Brief survey on online injecting fault/noise-based fault tolerant learning algorithms

John Sum[1], Kevin Ho[2], Jaromir Horejsi[3]
[1]Institute of E-Commerce, National Chung Hsing University, Taichung, Taiwan.
[2]Department of Computer Science and Communication Engineering
Providence University, Sha-Lu, Taiwan ROC
[3]Department of Computer Science and Engineering
Czech Technical University in Prague, Czech Republic.

January 6, 2009

Abstract

While injecting noise (input noise or weight noise) or faults (weight fault or node fault) during training has been applied to improve fault tolerance of a neural network, not much analysis has been done to reveal the success of such learning methods. In this paper, a list of eight fault-injection-based on-line learning algorithms will be described. Potential research problems alongside will be introduced.

Keywords : Gradient Descent, Fault Tolerance, KL Divergence, Learning Theory, Neural Networks

1 Introduction

In conventional learning theory, neural networks (NN) are trained to achieve good generalization. Which could be accomplished by adding regularizer [27, 26, 35, 36, 49] or pruning [19, 25, 28, 26, 44, 41], so as to reduce the weights' magnitudes or model complexity. These methods work well under the assumption that the neural network after training can be ideally implemented (i.e. fault-free implementation). It is true if a neural network is hard-coded in a program that is running in a computer with very high precision data representation. However, it is not true for electronic implementations, like FPGA [20]. Component failure, sign bit change, open circuit [42], finite precision [46] and even exposure to radiation [60] could degrade the performance of such an implementation drastically. In such case, the performance of a neural network will be questionable even if it has been trained to achieve very good generalization.

In this regard, many methods have been developed throughout the last two decades in order to tackle such fault tolerant problem. One approach is to inject random fault or noise during training [47, 10], the other is introducing network redundancy [42], applying weight decay learning [12], formulating the training algorithm as a nonlinear constraint optimization problem [13, 39], bounding weight magnitude during training [11, 18, 22], and adding fault tolerant regularizer [5, 6, 8, 29, 56]. Please refer to [54, 57] for a survey in fault tolerant learning. Amongst all, only the technique of adding input noise during training has widely been studied. Analysis on injecting weight (or node) fault or weight (or node) noise during training are of little investigation. Extension of those fault (or noise) injection techniques together with weight decay will be even scarce.

Table 1 for a list of the research works related to on-line fault (noise)-injection-based learning algorithms. and apparent that only injecting additive input noise (or called jitter noise in some papers) has been widely analyzed, [1, 9, 16, 17, 45]. While An in [1] has attempted, by applying the theory developed by Bottou (Theorem 1 in [3]), to derive the objective function for injecting weight noise on-line learning algorithm, his results are not convincing. It is because he has not shown that the on-line update equation can fulfill the conditions stated in Bottou Theorem.

For other cases, the objective functions that are minimizing as well as their convergence properties have yet been revealed. Objective function is an important piece of information for knowing the solution of a learning algorithm, and by which the similarities and differences amongst different algorithms can be analyzed. Their

Table 1: Research works related to fault (or noise) injection-based learning algorithms.

Year/Ref.	Fault	NN	Description
1991 [10, 47]	Node fault	MLP	Injecting random node fault during BP training
1993 [37, 38]	Weight noise	MLP	Adding weight noise during BP training
1994 [12]	Weight noise	MLP	Apply weight decay algorithm with random node fault injection during training
1995 [9]	–	–	Analysis on injecting input noise
1995 [45]	–	–	Analysis on injecting input noise
1996 [1]	–	–	Analysis on injecting input noise Analysis on injecting weight noise
1997 [16, 17]	–	–	Analysis on injecting input noise
1997 [18]	Node fault	MLP	Weight magnitude bounding (Heuristic modification of BP)
1999 [11]	Node fault	MLP	Weight magnitude bounding (Heuristic modification of BP)
1999 [48]	Multinode fault	MLP	Constraint BP[1] (Heuristic modification of BP)
2000 [22]	Node fault	MLP	Weight magnitude bounding (Heuristic modification of BP)
2000 [40]	Multiplicative weight noise	RBF	Apply weight decay algo.
2000 [5, 7]	Multiplicative weight noise	MLP	Explicit regularization (Gradient descent algorithm)
2002 [43]	Single node fault	MLP	Nonlinear program[2] (Batch mode learning)
2004 [59]	Node fault	MLP	Apply penalty term[3] (Stochastic gradient descent)
2007 [55]	Multiplicative weight noise	RBF	Apply KL divergence (Stochastic gradient descent)
2008 [30]	Multiplicative weight noise	RBF	Analysis on the generalization error Apply weight decay
2008 [21]	–	RBF	Convergence proof of injecting multinode fault
2008 [58]	–	RBF	Convergence proof of injecting weight noise

