Note on Weight Noise Injection During Training a MLP

Kevin Ho[1], **John Sum**[2]
[1]Department of Computer Science and Communication Engineering
Providence University, Sha-Lu, Taichung, Taiwan ROC.
[2]Institute of Technology and Innovation Management
National Chung Hsing University, Taichung, Taiwan ROC.

Abstract— **Although many analytical works have been done to investigate the change of prediction error of a trained NN if its weights are injected by noise, seldom of them has truly investigated on the dynamical properties (such as objective functions and convergence behavior) of injecting weight noise during training. In this paper, four different online weight noise injection training algorithms for multilayer perceptron (MLP) are analyzed and their objective functions are derived. Most importance, the objective function of injecting multiplicative weight noise during training is shown to be different from the prediction error of a trained MLP if its weights are injected by the same multiplicative weight noise. It provides a firm response to a question being posed for 14 years [8]:** *Can deterministic penalty terms model the effects of synaptic weight noise on network fault-tolerance?*. **Besides, we show that the objective function of injecting additive weight noise during training is equivalent to adding a regularizer penalizing the magnitude of the gradient vector of the MLP output with respect to its weight vector. Finally, the issue on their convergence proofs will be discussed.**

1 Introduction

Many methods have been developed throughout the last two decades to improve tolerance of a neural network towards random node fault, stuck-at node fault and weight noise. Known methods include injecting random or stuck-at node fault during training [20], [3], injecting (synaptic) weight noise during training (specially for multilayer perceptrons (MLP) [16], [17], a recurrent neural network (RNN) [12], or a pulse-coupled neural networks (PCNN) [9], injecting node noise (response variability) during training [2] (specifically for a model of PCNN) applying weight decay training [6], introducing network redundancy [19], formulating the training algorithm as a nonlinear constraint optimization problem [7], [18], bounding weight magnitude during training [5], [10], [13], and adding fault tolerant regularizer [4], [14], [21]. Amongst all, the weight/node noise-injection-based on-line training algorithms are of least theoretical studied [1], [2], [16], [17]. Especially, the objective functions and the convergence properties of these algorithms for multilayer perceptron (MLP) have not been derived and proved.

Murray & Edward in [8], [17] have derived the prediction error of a (trained) MLP if multiplicative weight noise is injected after training (see Section II.A and II.B in [17]). For the dynamics of the weight vector during training, only a qualitative analysis has been presented (see Section II.C in [17]). Convergence proof and objective function have not been analyzed. An has attempted to derive an objective function for this weight-noise injection training algorithm (see Section 4 in [1]). However, An has not succeeded to a convergence proof. The objective function derived is not true objective function for *training with weight noise injection*. It is again the prediction error of a trained MLP if weight noise is injected after training. Until recently, Ho *et al* [11] have showed that the convergence of output weight noise injection-based training a radial basis function (RBF) network is almost sure. For MLP, the convergence proof is still missing.

In this regard, the objective functions of online weight noise injection training algorithms for MLP are derived in this paper. The next section will introduce four different online weight noise injection training algorithms. Their corresponding objective function will be presented in Section 3. The objective functions for pure multiplicative weight noise injection and pure additive weight noise injection during training will be derived in Section 4. The issue on their convergence behavior will be discussed in Section 5. Section 6 will present the conclusion.

2 Weight noise injection during training

Let $\mathbf{f}(\cdot,\cdot) \in R^l$ be a single output multilayer perceptron (MLP) consisting of m hidden nodes, n input nodes and l linear output nodes.

$$\mathbf{f}(\mathbf{x},\mathbf{w}) = \mathbf{D}^T\mathbf{z}(\mathbf{A}^T\mathbf{x}+\mathbf{c}), \tag{1}$$

Corresponding author: John Sum (pfsum@nchu.edu.tw).

where $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_l] \in R^{m\times l}$ is the hidden to output weight vector, $\mathbf{z} = (z_1, z_2, \cdots, z_m)^T \in R^m$ is the output of the hidden nodes, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_m] \in R^{n\times m}$ is the input to hidden weight matrix, $\mathbf{a}_i \in R^n$ is the input weight vector of the i^{th} hidden node and $\mathbf{c} \in R^m$ is the input to hidden bias vector.

