad even no regularizer is added. Otherwise, it will be difficult to contrast the benefit of adding such a regularizer we have derived.

Obviously, for large σ (say $\sigma = 1$), the width of a RBF will be large and the matrix

$$\int \Phi(x)\Phi^T(x)\mathcal{P}(x)dx$$

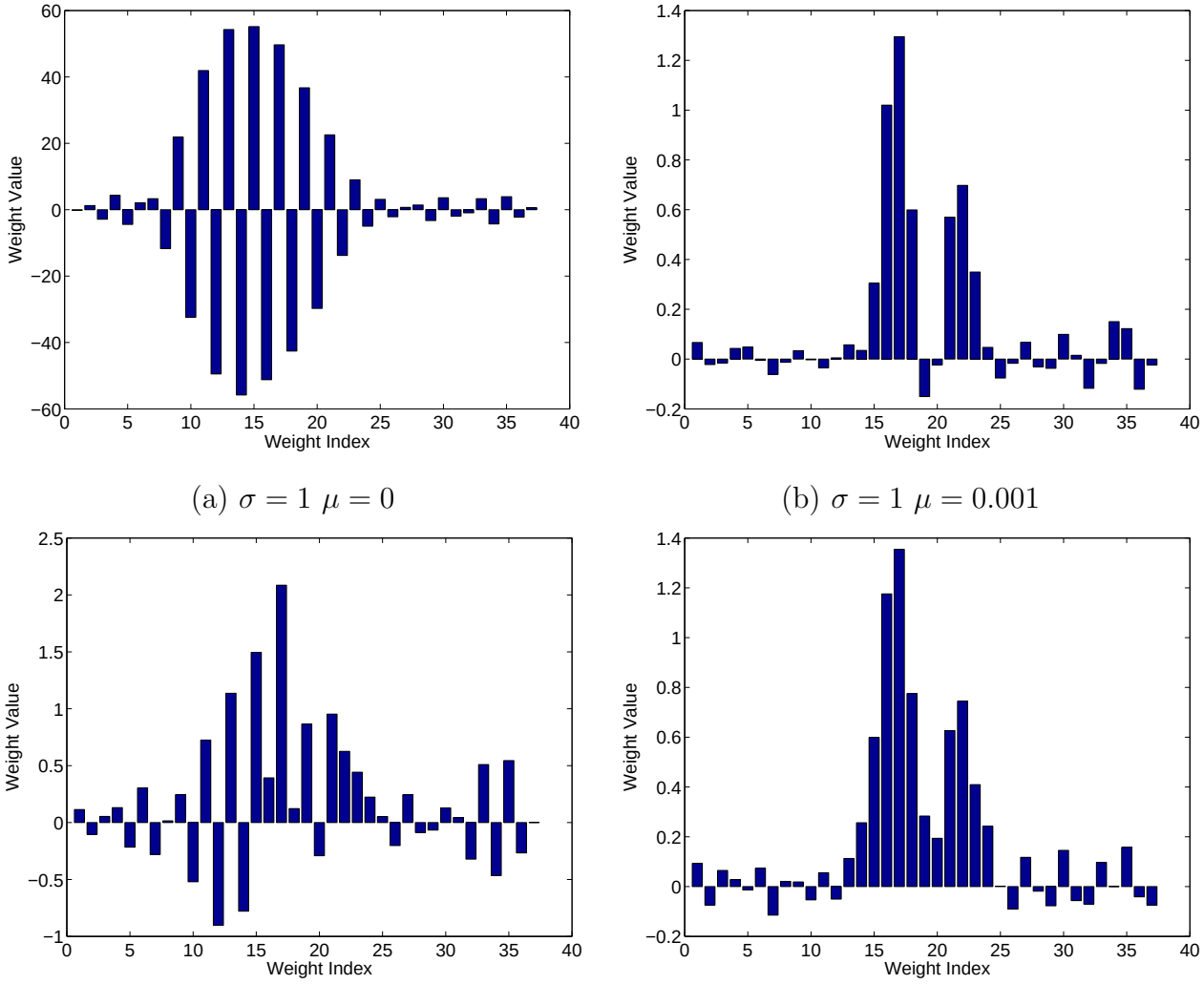
will be near singular. The parametric vector θ for the RBF,

$$\mathbf{w} = \left(\int \Phi(x)\Phi^T(x)\mathcal{P}(x)dx\right)^{-1}\left(\int \Phi(x)y\mathcal{P}(x)dx\right)$$

will be of large magnitude. To select an appropriate σ, we examine on four models attained by the following equation

$$\hat{\mathbf{w}} = \left(\frac{1}{N}\sum_{j=1}^{N}\Phi(x_j)\Phi^T(x_j) + \mu I_{M\times M}\right)^{-1}\left(\frac{1}{N}\sum_{j=1}^{N}\Phi(x_j)y_j\right)$$

with $\sigma = 0.49, 1$ and $\mu = 0, 0.001$. $\{(x_j, y_j\}_{j=1}^{200}$ are the training data as shown in Figure 1, the '+' signs. This is equivalent to adding a weight decay regularizer penalizing the weight magnitude. Figure 2 shows the weight values of the corresponding weight vectors. Not

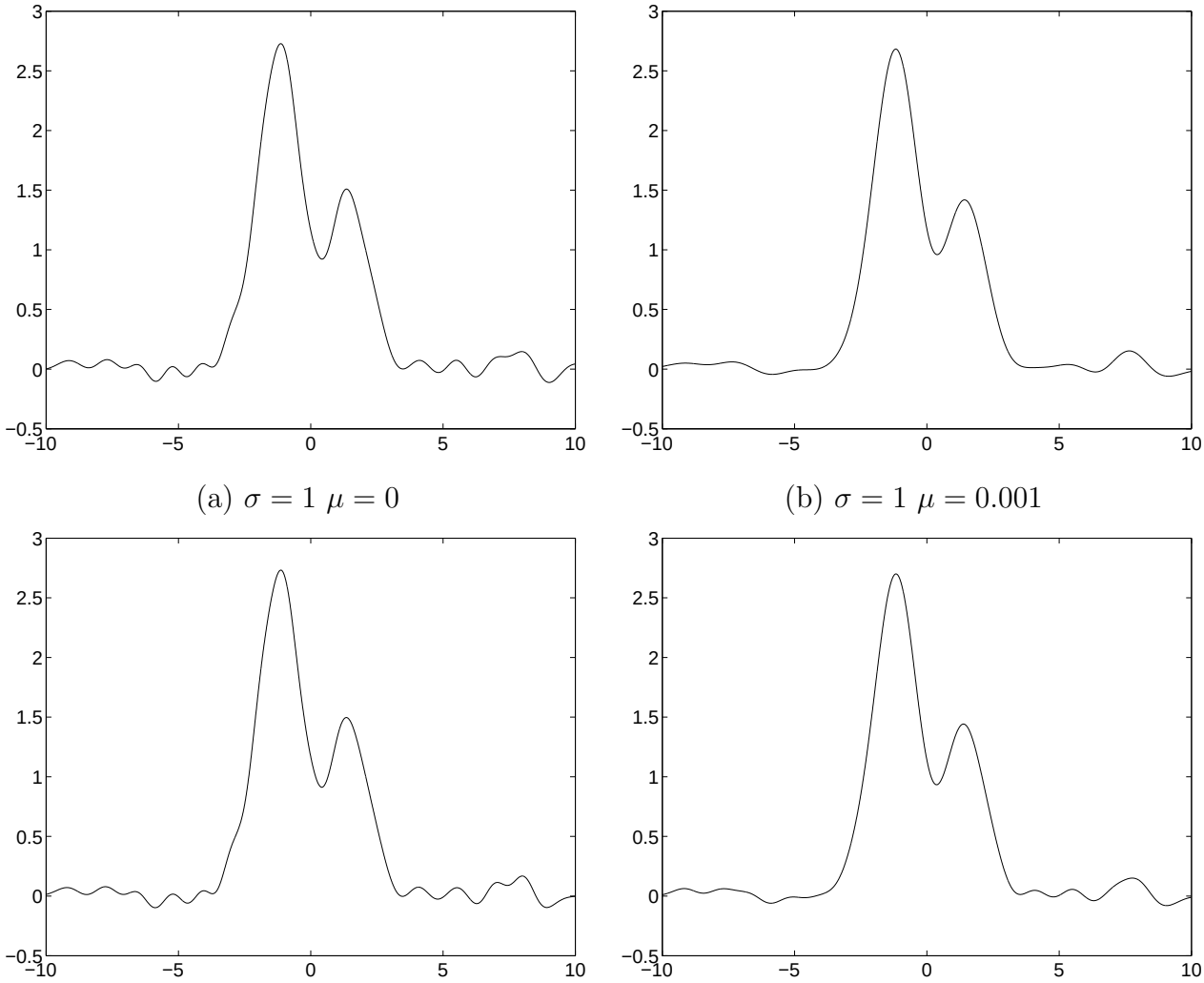


(a) $\sigma = 1$ $\mu = 0$ (b) $\sigma = 1$ $\mu = 0.001$

(c) $\sigma = 0.49$ $\mu = 0$ (d) $\sigma = 0.49$ $\mu = 0.001$

Fig. 2. The weight magnitudes under different value of σ and μ. The system noise S_e is 10^{-2}.