[1] The objective function is $\min_\theta\{\max_{\tilde{\theta}} \max_k (y_k - f(x_k, \tilde{\theta}|\theta))^2\}$

[2] The objective function is $1/N \sum_{k=1}^{N} (y_k - f(x_k, \theta))^2 + \alpha|\Omega_{\tilde{\theta}}|^{-1} \sum_{\tilde{\theta}\in\Omega_{\tilde{\theta}}} 1/N \sum_{k=1}^{N} (y_k - f(x_k, \tilde{\theta}|\theta))^2$

[3] The objective function is $1/N \sum_{k=1}^{N} (y_k - f(x_k, \theta))^2 + \frac{\alpha}{n}|\theta|^n$

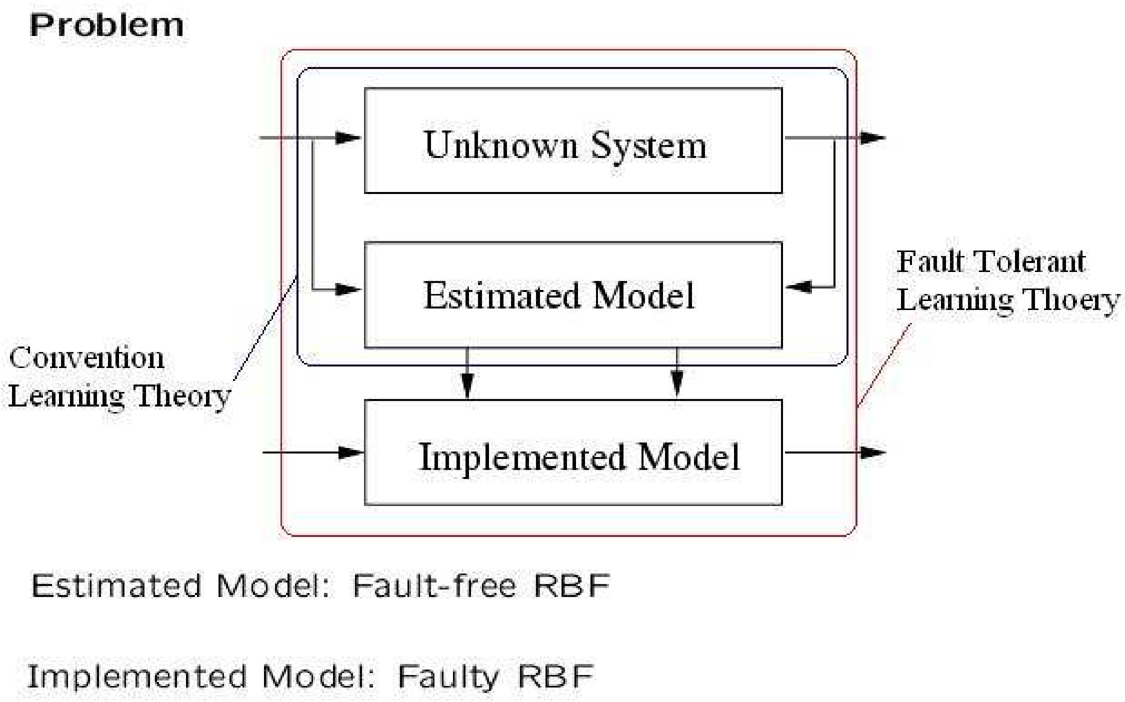

Figure 1: Conventional learning theory and fault tolerant learning theory.