$\mathbf{w}$ in (1) is a vector augmenting all the parameters, i.e.

$$\mathbf{w} = (\mathbf{d}_1^T, \mathbf{d}_2^T, \cdots, \mathbf{d}_l^T, \mathbf{a}_1^T, \mathbf{a}_2^T, \cdots, \mathbf{a}_m^T, \mathbf{c}^T)^T.$$

For $i = 1, 2, \cdots, m$, $z_i(\mathbf{x}, \mathbf{a}_i, c_i) = \tau(\mathbf{a}_i^T\mathbf{x} + c_i)$, where $\tau(\cdot)$ is the neuronal transfer function. Training dataset is denoted by $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^N$. The random noise vector is denoted by $\mathbf{b}$.

For simplicity, we assume that there is only one output node, i.e. $l = 1$. In such case, the gradient of $f(\mathbf{x}, \mathbf{w})$ with respect to $\mathbf{w}$ is denoted by $g(\mathbf{x}_t, \mathbf{w}(t))$. The Hessian matrix of $f(\mathbf{x}, \mathbf{w})$ is denoted by $g_{\mathbf{w}}(\mathbf{x}_t, \mathbf{w}(t))$.

The online **weight noise injection** training for $f(\mathbf{x}, \mathbf{w})$ given a dataset $\mathcal{D}$ can be written as follows :

$$\begin{aligned}
\mathbf{w}(t+1) &= \mathbf{w}(t) + \mu_t(y_t - f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)). && (2)\\
\tilde{\mathbf{w}}(t) &= \mathbf{w}(t) + \mathbf{b}\odot\mathbf{w}(t). \quad \text{(multi. noise)} && (3)\\
\tilde{\mathbf{w}}(t) &= \mathbf{w}(t) + \mathbf{b}. \quad \text{(additive noise)} && (4)
\end{aligned}$$

Here $\mathbf{b}\odot\mathbf{w} = (b_1w_1, b_2w_2, \cdots, b_Mw_M)^T$ and b_i, for all i, is a mean zero Gaussian distribution with variance S_b.

For **simultaneous weight noise injection and weight decay**, the update equations are similar except the decay term is added.

$$\begin{aligned}
\mathbf{w}(t+1) &= \mathbf{w}(t) + \mu_t\{(y_t - f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t))\\
&\quad -\alpha\mathbf{w}(t)\}. && (5)\\
\tilde{\mathbf{w}}(t) &= \mathbf{w}(t) + \mathbf{b}\odot\mathbf{w}(t). \quad \text{(multi. noise)} && (6)\\
\tilde{\mathbf{w}}(t) &= \mathbf{w}(t) + \mathbf{b}. \quad \text{(additive noise)} && (7)
\end{aligned}$$

Clearly, the difference between pure noise injection during training, and the one with weight decay lies in the last term of the update equation, i.e. $-\alpha\mathbf{w}(t)$, which can limit the growth of $\|\mathbf{w}(t)\|$ to infinity.

3 Objective functions

In this section the mean update equations of the online weight noise injection training algorithms together with their corresponding objective functions will be presented. The derivation of those objective functions will be presented later in the subsequent sections.

3.1 Multiplicative weight noise

Let $\mathbf{g}(\mathbf{x}, \mathbf{w})$ and $\mathbf{g}_{\mathbf{w}}(\mathbf{x}, \mathbf{w})$ be the gradient and the Hessian matrix of $f(\mathbf{x}, \mathbf{w})$. For small S_b, one can assume that $\tilde{\mathbf{w}}$ is close to $\mathbf{w}$ and then apply Taylor expansion to $f(\cdot,\cdot)$ and $\mathbf{g}(\cdot,\cdot)$ and get that

$$\begin{aligned}
f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)) &\approx f(\mathbf{x}_t, \mathbf{w}(t)) + \mathbf{g}(\mathbf{x}_t, \mathbf{w}(t))^T\mathbf{b}\odot\mathbf{w}(t), && (8)\\
\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)) &\approx \mathbf{g}(\mathbf{x}_t, \mathbf{w}(t)) + \mathbf{g}_{\mathbf{w}}(\mathbf{x}_t, \mathbf{w}(t))\mathbf{b}\odot\mathbf{w}(t). && (9)
\end{aligned}$$