showing in this paper, we have found that the magnitude of the weight values will be even larger if the system noise variance S_e increases from 0.01 to 0.25. Adding a regularizer $\mu \mathbf{w}^T \mathbf{w}$ of very small μ (10^{-3}) can then control the weight magnitude. The functions reconstructed by the respectively weight vectors are shown in Figure 3. As expected, ripples appear in both cases when no regularizer has been added. Whenever a weight decay regularizer of small μ has been added, their weight magnitudes are controlled to smaller values. The approximated functions are smoothened, Figure 3. In case the weight vectors are corrupted by multiplicative weight noise, their corresponding functions are shown in Figure 4. Because of the weight values are large, the model attained by setting



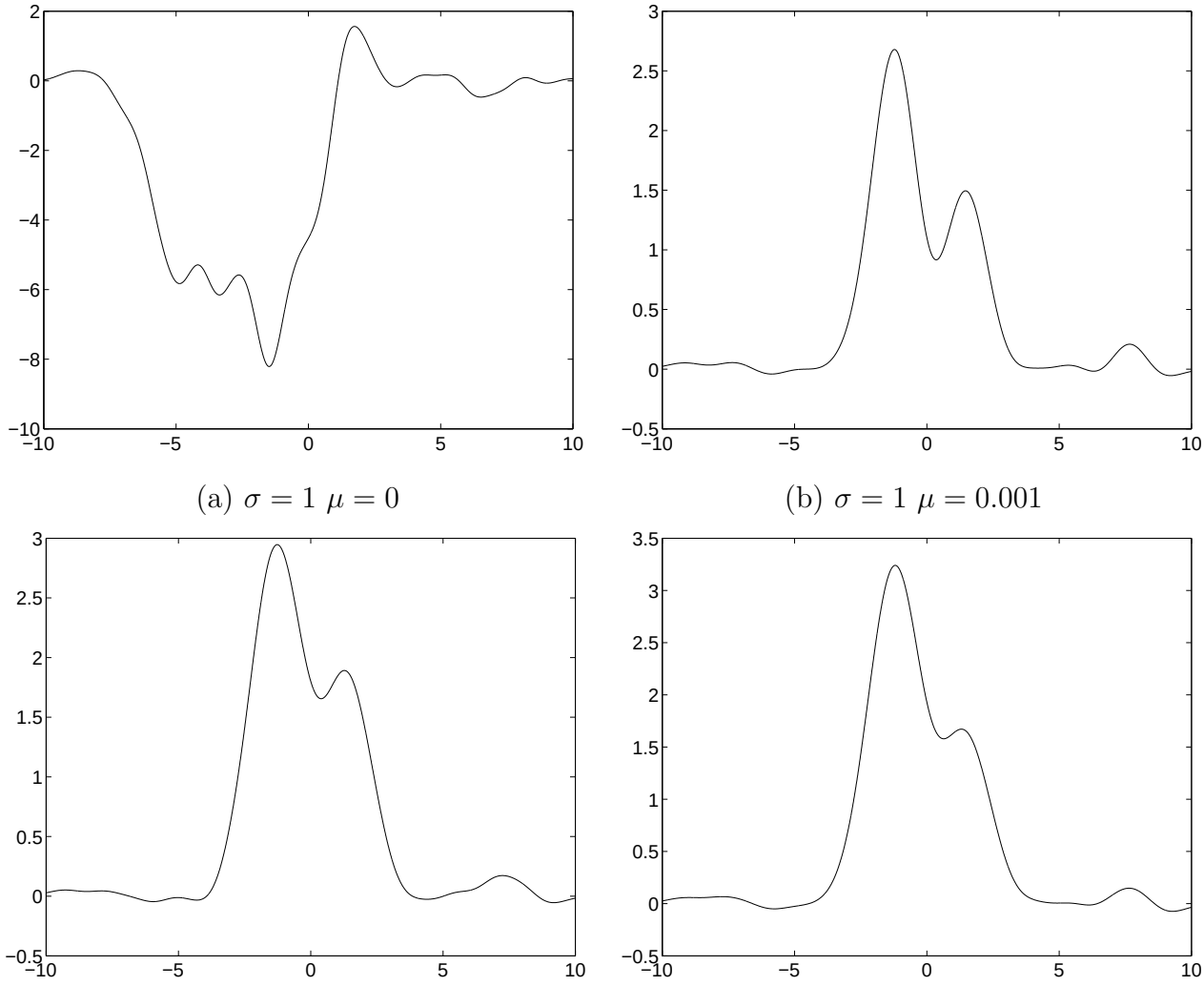
(a) $\sigma = 1$ $\mu = 0$ (b) $\sigma = 1$ $\mu = 0.001$

(c) $\sigma = 0.49$ $\mu = 0$ (d) $\sigma = 0.49$ $\mu = 0.001$

Fig. 3. The reconstructed function after training. The system noise S_e is 10^{-2}.

$\sigma = 1$ will be very sensitive to multiplicative weight noise. The model attained by setting $\sigma = 0.49$ will be not that sensitive.

Therefore, to avoid adding additional regularizer other than $\mathbf{R}_{exp}$, the value of σ is assigned to 0.49 simply because without adding $\mathbf{R}_{exp}$ the shape of a reconstructed function using $\sigma = 0.49$ and $\mu = 0$ is basically identical to the one attained by using $\sigma = 1$ and $\mu = 0$. In term of weight magnitude, the one obtained by setting $\sigma = 0.49$ and $\mu = 0$ will have similar value as the one obtained by setting $\sigma = 0.49$ and $\mu = 0.001$ or $\sigma = 1$ and $\mu = 0.001$, as shown in Figure (3). Furthermore, in term of the sensitivity to multiplicative weight noise, setting $\sigma = 0.49$ and $\mu = 0$ will have similar effect as the one obtained by



(a) $\sigma = 1$ $\mu = 0$ (b) $\sigma = 1$ $\mu = 0.001$

(c) $\sigma = 0.49$ $\mu = 0$ (d) $\sigma = 0.49$ $\mu = 0.001$

Fig. 4. The reconstructed function while multiplicative weight noise ($S_b = 10^{-2}$) has been added. The system noise S_e is 10^{-2}.

setting $\sigma = 0.49$ and $\mu = 0.001$ or $\sigma = 1$ and $\mu = 0.001$, as shown in Figure (4). $\sigma = 0.49$ is an appropriate choice for the width of the radial basis function. Thus we can ignore the regularizer term $\mu \mathbf{w}^T \mathbf{w}$ in all the experiments and focus on studying the tolerance ability of the trained RBF using the explicit regularizer.

5.1.2 Gaussian weight noise

For the study of multiplicative weight noise effect, we assume the following noise model :

$$\tilde{w}_i = w_i + b_i w_i \tag{33}$$

for all $i = 1, \cdots, M$ and b_i is a mean zero Gaussian noise with variance S_b. Furthermore, we assume that the random variables b_i and b_j are independent and S_b is small in order to ensure that $\tilde{w}_i$ and w_i are of same sign.

For each experiment, a set of training data and a set of testing data are generated by using noisy model, Equation (31). Fix a value for λ, the training set is used for obtaining the best $\hat{\mathbf{w}}$, i.e.

$$\hat{\mathbf{w}}(\lambda) = (\mathbf{G} + \lambda \mathbf{R})^{-1} \left(\frac{1}{N} \sum_{j=1}^{N} \Phi(x_j) y_j \right),$$

where

$$\mathbf{R} = \frac{1}{N} \begin{bmatrix} \sum_{j=1}^{N} \phi_1^2(\mathbf{x}_j) & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{N} \phi_2^2(\mathbf{x}_j) & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & 0 & \sum_{j=1}^{N} \phi_M^2(\mathbf{x}_j) \end{bmatrix}.$$

The training error $MTE(\hat{\mathbf{w}}(\lambda))$ is recorded. The best $\hat{\mathbf{w}}$ is plugged in the testing data and the testing error $MPE(\hat{\mathbf{w}}(\lambda))$ is recorded. Finally, a perturbed weight $\tilde{\mathbf{w}}(\lambda)$ is generated by the same $\hat{\mathbf{w}}(\lambda)$ using the following method :

$$\tilde{w}_i = \hat{w}_i(\lambda) + b_i \tilde{w}_i(\lambda),$$

where b_i is a random variable of zero mean variance S_b. The corresponding $\tilde{\theta}$ is thus plugged in the same testing set and the testing error, i.e. MPE$(\hat{\mathbf{w}}, b)$, is recorded. For the same $\hat{\mathbf{w}}(\lambda)$, we generate 200 random $\tilde{\mathbf{w}}$ and the error shown on the curve is the average over these 200 simulations.