prediction errors can be deduced. In this regards, a complete analysis on the properties of these algorithms is inevitable. In this paper, eight fault-injection-based on-line learning algorithms will briefly described. Four of them are pure fault-injection-based algorithms : (1) training with weight noise injection, (2) training with node noise injection, (3) training with weight fault injection, and (4) training with node fault injection. While the other four are algorithms that combine fault-injection together with weight decay : (1) training with weight noise injection plus weight decay, (2) training with node noise injection plus weight decay, (3) training with weight fault injection plus weight decay, and (4) training with node fault injection plus weight decay. Discussion on the difference between fault/noise injection learning and training based on the idea of noise immunity will be highlighted. Potential future works along the direction will be suggested.

2 Fault tolerant models

Let $\mathcal{M}_0$ be the unknown system to be modeled. The input and output of $\mathcal{M}_0$ are denoted by x and y respectively. The only information we know about $\mathcal{M}_0$ is a set of measurement data $\mathcal{D}$, where $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^{N}$. Making use of this data set, an estimated model $\hat{\mathcal{M}}$ that is *good* enough to capture the *general behavior* of the unknown system can be obtained. For many real-time applications, this *good* model $\hat{\mathcal{M}}$ will furthermore be mapped onto a hardware implementation, like FPGA or DSP chip. We denote the inaccurate implementation of $\hat{\mathcal{M}}$ by $\tilde{\mathcal{M}}$. The conceptual difference amongst $\mathcal{M}_0$, $\hat{\mathcal{M}}$ $\tilde{\mathcal{M}}$ is shown in Figure 1. Finally, we let Ω be the set of models in which $\hat{\mathcal{M}}$ and $\tilde{\mathcal{M}}$ are defined.

In conventional learning theory, it is assumed that the implementation of a model $\mathcal{M}_0$ is fault-free. Therefore $\tilde{\mathcal{M}}$ will be identical to $\hat{\mathcal{M}}$. In FTL, such assumption is not existed. An implementation of a model $\mathcal{M}_0$, denoted by $\tilde{\mathcal{M}}$, is a random model probabilistically depended on the model $\hat{\mathcal{M}}$. Let the set of models in which $\tilde{\mathcal{M}}$ can be defined is denoted by $\tilde{\Omega}_{\mathcal{M}}$, it is clear that $\tilde{\Omega}_{\mathcal{M}}$ must be a subset of Ω and the distribution of $\tilde{\mathcal{M}}$ is given by $P(\tilde{\mathcal{M}}|\hat{\mathcal{M}})$.

For $k = 1, 2, \cdots, N$

$$\mathcal{M}_0 \;:\quad y_k = f(x_k) + e_k, \tag{1}$$

where (x_k, y_k) is the k^{th} input-output pair that is measured from an unknown deterministic system $f(x)$ with random output noise e_k, $e_k \sim \mathcal{N}(0, S_e)$. To model the unknown system, we assume that $f(x)$ can be realized by an RBF network, i.e.

$$\mathcal{M} \;:\quad y_k = \sum_{i=1}^{M} \theta_i \phi_i(x_k) + e_k \tag{2}$$

for all $k = 1, 2, \cdots, N$ and $\phi_i(x)$ for all $i = 1, 2, \cdots, M$ are the radial basis functions given by

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

c_is are the radial basis function centers and the positive parameter $\sigma > 0$ controls the width of the radial basis functions. Thus, a model $\mathcal{M}$ in Ω is represented by an M-vector, $\theta = (\theta_1, \theta_2, \cdots, \theta_M)^T$ and the model set Ω will be isomorphic to R^M.