The mean update of Equation (2) together with Equation (3) will be given by

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t\mathbf{h}(\mathbf{w}(t)). \tag{10}$$

In which,

$$\mathbf{h}(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^N\int(y_k - f(\mathbf{x}_k, \tilde{\mathbf{w}}))\mathbf{g}(\mathbf{x}_k, \tilde{\mathbf{w}})P(\tilde{\mathbf{w}})d\tilde{\mathbf{w}}, \tag{11}$$

where $P(\tilde{\mathbf{w}})$ is defined as a Gaussian distribution with mean $\mathbf{w}$ and covariance matrix

$$\mathbf{diag}\{S_bw_1^2, S_bw_2^2, \cdots, S_bw_M^2\}.$$

Putting the above approximations (8) and (9) into (11), $\mathbf{h}(\mathbf{w}(t))$ in the mean update Equation (10) together with Equation (3) can be given by

$$\begin{aligned}
\mathbf{h}(\mathbf{w}) &= \frac{1}{N}\sum_{k=1}^N(y_k - f(\mathbf{x}_k, \mathbf{w}))\mathbf{g}(\mathbf{x}_k, \mathbf{w})\\
&\quad -\frac{S_b}{N}\sum_{k=1}^N\mathbf{g}_{\mathbf{w}}(\mathbf{x}_k, \mathbf{w})\mathbf{v}(\mathbf{x}_k, \mathbf{w}), && (12)
\end{aligned}$$

where

$$\mathbf{v}(\mathbf{x}_k, \mathbf{w}(t)) = \begin{bmatrix} w_1(t)^2g_1(\mathbf{x}_k, \mathbf{w}(t))\\ w_2(t)^2g_2(\mathbf{x}_k, \mathbf{w}(t))\\ \vdots\\ w_M(t)^2g_M(\mathbf{x}_k, \mathbf{w}(t))\end{bmatrix}. \tag{13}$$

It can be shown that the objective function being minimized by Equation (12) is given by

$$\begin{aligned}
\mathbf{V}(\mathbf{w}) &= \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^N(y_k - f(\mathbf{x}_k, \mathbf{w}))^2\right.\\
&\quad \left.+\frac{S_b}{N}\sum_{k=1}^N\sum_{j=1}^M w_j^2g_j(\mathbf{x}_k, \mathbf{w})^2\right\}\\
&\quad -\frac{S_b}{N}\sum_{k=1}^N\int_{\mathbf{w}_0}^{\mathbf{w}}\mathbf{u}(\mathbf{x}_k, \mathbf{r})\mathbf{dr}, && (14)
\end{aligned}$$

where

$$\mathbf{u}(\mathbf{x}_k, \mathbf{w}) = \begin{bmatrix} w_1g_1(\mathbf{x}_k, \mathbf{w})^2\\ w_2g_2(\mathbf{x}_k, \mathbf{w})^2\\ \vdots\\ w_Mg_M(\mathbf{x}_k, \mathbf{w})^2\end{bmatrix} \tag{15}$$

and the last term is the line integral taking any path from $\mathbf{w}_0$ to $\mathbf{w}$.

3.2 Additive weight noise

For the case that the injection weight noise is additive, $P(\tilde{\mathbf{w}})$ is defined as a Gaussian distribution with mean $\mathbf{0}$ and covariance matrix $\mathbf{diag}\{S_b, S_b, \cdots, S_b\}$. Apply Taylor expansion to $f(\cdot,\cdot)$ and $\mathbf{g}(\cdot,\cdot)$ and get that

$$\begin{aligned} f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)) &\approx f(\mathbf{x}_t, \mathbf{w}(t)) + \mathbf{g}(\mathbf{x}_t, \mathbf{w}(t))^T \mathbf{b}, \quad (16) \\ \mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)) &\approx \mathbf{g}(\mathbf{x}_t, \mathbf{w}(t)) + \mathbf{g}_{\mathbf{w}}(\mathbf{x}_t, \mathbf{w}(t))\mathbf{b}. \quad (17) \end{aligned}$$

Thus, the mean update equation can be expressed as follows :