Figure (5 – 7) show the results obtained by our simulation for different S_b and S_e. On the left column, the results for which S_b equals to 0.16 are shown. While on the right column, the results are for S_b equals to 0.25. For each row of figures, the observation noise variance is different. Six different values of S_e are examined, $0, 0.01, 0.04, 0.09, 0.16$ and 0.25. For each figures, three different curves — the training error (no multiplicative weight noise), the testing error (no multiplicative weight noise) and the testing error in which the RBF is perturbed by multiplicative weight noise — are shown. In the figures, we name them as *Training Error* curve MTE$(\hat{\mathbf{w}}(\lambda))$, *Testing Error* curve MPE$(\hat{\mathbf{w}}(\lambda))$ and *Weight Noise* curve MPE$(\hat{\mathbf{w}}(\lambda), b)$. A few points can be noted from these figures.

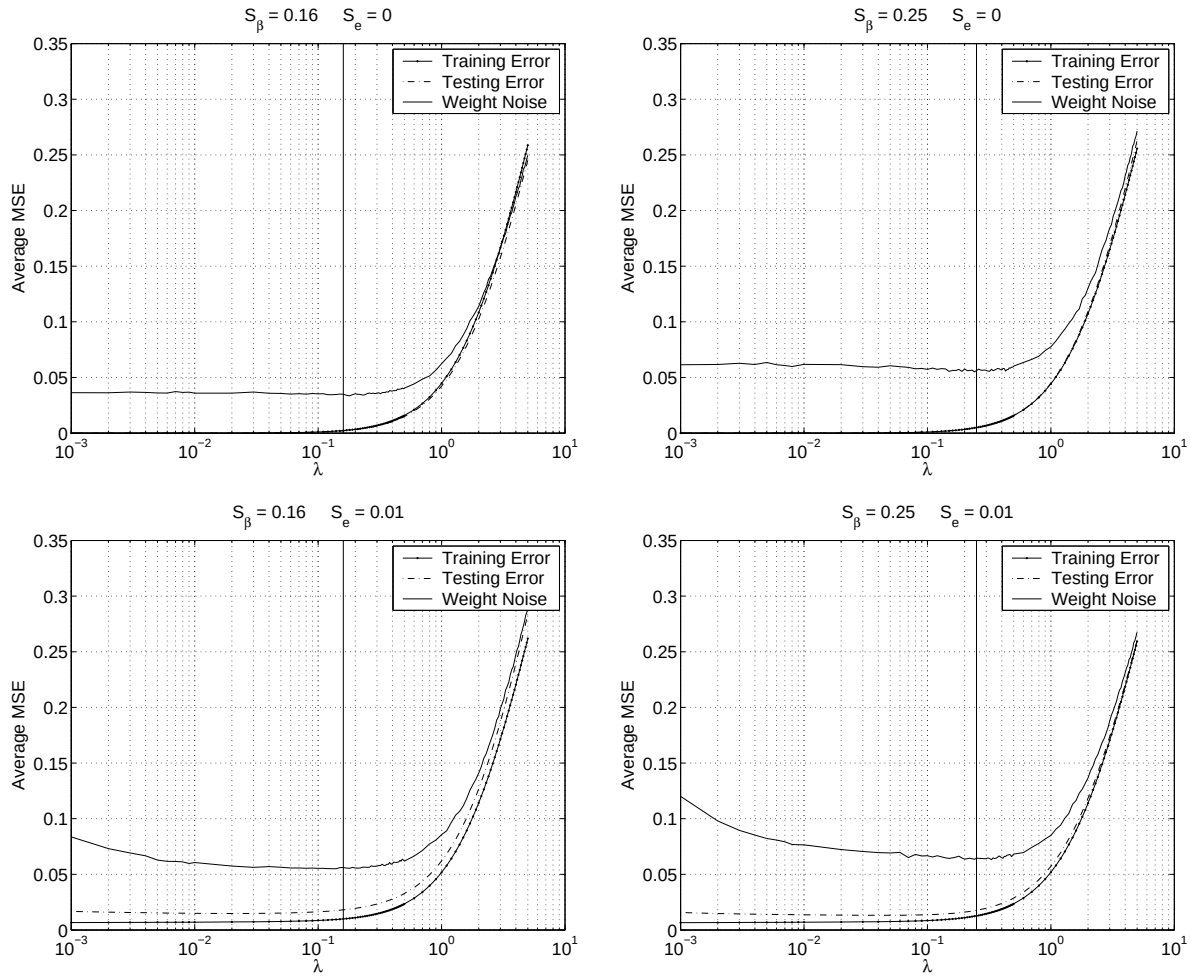


Fig. 5. Simulation results for different values of multiplicative weight noise variance S_b and the observation noise variance S_e is small. The figures on left column are for the S_b being 0.16 while the figures on the right column are for the S_b being 0.25. Besides, the results shown on different rows are simulated by using different S_e values. The solid lines, labeled "Weight Noise", are corresponding to the simulation results that multiplicative weight noise has been added. Showing on this figures, the values for S_e are defined to be 0 and 0.01. The vertical dash line corresponds to $\lambda = S_b$.

• As usual, whenever the model noise variance S_e is large, training with no regularizer (i.e. $\lambda = 0$), the weight noise free testing error must be larger than the weight noise free training error. But we are able to improve the generalization ability of a weight noise free model by adding a regularizer with small λ, as observed from the *Testing Error* curves.

• Whenever multiplicative weight noise takes effect, training with no or small regularizer can give a very poor fitting. It can be noted from the *Weight Noise* curves when λ is small. Besides, the larger the S_e is, the larger the error. Compare the $\mathrm{MPE}(\hat{\mathbf{w}}(\lambda), b)$ and

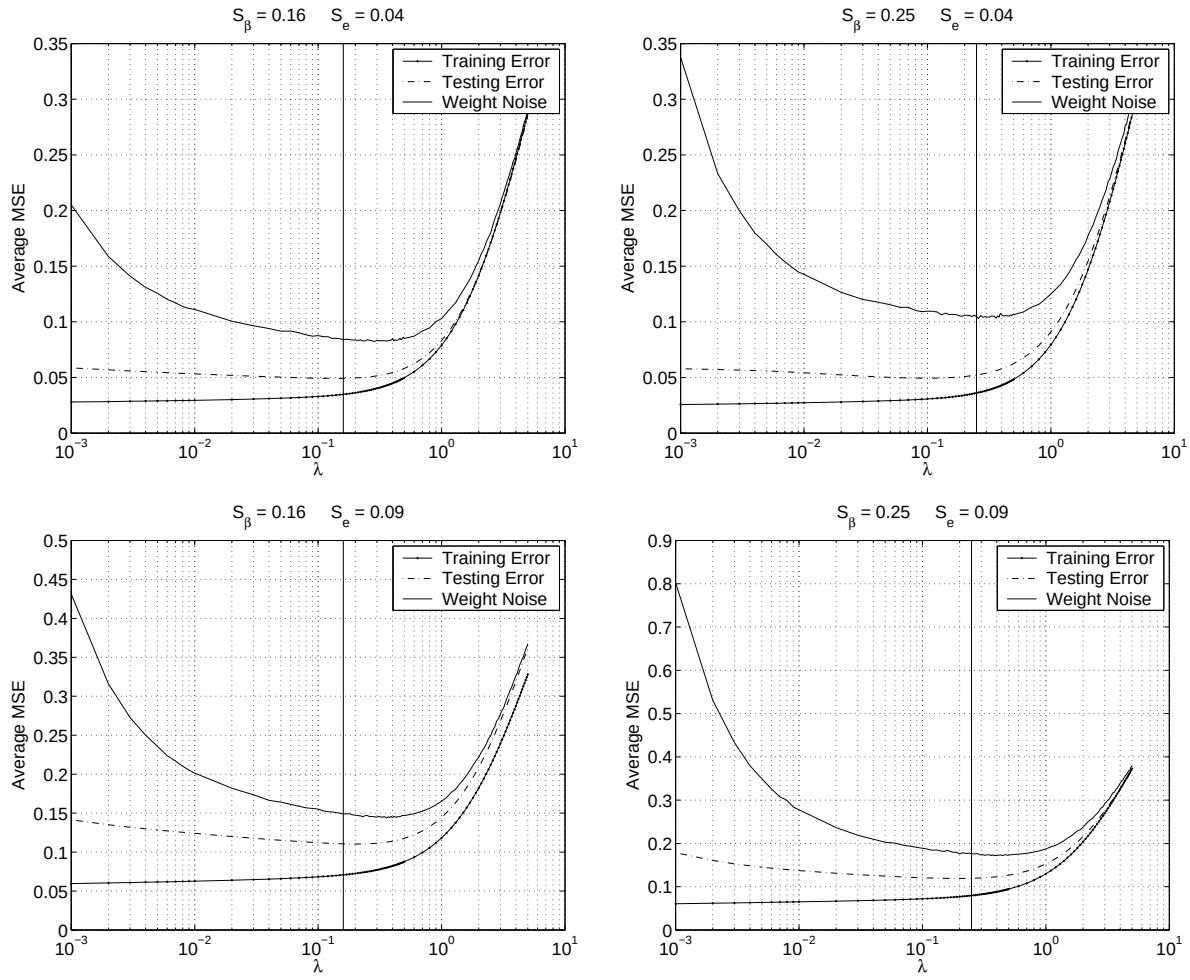


Fig. 6. Simulation results for different values of multiplicative weight noise variance S_b and observation noise variance S_e. The figures on left column are for the S_b being 0.16 while the figures on the right column are for the S_b being 0.25. The solid lines, labeled "Weight Noise", are corresponding to the simulation results that multiplicative weight noise has been added. Showing on this figures, the values for S_e are defined to be 0.04 and 0.09. The vertical dash line corresponds to $\lambda = S_b$.