2.1 Multiplicative noise

Multiplicative noise exists whenever a weight value or a node output is encoded in a low precision binary form. Let $\beta = (\beta_1, \beta_2, \cdots, \beta_M)^T$ be the implementation of a model θ, denoted by $\tilde{\theta}$ is given by

$$\begin{aligned} \tilde{\theta} &= \theta + A_\theta \beta, && (4)\\ A_\theta &= \mathbf{diag}\{\theta_1, \theta_2, \cdots, \theta_M\}, && (5) \end{aligned}$$

where β_i for all $i = 1, 2, \cdots, M$ are independent mean zero Gaussian noise with variance S_β.

$$P(\beta_i) = \frac{1}{\sqrt{2\pi S_\beta}} \exp\left(-\frac{\beta_i^2}{2S_\beta}\right). \tag{6}$$

For the case that the output of a node is corrupted by multiplicative noise, the output of the i^{th} node $\tilde{\phi}_i$ will be given by

$$\begin{aligned} \tilde{\phi} &= \phi + A_\phi \beta, && (7)\\ A_\phi &= \mathbf{diag}\{\phi_1, \phi_2, \cdots, \phi_M\}, && (8) \end{aligned}$$

where β_i is defined as Equation (6).

2.2 Multinode/weight fault

We assume that a node, or a weight, fault is equivalent to permanently set the output of the node, or the value of a weight, zero. A faulty RBF, with $\hat{f}(x, \tilde{\theta})$, where $\tilde{\phi} = (\tilde{\phi}_1, \tilde{\phi}_2, \cdots, \tilde{\phi}_M)^T$ and

$$\begin{aligned} \tilde{\phi} &= \phi - A_\phi \beta, && (9)\\ A_\phi &= \mathbf{diag}\{\phi_1, \phi_2, \cdots, \phi_M\}. && (10) \end{aligned}$$

where $\beta_i = 1$ if the i^{th} node is normal and $\beta_i = 0$ if the i^{th} node is fault. We assume that all nodes are of equal fault rate p, i.e.

$$P(\beta_i) = \begin{cases} p & \text{if } \beta_i = 1 \\ 1 - p & \text{if } \beta_i = 0. \end{cases} \tag{11}$$

for $i = 1, 2, \cdots, M$, Besides, $\beta_1, \cdots, \beta_M$ are independent random variables.

For multiweight fault, the model is similar. With the same definition on the random vector β as Equation (11),

$$\begin{aligned} \tilde{\theta} &= \theta - A_\theta \beta, && (12)\\ A_\theta &= \mathbf{diag}\{\theta_1, \theta_2, \cdots, \theta_M\}, && (13) \end{aligned}$$

where $\beta_i = 1$ if the i^{th} node is normal and $\beta_i = 0$ if the i^{th} node is fault. Note that the model is also similar to the case of multiplicative weight noise. But their definitions on β_i are different.

3 Fault/Noise Injection-Based FT Learning

In conventional training by minimizing mean square errors, the update equation for $\theta(t)$ is given by

$$\theta(t+1) = \theta(t) + \mu_t(y_t - \phi^T(x_t)\theta(t))\phi(x_t), \tag{14}$$

where μ_t (for $t \geq 1$) is the step size at the t^{th} iteration. While in online fault/noise injection-bsed learning, the form of the update equation is similar to Equation (14) except that the $\theta(t)$ or $\phi(x_t)$ in the second term of the right hand side is replaced by the fault/noise injection form, denoted by $\tilde{\theta}(t)$ or $\tilde{\phi}(x_t)$.

3.1 Pure fault/noise injection

In this subsection, four different type of pure fault/noise injection-based FT learning algorithms will be summarized.

Weight noise injection: While a network is trained by the idea of weight noise injection, the update equation will be given by

$$\theta(t+1) = \theta(t) + \mu_t(y_t - \phi^T(x_t)\tilde{\theta}(t))\phi(x_t), \tag{15}$$

where μ_t is (for $t \geq 1$) the step size at the t^{th} iteration,

$$\tilde{\theta}_i(t) = \begin{cases} \theta_i(t) + \beta_i & \text{for additive noise injection,} \\ \theta_i(t) + \beta_i\theta_i(t) & \text{for multiplicative noise injection.} \end{cases} \tag{16}$$

β_i for all $i = 1, 2, \cdots, M$ are independent mean zero Gaussian noise with variance S_β. Normally, it is assumed that the value of S_β is small. Although, the theoretical proof presented later in this paper applies to any bounded value, it is meaningless to consider a large value of S_β.