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t \mathbf{h}(\mathbf{w}(t)), \quad (18)$$

where

$$\begin{aligned} \mathbf{h}(\mathbf{w}(t)) &= \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \mathbf{w}(t)))\mathbf{g}(\mathbf{x}_k, \mathbf{w}(t)) \\ &\quad -\frac{S_b}{N}\sum_{k=1}^{N}\mathbf{g}_{\mathbf{w}}(\mathbf{x}_k, \mathbf{w}(t))\mathbf{g}(\mathbf{x}_k, \mathbf{w}(t)). \quad (19) \end{aligned}$$

Similarly, it can be shown that the objective function being minimized by Equation (19) is given by

$$\begin{aligned} \mathbf{V}(\mathbf{w}) &= \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \mathbf{w}))^2 \right. \\ &\quad \left. +\frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} g_j(\mathbf{x}_k, \mathbf{w})^2\right\}. \quad (20) \end{aligned}$$

3.3 Multiplicative weight noise with weight decay

The mean update equation can be expressed as follows :

$$\begin{aligned} &\mathbf{w}(t+1) \\ &= \mathbf{w}(t) + \mu_t \left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \mathbf{w}(t)))\mathbf{g}(\mathbf{x}_k, \mathbf{w}(t))\right. \\ &\quad -\frac{S_b}{N}\sum_{k=1}^{N}\mathbf{g}_{\mathbf{w}}(\mathbf{x}_k, \mathbf{w}(t))\mathbf{v}(\mathbf{x}_k, \mathbf{w}(t)) \\ &\quad \left. -\alpha\mathbf{w}(t)\right\}. \quad (21) \end{aligned}$$

Similarly, it can be shown that the objective function being minimized by Equation (21) is given by

$$\begin{aligned} \mathbf{V}(\mathbf{w}) &= \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \mathbf{w}))^2 \right. \\ &\quad \left. +\frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} w_j^2 g_j(\mathbf{x}_k, \mathbf{w})^2\right\} \\ &\quad -\frac{S_b}{N}\sum_{k=1}^{N}\int_{\mathbf{w}_0}^{\mathbf{w}}\mathbf{u}(\mathbf{x}_k, \mathbf{r})\mathbf{dr} + \frac{\alpha}{2}\|\mathbf{w}\|^2. \quad (22) \end{aligned}$$

where $\mathbf{u}(\mathbf{x}_k, \mathbf{w})$ is given by Equation (15) and the last second term is the line integral taking any path from $\mathbf{w}_0$ to $\mathbf{w}$.

3.4 Additive weight noise with weight decay

The mean update equation can be expressed as follows :

$$\begin{aligned} &\mathbf{w}(t+1) \\ &= \mathbf{w}(t) + \mu_t \left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \mathbf{w}(t)))\mathbf{g}(\mathbf{x}_k, \mathbf{w}(t))\right. \\ &\quad -\frac{S_b}{N}\sum_{k=1}^{N}\mathbf{g}_{\mathbf{w}}(\mathbf{x}_k, \mathbf{w}(t))\mathbf{g}(\mathbf{x}_k, \mathbf{w}(t)) \\ &\quad \left. -\alpha\mathbf{w}(t)\right\}. \quad (23) \end{aligned}$$

Similarly, it can be shown that the objective function being minimized by Equation (23) is given by

$$\begin{aligned} \mathbf{V}(\mathbf{w}) &= \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k, \mathbf{w}))^2 \right. \\ &\quad \left. + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} g_j(\mathbf{x}_k, \mathbf{w})^2 + \alpha\|\mathbf{w}\|^2\right\} \quad (24) \end{aligned}$$

4 Derivations of the objective functions

In this section, the derivations of the objective functions will be presented. Detail steps will only be presented for the cases of injecting multiplicative weight noise and injecting additive weight noise. For the cases of combining injecting weight noise and weight decay, their derivations are just a simple extension of the first two cases.