$\mathrm{MPE}(\hat{\mathbf{w}}(\lambda))$, it is observed that

$$\mathrm{MPE}(\hat{\mathbf{w}}(\lambda), b) > \mathrm{MPE}(\hat{\mathbf{w}}(\lambda)),$$

whenever $S_b > 0$ and $\lambda < 1$. Nevertheless, their difference increases as S_e increases. This results reflect that adding regularizer not just can improve the performance of a model against large observation noise, but also can improve the vulnerability of a model against multiplicative weight noise perturbation.

- For the cases that $S_e = 0$, the optimal regularization constant λ can simply be defined

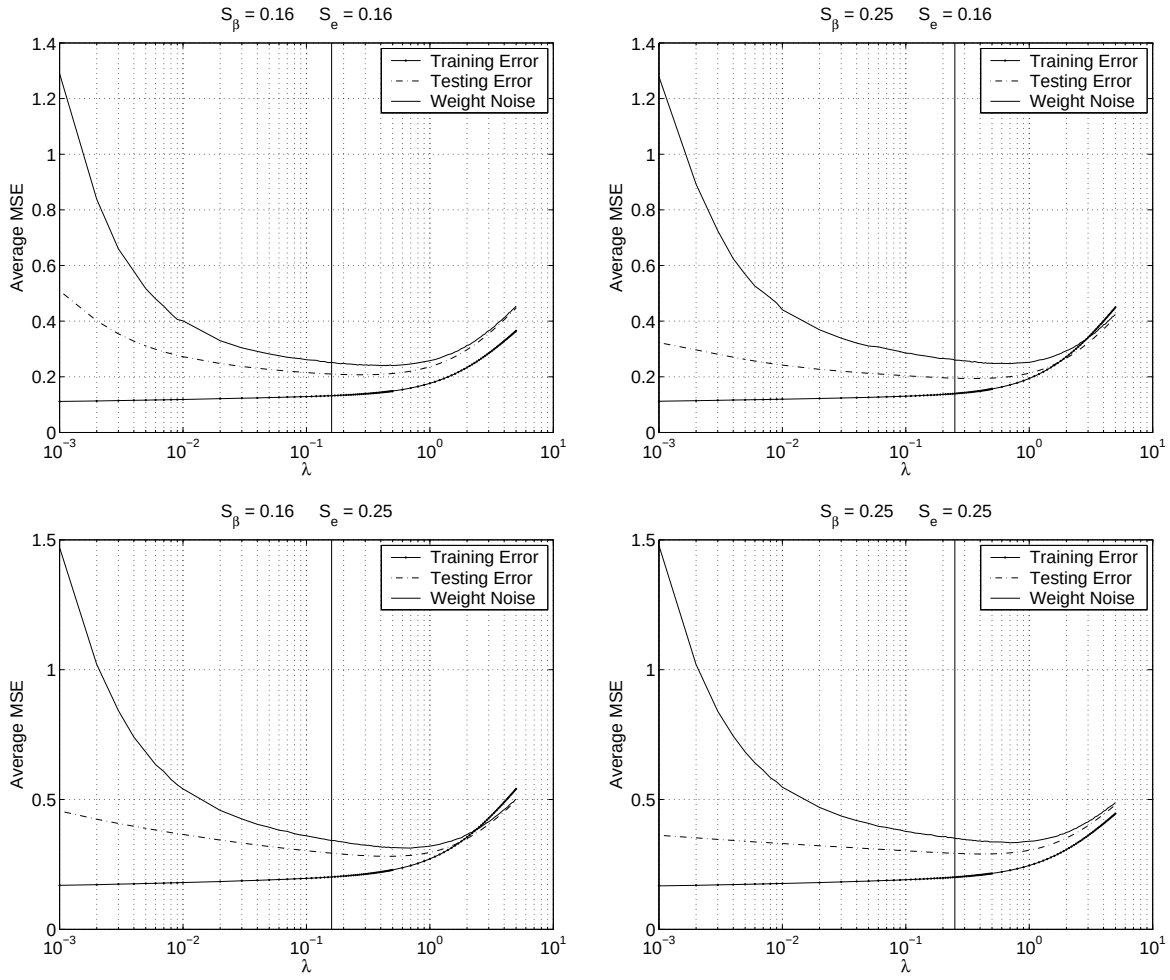


Fig. 7. Simulation results for different values of multiplicative weight noise variance S_b and the observation noise variance S_e is large. The figures on left column are for the S_b being 0.16 while the figures on the right column are for the S_b being 0.25. The solid lines, labeled "Weight Noise", are corresponding to the simulation results that multiplicative weight noise has been added. Showing on this figures, the values for S_e are defined to be 0.25. The vertical dash line corresponds to $\lambda = S_b$.

as S_b, as observed from Figure 8. It indeed confirms our theoretical result. Whenever $S_e > 0$, the optimal regularizer constant λ value shifts to right. It is believed that it is due to the over fitting problem when the number of training data is small. The larger the S_e is, the effect of over fitting will be larger than the effect of weight noise. To compensate the effect, λ will have to be a value larger than S_b or the number of training data have to be increased. From another experiment not shown here, we have found that the optimal λ converges to S_b when N increases.

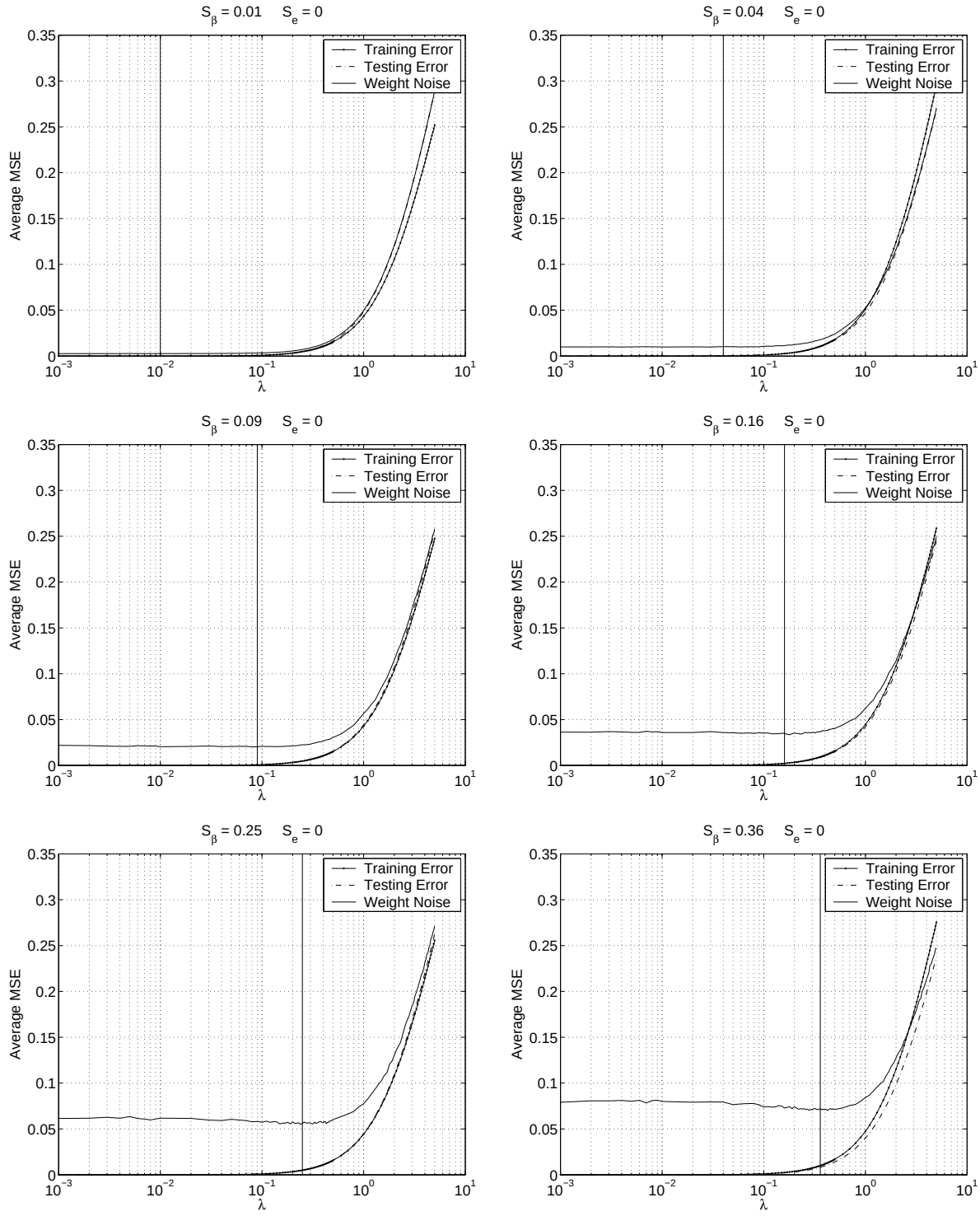


Fig. 8. For the cases that $S_e = 0$, the regularization constant λ can simply be defined as S_b. The solid lines, labeled "Weight Noise", are corresponding to the simulation results that multiplicative weight noise has been added. The vertical dash line corresponds to $\lambda = S_b$.