Multiplicative node noise injection: For a RBF network that is trained by injecting multiplicative node noise, the update equation is given by

$$\begin{aligned} \theta(t+1) &= \theta(t) + \mu_t(y_t - \tilde{\phi}^T(x_t)\theta(t))\tilde{\phi}(x_t), & (17) \\ \tilde{\phi}_i &= (1+\beta_i)\phi_i, \quad \beta_i \sim \mathcal{N}(0, S_\beta), & (18) \end{aligned}$$

for all $i = 1, 2, \cdots, M$.

Multiweight fault injection: For a RBF network that is trained by injecting multiweight fault, the update equation is given by

$$\begin{aligned} \theta(t+1) &= \theta(t) + \mu_t(y_t - \sum_{i=1}^{M} \phi_i^T(x_t)(1-\beta_i)\theta_i(t))\phi(x_t), & (19) \\ P(\beta_i) &= \begin{cases} p & \text{if } \beta_i = 1 \\ (1-p) & \text{if } \beta_i = 0 \end{cases} & (20) \end{aligned}$$

for all $i = 1, 2, \cdots, M$.

Multinode fault injection: While an RBF network is trained by multinode fault injection, the update equation is given by

$$\begin{aligned} \theta(t+1) &= \theta(t) + \mu_t(y_t - \tilde{\phi}^T(x_t)\theta(t))\tilde{\phi}(x_t), & (21) \\ \tilde{\phi}_i &= (1-\beta_i)\phi_i. & (22) \end{aligned}$$

We assume that all nodes are of equal fault rate p, i.e. $P(\beta_i) = p$ if $\beta_i = 1$ and $P(\beta_i) = (1-p)$ of $\beta_i = 0$, for $i = 1, 2, \cdots, M$. Besides, $\beta_1, \cdots, \beta_M$ are independent random variables.

3.2 Fault/Noise injection plus weight decay

This type of training algorithms extends the idea of pure fault injection by adding a decay term, either $\lambda\theta$ or $\lambda\tilde{\theta}$ ($0 < \lambda \ll 1$), in the update equation. The four counter algorithms, extended from the *pure fault/noise injection-based algorithms* will be described in the subsequent paragraphs.

Weight noise injection: While a network is trained by the idea of weight noise injection together with weight decay, the update equation will be given by

$$\theta(t+1) = \theta(t) + \mu_t \left\{ (y_t - \phi^T(x_t)\tilde{\theta}(t))\phi(x_t) - \lambda\tilde{\theta} \right\}, \tag{23}$$

where μ_t is (for $t \geq 1$) the step size at the t^{th} iteration,

$$\tilde{\theta}_i(t) = \begin{cases} \theta_i(t) + \beta_i & \text{for additive noise injection,} \\ \theta_i(t) + \beta_i\theta_i(t) & \text{for multiplicative noise injection.} \end{cases} \tag{24}$$

β_i for all $i = 1, 2, \cdots, M$ are independent mean zero Gaussian noise with variance S_β.

Multiplicative node noise injection: For a RBF network that is trained by multiplicative node noise injection together with weight decay, the update equation is given by

$$\theta(t+1) = \theta(t) + \mu_t \left\{ (y_t - \tilde{\phi}^T(x_t)\theta(t))\tilde{\phi}(x_t) - \lambda\theta(t) \right\}, \tag{25}$$

$$\tilde{\phi}_i = (1 + \beta_i)\phi_i, \quad \beta_i \sim \mathcal{N}(0, S_\beta), \tag{26}$$

for all $i = 1, 2, \cdots, M$.