4.1 Derivation of Equation (14)

By the fact that the i^{th} row of $\mathbf{g}_w$ is equal to $\frac{\partial g_i}{\partial \mathbf{w}}^T$ and thus the i^{th} element of $\mathbf{h}(\mathbf{w}(t))$ in Equation (11) is

given by

$$\begin{aligned} & h_i(\mathbf{w}) \\ = & \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))g_i(\mathbf{x}_k,\mathbf{w}) \\ & -\frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} w_j^2 g_j(\mathbf{x}_k,\mathbf{w})\frac{\partial g_i(\mathbf{x}_k,\mathbf{w})}{\partial w_j}, \quad (25) \end{aligned}$$

the following equations can be obtained.

$$\begin{aligned} & (y_k - f(\mathbf{x}_k,\mathbf{w}))g_i(\mathbf{x}_k,\mathbf{w}) \\ = & -\frac{1}{2}\frac{\partial}{\partial w_i}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2. \quad (26) \\ & \sum_{j=1}^{M} w_j^2 g_j(\mathbf{x}_k,\mathbf{w})\frac{\partial}{\partial w_j}g_i(\mathbf{x}_k,\mathbf{w}) \\ = & \frac{1}{2}\frac{\partial}{\partial w_i}\sum_{j=1}^{M} w_j^2 g_j(\mathbf{x}_k,\mathbf{w})^2 \\ & -w_i g_i(\mathbf{x}_k,\mathbf{w})^2. \quad (27) \end{aligned}$$

Using Equation (26) and Equation (27), Equation (12) can be rewritten as follows :

$$\mathbf{h}(\mathbf{w}(t)) = -\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}(t)) + \frac{S_b}{N}\sum_{k=1}^{N}\mathbf{u}(\mathbf{x}_k,\mathbf{w}(t)), \quad (28)$$

where

$$\begin{aligned} \mathcal{L}_1(\mathbf{w}) = & \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2\right. \\ & \left. + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} w_j^2 g_j(\mathbf{x}_k,\mathbf{w})^2\right\} \quad (29) \end{aligned}$$

and $\mathbf{u}(\mathbf{x}_k,\mathbf{w})$ in Equation (28) is given by Equation (15). As a result, the mean update equation (10) with additive weight noise injection can be expressed as follows :

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \mu_t\frac{\partial}{\partial \mathbf{w}}\mathcal{L}_1(\mathbf{w}(t)) + \mu_t\frac{S_b}{N}\sum_{k=1}^{N}\mathbf{u}(\mathbf{x}_k,\mathbf{w}(t)). \quad (30)$$

This recursive algorithm is identical to the gradient descent algorithm which minimizes the objective function given by Equation (14).

One should note that $\mathcal{L}_1(\mathbf{w})$ in Equation (29) is basically the objective function derived in [1], [4], [8] and [17]. It is the prediction error of a trained MLP if its weight is injected by multiplicative weight noise.

4.2 Derivation of Equation (20)

Again, by the same fact that the i^{th} row of $\mathbf{g}_w$ is equal to $\frac{\partial g_i}{\partial \mathbf{w}}^T$ and thus the i^{th} element of $\mathbf{h}(\mathbf{w}(t))$ in Equation (19) is given by

$$\begin{aligned} & h_i(\mathbf{w}(t)) \\ = & \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}(t)))g_i(\mathbf{x}_k,\mathbf{w}(t)) \\ - & \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} g_j(\mathbf{x}_k,\mathbf{w}(t))\frac{\partial}{\partial w_j}g_i(\mathbf{x}_k,\mathbf{w}(t)), \quad (31) \end{aligned}$$

the following equation can be obtained.

$$\begin{aligned} & \sum_{j=1}^{M} g_j(\mathbf{x}_k,\mathbf{w}(t))\frac{\partial}{\partial w_j}g_i(\mathbf{x}_k,\mathbf{w}(t)) \\ = & \frac{1}{2}\frac{\partial}{\partial w_i}\sum_{j=1}^{M} w_j(t)^2 g_j(\mathbf{x}_k,\mathbf{w}(t))^2. \quad (32) \end{aligned}$$

Using Equation (26) and Equation (32), Equation (12) can be rewritten as follows :

$$\mathbf{h}(\mathbf{w}(t)) = -\frac{\partial}{\partial \mathbf{w}}\mathcal{L}(\mathbf{w}(t)), \quad (33)$$

where

$$\begin{aligned} \mathcal{L}_2(\mathbf{w}) = & \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2\right. \\ & \left. + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} g_j(\mathbf{x}_k,\mathbf{w})^2\right\}. \quad (34) \end{aligned}$$

As a result, the mean update equation (10) with additive weight noise injection can be expressed as follows :

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \mu_t\frac{\partial}{\partial \mathbf{w}}\mathcal{L}_2(\mathbf{w}(t)). \quad (35)$$

This recursive algorithm is identical to the gradient descent algorithm which minimizes the objective function given by Equation (20). The second term in Equation (34) acts like a regularizer penalizing the magnitude of the gradient vector $\frac{\partial f}{\partial \mathbf{w}}$.