5.1.3 Prediction error estimation

In this simulation, we have compared the estimated prediction error against the actual predict error for N being 100 (Figure 9a) 1000 (Figure 9b). The simulation procedure is

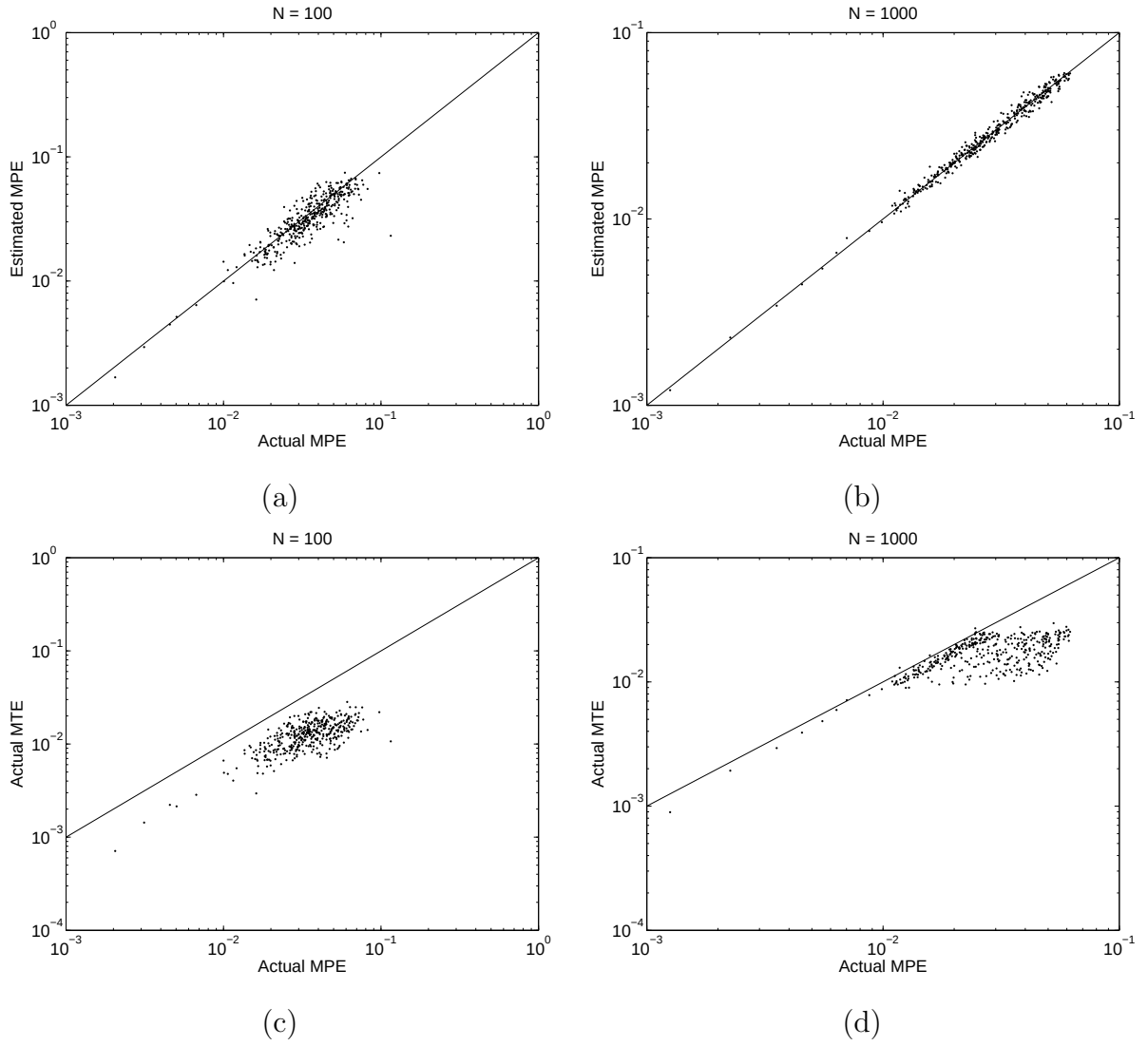


Fig. 9. Estimated prediction error versus actual prediction error for N equals to 100 (a) and 1000 (b). The plots on the training error against actual prediction error (c) (d) are for reference. A straight line corresponding to $y = x$ is added for clarification.

shown in Figure 10. Obviously, the larger the value of N, the better the approximation is. For $N = 1000$, the estimated error and the actual error almost coincides with the straight. While for $N = 100$, some data points deviate from the straight. It is due to the fact that the training set and the testing set are generated independently. As the estimation

```
Se = [0.001:0.001:0.025];
Sb = [[0.001:0.001:0.009] [0.01:0.01:0.16]];

FOR Each Se and Sb
        Randomly generate a Training Set of N data;
        Randomly generate a Test Set of N data;
        Obtain an optimal model M;
        Calculate the Testing Error (TE) and Training Error (TR);
        Estimate the Expected Testing Error (MTN);
        Repeat 1000 times
             (1) FOR Each Weight
                      Generate random Gaussian noise 'rand'
                      Weight = Weight (1 + rand);
                 END
             (2) Calculate the Error (TN);
         Calculate the Mean TN;
END
```

Fig. 10. Simulation procedure for prediction error estimation.

is relied on the training error, an exceptional small training error will eventually lead to a poor estimation, as can be observed from Figure 9c.

5.2 Time series prediction

We consider the following nonlinear autoregressive (NAR) time series [31], given by

$$\begin{aligned} y(i) &= \left(0.8 - 0.5\exp(-y^2(i-1))\right) y(i-1) - \left(0.3 + 0.9\exp(-y^2(i-1))\right) y(i-2) \\ &\quad +0.1\sin(\pi y(i-1)) + e(i), \end{aligned} \tag{34}$$

where $e(i)$ is a mean zero Gaussian random variable that drives the series. Its variance is equal to 0.09. Figure 11 shows a typical phase plot of the series with $y(0) = y(-1) = 0.1$.

In each of the following simulation, two independent sets of data are generated by the

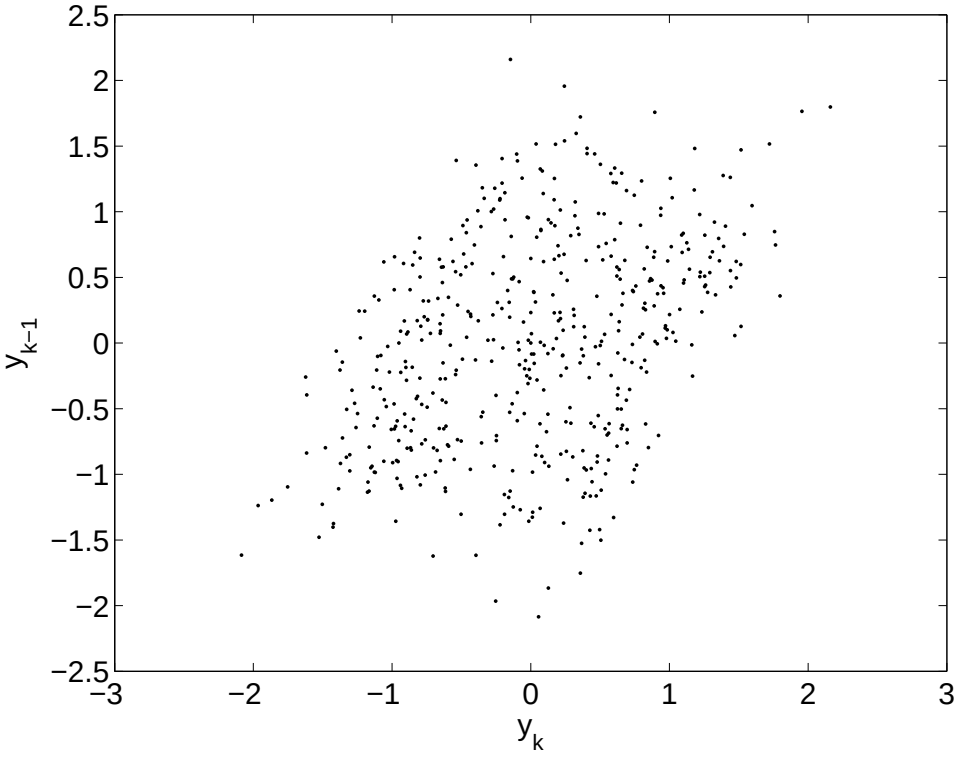


Fig. 11. The typical phase plot of the nonlinear system with initial condition: $y_0 = y_1 = 0.1$. The output noise S_e is 0.09.