Multiweight fault injection: For a RBF network that is trained by injecting multiweight fault together with weight decay, the update equation is given by

$$\theta(t+1) = \theta(t) + \mu_t \left\{ (y_t - \sum_{i=1}^{M} \phi_i^T(x_t)(1 - \beta_i)\theta_i(t))\phi(x_t) - \lambda\tilde{\theta} \right\}, \tag{27}$$

$$P(\beta_i) = \begin{cases} p & \text{if } \beta_i = 1 \\ (1-p) & \text{if } \beta_i = 0 \end{cases} \tag{28}$$

for all $i = 1, 2, \cdots, M$.

Multinode fault injection: For a RBF network, $f(x_t, \theta(t)) = \phi(x_t)^T\theta(t)$, that is trained by injecting multinode fault during weight decay learning,

$$\theta(t+1) = \theta(t) + \mu_t \left\{ (y_t - \tilde{\phi}^T(x_t)\theta(t))\tilde{\phi}(x_t) - \lambda\tilde{\theta}(t) \right\}, \tag{29}$$

$$\tilde{\phi}_i = (1 - \beta_i)\phi_i, \tag{30}$$

for all $i = 1, \cdots, M$. $P(\beta_i) = p$ if $\beta_i = 1$ and $P(\beta_i) = (1-p)$ of $\beta_i = 0$.

4 Discussions

4.1 Fault injection versus noise immunity

In some recent studies on fault tolerance, researchers based on the idea of noise immunity [6, 8, 29, 56]. It should be noted that the generic idea applied to develop learning algorihtms based on noise immunity and fault injection is quite different. The former refers to a *property* that will happen when a neural network is implemented. It can equally be considered as adding noise to a network after it has been well-trained. The latter refers to an on-line training *technique* for improving fault tolerance. In which, noise is added during training.

Therefore, an objective function derived from the sense of noise immunity reflects the error sensitivity of a network if its weights or nodes are perturbed, e.g. [9, 45]. Let $\mathcal{L}_{WN}(\theta)$ and $\mathcal{L}_{NF}(\theta)$ be the objective functions for weight noise immunity and node fault immunity respectively.

$$\mathcal{L}_{WN}(\theta) = \frac{1}{N}\sum_{k=1}^{N}\int (y_k - \phi^T(x_k)(\theta + \Delta\theta))^2 P(\Delta\theta) d\Delta\theta. \tag{31}$$

$$\mathcal{L}_{NF}(\theta) = \frac{1}{N}\sum_{k=1}^{N}\int (y_k - (\phi(x_k) + \Delta\phi(x_k))^T\theta)^2 P(\Delta\phi(x_k)) d\Delta\phi(x_k). \tag{32}$$

Here $P(\Delta\theta)$ and $P(\Delta\phi(x_k))$ correspond to the proabability density functions of the noise corrupted θ and the faulty ϕ. They are depended on the fault/noise model defined. Since the factor inside the summation is of second order form, extra term will be introduced once the integration has been taken. It can readily be shown that the objective functions can be written as follows [5, 56] [29]:

$$\mathcal{L}_{WN}(\theta) = \frac{1}{N}\sum_{k=1}^{N}(y_k - \phi^T(x_k)\theta)^2 + \theta^T \mathbf{R}_{WN}\theta. \tag{33}$$

$$\mathcal{L}_{NF}(\theta) = \frac{1}{N}\sum_{k=1}^{N}\int (y_k - \phi^T(x_k)\theta)^2 + \theta^T \mathbf{R}_{NF}\theta. \tag{34}$$

The regularization matrices $\mathbf{R}_{WN}$ and $\mathbf{R}_{NF}$ are defined in terms of

$$\frac{1}{N}\sum_{k=1}^{N}\phi(x_k)\phi^T(x_k), \quad \text{and} \quad \mathbf{diag}\left\{\frac{1}{N}\sum_{k=1}^{N}\phi_1^2(x_k), \frac{1}{N}\sum_{k=1}^{N}\phi_2^2(x_k)\cdots, \frac{1}{N}\sum_{k=1}^{N}\phi_M^2(x_k)\right\}.$$

Finally, fault tolerant learning algorithms are developed by apply gradient descent in accordance with these objective functions. Clearly, these objective functions are basically equivalent to regularization learning. The extra term play a role as a regularizer controlling the weight magnitudes.