4.3 Derivation of Equation (22)

The derivation of Equation (22) is accomplished by the following equality.

$$\begin{aligned}
&\frac{1}{2}\frac{\partial}{\partial \mathbf{w}}\left\{\mathcal{L}_1(\mathbf{w}) - \frac{S_b}{N}\sum_{k=1}^{N}\int_{\mathbf{w}_0}^{\mathbf{w}} \mathbf{u}(\mathbf{x}_k,\mathbf{r})\mathbf{dr} + \alpha\|\mathbf{w}\|^2\right\} \\
= \;& \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}(t)))\mathbf{g}(\mathbf{x}_k,\mathbf{w}(t)) \\
& -\frac{S_b}{N}\sum_{k=1}^{N}\mathbf{g}_{\mathbf{w}}(\mathbf{x}_k,\mathbf{w}(t))\mathbf{v}(\mathbf{x}_k,\mathbf{w}(t)) + \alpha\mathbf{w}(t). \qquad (36)
\end{aligned}$$

4.4 Derivation of Equation (24)

The derivation of Equation (24) is accomplished by the following equality.

$$\begin{aligned}
&\frac{1}{2}\frac{\partial}{\partial \mathbf{w}}\left\{\mathcal{L}_2(\mathbf{w}) + \alpha\|\mathbf{w}\|^2\right\} \\
= \;& \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}(t)))\mathbf{g}(\mathbf{x}_k,\mathbf{w}(t)) \\
& +\alpha\mathbf{w}(t). \qquad (37)
\end{aligned}$$

5 Convergence issue

Even though we are able to derive the objective function for online **multiplicative weight noise injection training**, it is not sufficient to prove the convergence of the learning algorithm based on Equation (2) and Equation (3). It is because the lower bound of Equation (14) has not been founded. Therefore, the convergence of this training algorithm cannot be claimed. As a matter of fact, we have identified a few counter examples showing the divergence of training MLP with online multiplicative weight noise injection. Interesting readers can refer to [22].

For the algorithms based on injecting additive weight noise, (2) and (4), and adding weight decay during training (5), simulation results have demonstrated that their weight vectors are able to converge [23]. Their convergence proofs are underway and will be reported in the future paper. The major idea of proof is similar (but not identical to) stochastic gradient descent. For a neural network that is injected by input noise, stochastic gradient descent can be applied right the way. For injecting weight noise, we need to apply the ODE method [15].

6 Conclusion

In our recent works, we have derived the objective functions for some online noise/fault injection training algorithms for radial basis function networks [11]. In this paper, we follow the same direction and derived the true objective functions for online weight noise injection training algorithms for multilayer perceptron. By analysis on the actual update equations, we have derived the true objective functions for four online weight noise injecting training algorithms, including (1) multiplicative weight noise injection, (2) additive weight noise injection, (3) simultaneous multiplicative weight noise injection and weight decay, (4) simultaneous additive weight noise injection and weight decay.

In accordance with these objective function, we have found that the objective function of injecting multiplicative weight noise during training a MLP is different from the prediction error of a trained MLP if its weights are injected by the same multiplicative weight noise. This result provides a clue responding to a question that have been posed for 14 years [8]: *Can deterministic penalty terms model the effects of synaptic weight noise on network fault-tolerance?*. For injecting additive weight noise during training a MLP, the objective function has also been derived and we have found that it is similar to adding a simple regularizer that penalizes the magnitude of the gradient of the network output with respect to the weight vector.

Even though we have derived the objective functions for these weight noise injection based training algorithms, their convergence (divergence) proofs are still underway and will be reported in future papers. We should point out that this possible divergence behavior could be a big problem in biological learning, as our brain is full of **intrinsic synaptic weight noises**. Those noises are not removable. Further investigation along this line would be another valuable future work.

Acknowledgement

The research work reported in this paper is supported in part by Taiwan National Science Council (NSC) Research Grant 97-2221-E-005-050 and 98-2221-E-005-048.