Equation (34) with the same initial condition : $y(0) = y(-1) = 0.1$, and S_e is set to 0.01. The first set will be used for training, while the second set is used for testing. Our RBF model is used to predict $y(i)$ based on the past observations $y(i-1)$ and $y(i-2)$, given by

$$\hat{y}(i) = \hat{f}(\mathbf{x}_i, \mathbf{w}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}(i)), \tag{35}$$

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

The structure of the model consists of 500 basis functions :

$$\hat{f}(\mathbf{x}, \mathbf{w}) = \sum_{j=1}^{500} w_j \phi_j(\mathbf{x}) \tag{36}$$

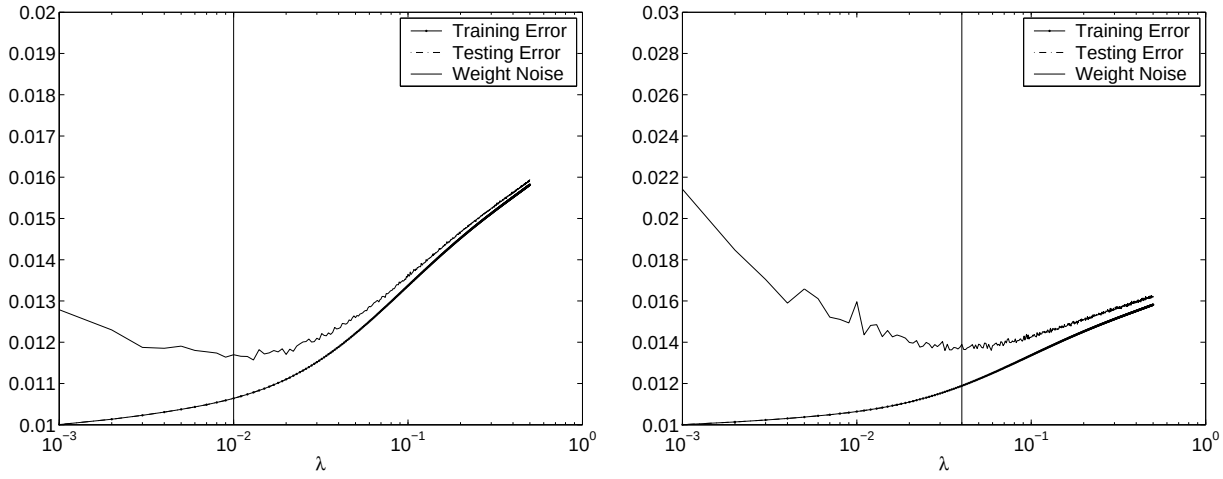
where $\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{c}_i\|^2}{\Delta}\right)$. The centers are assigned as $\mathbf{c}_1 = [y(0), y(1)]^T, \cdots, \mathbf{c}_{500} = [y(499), y(500)]^T$, where $y(\cdot)$s are from the training dataset. The width of the centers Δ is set to 0.81.

5.2.1 Gaussian weight noise

Similar to the simulation conducted for Hermite function approximation, we would like to see how robust the learning algorithm on the value of S_b. Without loss of generality, we consider two values of S_b : $S_b = 0.01$ and $S_b = 0.04$. For each S_b, the training set is used for obtaining the best $\hat{\mathbf{w}}$ for each $\lambda \in [0.001, 0.5]$ by the following equation.

$$\hat{\mathbf{w}}(\lambda) = (\mathbf{G} + \lambda \mathbf{R})^{-1} \left(\frac{1}{N} \sum_{j=1}^{N} \Phi(\mathbf{x}_j) y_j \right). \tag{37}$$

Then 100 perturbed $\hat{\mathbf{w}}(\lambda)$ are generated by adding Gaussian multiplicative weight noise, and plugged in the testing dataset. The noise-free training error, noise-free testing error and the average multiplicative weight noise corrupted testing error are recorded. The above experiment is then repeated for 10 times and the results are depicted in Figure 12.



(a) $S_b = 0.01$ (b) $S_b = 0.04$

Fig. 12. Average mean square errors plot against different value of λ. The solid lines, labeled "Weight Noise", are corresponding to the simulation results that multiplicative weight noise has been added. The vertical dash line corresponds to $\lambda = S_b$.

Similar to the previous example, increasing the value of λ does not improve the generalization ability of the network. However, it can improve the weight noise tolerance ability. Besides, the network can have its best tolerant ability if λ is close to S_b.

5.2.2 Prediction error estimation

To illustrate the estimation of the prediction error, a similar procedure as shown in Figure 10 has been taken. In this experiment, 50 values of S_b and 4 values of S_e are examined.

$$\begin{aligned} S_b &\in \{0.01, 0.02, \cdots, 0.49, 0.5\} \\ S_e &\in \{0.01, 0.04, 0.09, 0.16\} \end{aligned}$$

For each (S_b, S_e), 50 training datasets and 50 testing datasets are generated. Let $\mathcal{D}_k^{Train}$ and $\mathcal{D}_k^{Test}$ be the k^{th} training and testing datasets, where $k = 1, 2, \cdots, 50$. The following steps are then carried out for each k.

1. Obtain $\hat{\mathbf{w}}$, matrix $\mathbf{G}$, $\mathbf{R}_{exp}$ and the training error (say MTE_k) using $\mathcal{D}_k^{Train}$.

2. Calculate the effective number of parameter M_{eff} by using the factors obtained.

3. Estimate the prediction error (say $PE_k^{Estimated}$) by

$$PE_k^{Estimated} = \left(\frac{N + M_{eff}}{N - M_{eff}}\right) MTE + \hat{\mathbf{w}}^T \mathbf{R}_{exp} \hat{\mathbf{w}}.$$

4. Generate 100 perturbed weight vectors with respect to $\hat{\mathbf{w}}$ and then plug the perturbed weight vectors in $\mathcal{D}_k^{Test}$ to obtain the actual prediction error PE_k^{Actual}.

Once the above steps have been repeated for $k = 1, \cdots, 50$, the estimated MPE (i.e. $\langle MPE \rangle$) is obtained by

$$\frac{1}{50} \sum_{k=1}^{50} PE_k^{Estimated}$$

and the actual MPE is obtained by

$$\frac{1}{50} \sum_{k=1}^{50} PE_k^{Actual}.$$

Figure 13 shows the plot of estimated MPE against the actual MPE. The four segments of points are corresponding to four different values of S_e. Starting from the bottom left to the upper right, the values are 0.01, 0.04, 0.09 and 0.16. It is clear that the estimated mean prediction error fits well to the actual mean prediction error.

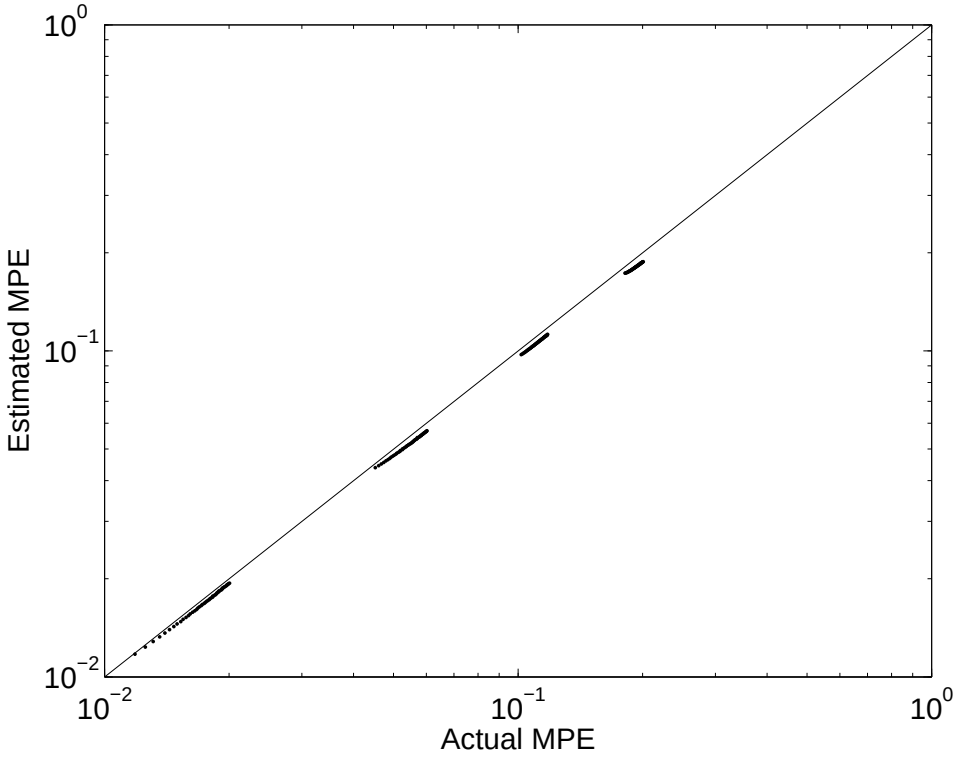


Fig. 13. Estimated MPE versus the actual MPE for the time series data.

5.3 Astrophysical data

The astrophysical data is a the time series recording the time variation of the intensity of the white dwarf star PG1159-035 during March 1989 [32] [33]. The data samples are noisy and nonlinear in nature and can be downloaded from http://www-psych.stanford.edu/∼andreas/Time-Series/. The whole dataset is composed of 17 parts. Each is of different size. In the first experiment, we only use the Part I dataset. While in the second experiment about the estimation of mean prediction error, we use all 17 parts.

5.3.1 Gaussian weight noise

Similar to the experiments conducted in the previous sections, we would like to investigate the performance of a RBF network if the constant factor in $\mathbf{R}_{exp}$ is not identical to S_b. The study will be based on the Part I of the dataset. The time series of this part is shown in Figure 14. It consists of 618 data samples. An RBF network is treated as a nonlinear regressor, where the output is the prediction of $y(i)$ and

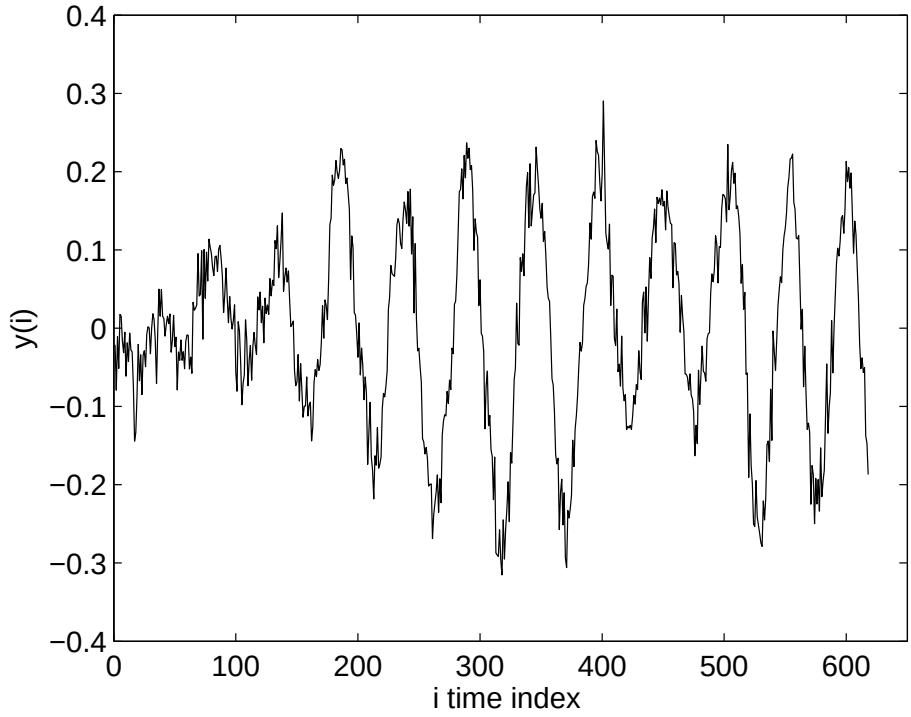


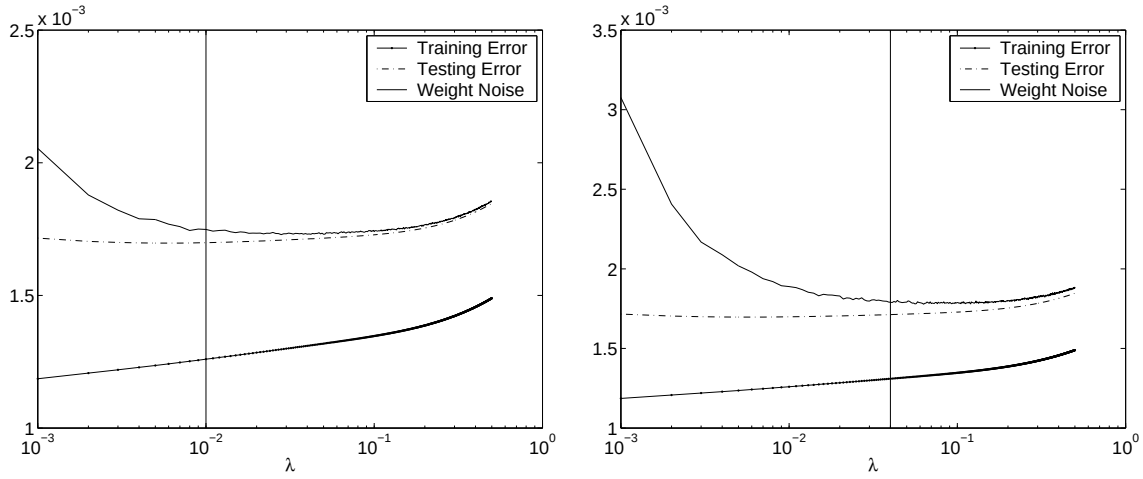
Fig. 14. The Part I of the Astrophysical data.

it is predicted by means of 5 inputs values $y(i-1), \cdots, y(i-5)$. Before the experiment start, the dataset is preprocessed by concatenating six consecutive values of y as a pattern vector. In such case, there are totally 613 vectors : $\{(\mathbf{x}_k, y(k+5))\}_{k=1}^{613}$ $= [y(1), y(2), \cdots, y(6)]^T$, $[y(1), y(2), \cdots, y(7)]^T$, $\cdots$, $[y(1), y(2), \cdots, y(618)]^T$. The corresponding input $\mathbf{x}$ is defined as $\mathbf{x}_1 = [y(1), y(2), \cdots, y(5)]^T$, $\mathbf{x}_2 = [y(2), y(3), \cdots, y(6)]^T$, $\cdots$, $\mathbf{x}_{613} = [y(613), y(614), \cdots, y(617)]^T$.

Two values of S_b, 0.01 and 0.04 respectively, are examined. The parameter λ is set to 0.001, 0.002 and so on up to 0.5. For each particular (S_b, S_e) pair, 50 training sets (and the corresponding testing sets) are generated by randomly selection of 306 patterns from the 613 pattern vectors to be the training set and then the reminding 307 pattern vectors to be the corresponding testing set.

An RBF network is thus generated for each of these training sets. Each RBF network consists of 306 radial basis functions, in which the centers are defined as the selected $\mathbf{x}_k$ in the training set and the corresponding weight vector $\hat{\mathbf{w}}$ is obtained by Equation 37.

Finally, 100 perturbed weight vectors are generated and plugged in the corresponding testing set to check with the testing error.



(a) $S_b = 0.01$ (b) $S_b = 0.04$

Fig. 15. Average mean square errors plot against different value of λ. The solid lines, labeled "Weight Noise", are corresponding to the simulation results that multiplicative weight noise has been added. The vertical dash line corresponds to $\lambda = S_b$.

The results are depicted in Figure 15. Each point shown in the figure is the average over 100 perturbed weight vectors and 50 training sets. It is clearly found that the fault tolerant performance of the network is approximately the same as the best performance whenever λ is set to S_b.

5.3.2 Prediction error estimation

To demonstrate the applicability of the mean prediction error equation, we follow the same procedure as described for time series prediction problem in Section 5.2.2. All 17 parts of the Astrophysical data are examined and the values of S_b are set to be 0.005, 0.010, 0.015, $\cdots$, 0.250. For each of the 17 datasets, 50% of the pattern vectors are randomly selected as training set and the reminding 50% are assigned as testing set. Then, we follow the same procedure as in Section 5.2.2 to obtain the actual and estimated mean prediction error. The steps are repeated ten times for each particular S_b and for each part. The results are depicted in Figure 16. It is clear that the estimated mean prediction error is similar to the actual mean prediction error for all 17 parts of Astrophysical data.