References

[1] An G. The effects of adding noise during backpropagation training on a generalization performance, *Neural Computation*, Vol.8, 643-674, 1996.

[2] Basalyga G. and E. Salinas, When response variability increases neural network robustness to synaptic noise, *Neural Computation*, Vol.18, 1349-1379, 2006.

[3] Bolt G., *Fault tolerant in multi-layer Perceptrons*. PhD Thesis, University of York, UK, 1992.

[4] Bernier J.L. *et al*, Obtaining fault tolerance multilayer perceptrons using an explicit regularization, *Neural Processing Letters*, Vol.12, 107-113, 2000.

[5] Cavalieri S. and O. Mirabella, A novel learning algorithm

which improves the partial fault tolerance of multilayer NNs, *Neural Networks*, Vol.12, 91-106, 1999.

[6] Chiu C.T. *et al.*, Modifying training algorithms for improved fault tolerance, ICNN'94 Vol.I, 333-338, 1994.

[7] Deodhare D., M. Vidyasagar and S. Sathiya Keerthi, Synthesis of fault-tolerant feedforward neural networks using minimax optimization, *IEEE Transactions on Neural Networks*, Vol.9(5), 891-900, 1998.

[8] Edwards P.J. and A.F. Murray, Can deterministic penalty terms model the effects of synaptic weight noise on network fault-tolerance? *International Journal of Neural Systems*, 6(4):401-16, 1995.

[9] Edwards P.J. and A.F. Murray, Fault tolerant via weight noise in analog VLSI implementations of MLP's – A case study with EPSILON, *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol.45, No.9, p.1255-1262, Sep 1998.

[10] Hammadi N.C. and I. Hideo, A learning algorithm for fault tolerant feedforward neural networks, *IEICE Transactions on Information & Systems*, Vol. E80-D, No.1, 1997.

[11] Ho K., C.S. Leung, and J. Sum, On weight-noise-injection training, M.Koeppen, N.Kasabov and G.Coghill (Eds.), *Advances in Neuro-Information Processing*, Springer LNCS 5507, pp. 919V926, 2009.

[12] Jim K.C., C.L. Giles and B.G. Horne, An analysis of noise in recurrent neural networks: Convergence and generalization, *IEEE Transactions on Neural Networks*, Vol.7, 1424-1438, 1996.

[13] Kamiura N., *et al*, On a weight limit approach for enhancing fault tolerance of feedforward neural networks, *IEICE Transactions on Information & Systems*, Vol. E83-D, No.11, 2000.

[14] Leung C.S., J. Sum, A fault tolerant regularizer for RBF networks, *IEEE Transactions on Neural Networks*, Vol. 19 (3), pp.493-507, 2008.

[15] Ljung L., Analysis of recursive stochastic algorithms, *IEEE Transactions on Automatic Control*, Vol.AC-22, No.4, pp.551-575, August, 1977.

[16] Murray A.F. and P.J. Edwards, Synaptic weight noise during multilayer perceptron training: fault tolerance and training improvements, *IEEE Transactions on Neural Networks*, Vol.4(4), 722-725, 1993.

[17] Murray A.F. and P.J. Edwards, Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training, *IEEE Transactions on Neural Networks*, Vol.5(5), 792-802, 1994.

[18] Neti C. M.H. Schneider and E.D. Young, Maximally fault tolerance neural networks, *IEEE Transactions on Neural Networks*, Vol.3(1), 14-23, 1992.

[19] Phatak D.S. and I. Koren, Complete and partial fault tolerance of feedforward neural nets., *IEEE Transactions on Neural Networks*, Vol.6, 446-456, 1995.

[20] Sequin C.H. and R.D. Clay, Fault tolerance in feedforward artificial neural networks, *Neural Networks*, Vol.4, 111-141, 1991.

[21] Sum J., C.S. Leung and K. Ho, On objective function, regularizer and prediction error of a learning algorithm for dealing with multiplicative weight noise, *IEEE Transactions on Neural Networks* Vol.20(1), Jan, 2009.

[22] Sum J., K. Ho and C.S. Leung, Further Analysis on Online Multiplicative Weight Noise Injection Training for MLP, in submission.

[23] Sum J., K. Ho, SNIWD: Simultaneous weight/node noise injection with weight decay for MLP training, in submission.