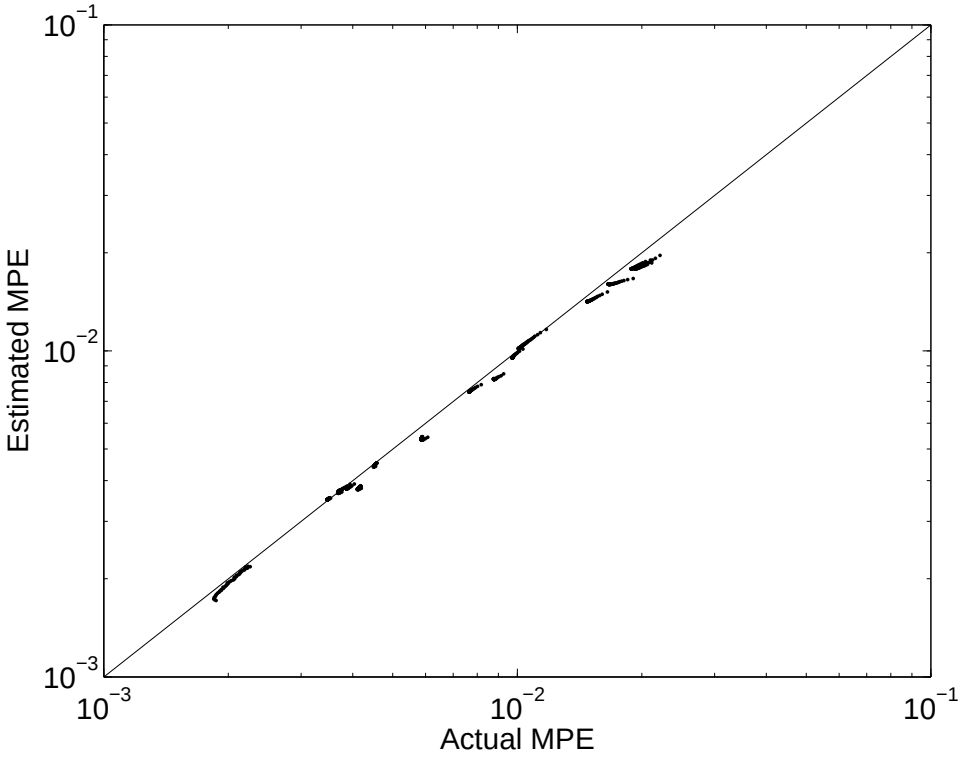


Fig. 16. Estimated MPE versus the actual MPE for the astrophysical data.

6 Conclusion

In this paper, an objective function for training a functional neural network to tolerate multiplicative weight noise has been derived. This objective function has the form similar to other regularizer-based objective functions. For a functional network is of the form $\Phi^T(\mathbf{x})\mathbf{w}$, this objective function is equal to

$$\text{Mean Square Training Error} + \mathbf{w}^T\mathbf{R}_{exp}\mathbf{w},$$

where $\mathbf{R}_{exp}$ is a diagonal matrix with elements depended on the multiplicative weight noise and the Gram matrix $\mathbf{G}$ defined as $\frac{1}{N}\sum_{j=1}^{N}\Phi^T(\mathbf{x}_j)\Phi(\mathbf{x}_j)$. For a special case that the functional network is a RBF network and its basis functions are of small variances, it has further been shown that this fault tolerant regularizer reduces to the conventional weight decay regularizer. Thus, it explains why *adding weight decay can improve the fault tolerant ability of a RBF network in dealing with multiplicative weight noise.* Based on the objective function being derived, a fault tolerant learning algorithm and a formula for

mean prediction error have also been derived analytically in the paper. It is demonstrated by the simulation results obtained in three problems – the Hermit function approximation, nonlinear time series prediction and the Astrophysical time series prediction – the applicability of this mean prediction error formulae. Finally, one should also noted that functional link network behaves similar to a generalised linear model [34]. Extended work on the fault tolerant ability of a generalised linear model is worthwhile for future investigation.

Acknowledgement

The authors would like to express their gratitude to the associate editor and the referees for their valuable comments. The work presented in this paper is supported by a research grant from the Hong Kong Special Administrative Region RGC Earmarked Grant (Project No. CityU 115606) and a grant from National Science Council ROC Project Grant (No. NSC 95-2221-E-040-009).

References

[1] J. Burr, "Digital neural network implementations," *in Neural Networks, Concepts, Applications, and Implementations, Vol III. Englewood Cliffs, New Jersey: Prentice Hall.*, 1991.

[2] Jordan Holt and Jenq-Neng Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Transactions on Computers.*, vol. 42, no. 3, pp. 281–290, 1993.

[3] Ping Man Lam, Chi Sing Leung, and Tien Tsin Wong, "Noise-resistant fitting for spherical harmonics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 2, pp. 254–265, 2006.

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

[5] J.L. Bernier, J. Ortega, I. Rojas, E. Ros, and Prieto, "Obtaining fault tolerance multilayer perceptrons using an explicit regularization," *Neural Processing Letters*, vol. 12, no. 2, pp. 107–113, 2000.

[6] J.L. Bernier, J. Ortega, I. Rojas, E. Ros, and Prieto, "A quantitative study of fault tolerance, noise immunity and generalization ability of mlps," *Neural Computation*, vol. 12, pp. 2941–2964, 2000.

[7] J.L. Bernier, J. Ortega, I. Rojas, E. Ros, and Prieto, "Improving the tolerance of multilayer perceptrons by minimizing the statistical sensitivity to weight deviations," *Neurocomputing*, vol. 31, no. 1-4, pp. 87–103, 2000.

[8] J. L. Bernier, A. F. Diaz, F. J. Fernandez, A. Canas, J. Gonzalez, P. Martin-Smith, and J. Ortega, "Assessing the noise immunity and generalization of radial basis function networks," *Neural Processing Letters.*, vol. 18, no. 1, pp. 35–48, 2003.

[9] M. Stevenson M., R. Winter, and B. Widrow, "Sensitivity of feedfoward neural networks to weight errors," *IEEE Transactions on Neural Networks*, vol. 1, pp. 71–80, 1990.

[10] S.W. Piche, "The selection of weight accuracies for madalines," *IEEE Transactions on Neural Networks*, vol. 6, pp. 432–445, 1995.

[11] J.Y. Choi and C.H. Choi, "Sensitivity of multilayer perceptrons with differentiable activation functions," *IEEE Transactions on Neural Networks*, vol. 3, pp. 101–107, 1992.

[12] N.W. Townsend and L. Tarassenko, "Estimations of error bounds for neural network function approximators," *IEEE Transactions on Neural Networks*, vol. 10, pp. 217–230, 1999.

[13] M.A. Catala and X.L. Parra, "Fault tolerance parameter model of radial basis function networks," in *Proceedings of IEEE ICNN'96*, 1996, vol. 2, pp. 1384–1389.

[14] O. Fontenla-Romero *et al*, "A measure of fault tolerance for functional networks," *Neurocomputing*, vol. 63, pp. 327–347, 2004.

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

[16] D. Simon, "Distributed fault tolerance in optimal interpolative nets," *IEEE Transactions on Neural Networks*, vol. 12, no. 6, pp. 1348–1357, 2001.

[17] X. Parra and A. Catala, "Fault tolerance in the learning algorithm of radial basis function networks," in *Proceedings of IJCNN 2000*, 2000, vol. 3, pp. 527–532.

[18] J.E. Moody, "Note on generalization, regularization, and architecture selection in nonlinear learning systems," in *First IEEE-SP Workshop on Neural Networks for Signal Processing*, 1991.

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

[20] Yoh-Han Pao and Stephen M. Phillips, "The functional link net and learning optimal control," *Neurocomputing*, vol. 9, no. 2, pp. 149–164, 1995.

[21] S. Chen, X. Hong, C.J. Harris, and P.M. Sharkey, "Sparse modelling using orthogonal forward regression with press statistic and regularization," *IEEE Trans. Systems, Man and Cybernetics, Part B*, pp. 898–911, 2004.

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

[23] 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 Networks*, vol. 8, pp. 996–985, 1997.

[24] N. C. Steele and J. H. Tabor, "On parity problems and the functional-link artificial neural network," *Neural Computing and Applications*, vol. 2, pp. 205–208, 1994.

[25] Yoh-Han Pao and Yoshiyasu Takefuji, "Functional-link net computing: Theory, system architecture, and functionalities," *Neural Computing and Applications*, vol. 25, pp. 76–79–208, 1992.

[26] Chi Sing Leung, G.H. Young, John Sum, and W.K. Kan, "On the regularization of forgetting recursive least square," *IEEE Transactions on Neural Networks*, vol. 10, pp. 1482–1486, 1999.

[27] Chi Sing Leung, A.C. Tsoi, and L.W. Chan, "On the regularization of forgetting recursive least square," *IEEE Transactions on Neural Networks*, vol. 12, pp. 1314–1332, 2001.

[28] Yong Xu, K.W. Wong, and C.S. Leung, "Generalized rls approach to the training of neural networks," *IEEE Transactions on Neural Networks*, vol. 17, pp. 19–34, 2006.

[29] N. Murata, S. Yoshizawa, and S. Amari, "Network information criterion–determining the number of hidden units for an artificial neural network model," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 865–872, 1994.

[30] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, pp. 716–723, 1974.

[31] S. Chen, "Local regularization assisted orthogonal least squares regression," *Neurocomputing*, vol. 69, no. 4-6, pp. 559–585, 2006.

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

[33] Eric W. M. Lee, Chee Peng Lim, Richard K. K. Yuen, and S. M. Lo, "A hybrid neural network model for noisy data regression," *IEEE Trans. Systens, Man, and Cybernetics PART B*, vol. 34, pp. 951–960, 2004.

[34] N. McCullagh, *Generalised Linear Models*, Chapman and Hall, 1989.