For on-line fault/noise injection-based learning algorithms, their development are solely based on heuristic modification of the original MSE based learning algorithm. At the time the algorithms firstly proposed, objective functions were unknown. Except in certain cases (like adding input noise and injecting multinode fault) their objective functions are proved to the same [9, 29, 45] as its immunity based counterpart. For other cases, their actual objective functions are still yet to be uncovered.

4.2 Stochastic approximation

Theory of stochastic approximation has been developed for more than half a century for the analysis of recursive algorithms. Advanced theoretical works for complicated recursive algorithms have still been under investigation [24]. The theorem applied to the proof could be based on Gladyshev Theorem [15]. Variant forms of the theorem can also be found in the Section II of Chapter 9 in [34] and the theorem stated in [33].

Let $\theta(t)$ and $M(\theta(t), \omega(t))$ for all $t = 0, 1, 2,$ and so on be m-vectors. $\omega(t)$ for all $t = 0, 1, 2,$ and so on are i.i.d. random vectors with probability density function $P(\omega)$ Consider a recursive algorithm defined as follows :

$$\theta(t+1) = \theta(t) - \mu_t M(\theta(t), \omega(t)). \tag{35}$$

In which, the expectation of $M(\theta, \omega)$ over ω, i.e.

$$\bar{M}(\theta) = \int M(\theta, \omega) P(\omega) d\omega, \tag{36}$$

has unique solution θ^* such that $\bar{M}(\theta^*) = 0$. Gladyshev Theorem states the conditions that $\theta(t)$ obtained by the Equation (35) can converge to θ^* as $t \to \infty$.

Table 2: Potential theoretical research problems.

Algorithms	RBF	MLP
Weight Noise	Done [21]	In progress [21]
Weight Fault	In progress	–
Node Noise	In progress	–
Node Fault	Done [58]	–
Weight Noise with WD	Done [30]	–
Weight Fault with WD	In progress	–
Node Noise with WD	In progress	–
Node Fault with WD	Done [30]	–

Theorem 1 (Gladyshev Theorem [15]) *Suppose $\bar{M}(\theta)$ has unique solution at θ^*, i.e. $\bar{M}(\theta^*) = 0$ and there exists positive constants κ_1 and κ_2 such that the following conditions are satisfied :*

(C1) $\mu_t \geq 0$, $\sum_t \mu_t = \infty$ *and* $\sum_t \mu_t^2 < \infty$.

(C2) $\inf_{\varepsilon < \|\theta - \theta^*\| < \varepsilon^{-1}} (\theta - \theta^*)^T \bar{M}(\theta) > 0$, *for all* $\varepsilon > 0$.

(C3) $\int \|M(\theta, \omega)\|^2 P(\omega) d\omega \leq \kappa_1 + \kappa_2 \|\theta\|^2$.

Then for $t \to \infty$, $\theta(t)$ obtained by Equation (35) converges to θ^* with probability one.

Normally, the first coniditon can easily be satisfied. It is because the step size μ_t could be defined as

$$\mu_t = \frac{\text{const.}}{t} \quad \text{for all } t \geq 1.$$

Therefore, the proof of Condition (C1) can be skipped. The core of the proof will be on the Condition (C2) and Condition (C3).

5 Future Works

While the convergence analysis on the *pure fault injection-based* learning algorithms can be done by applying the Galdyshev Theorems, many open problems are still yet to be solved. In addition to the literature survey provided in the earlier sections, some specific problems especially those *fault injection plus weight decay* together to fault tolerant learning are with particular valuable for futher investigation.

Convergence properties: What is the convergence properties of injecting fault (or noise) to weight (or node) during normal training, and during weight decay training ?

Objective functions: What is the convergence properties of injecting fault (or noise) to weight (or node) during normal training, and during weight decay training ?

Prediction errors: What is the prediction error of an RBF that is trained by injecting fault (or noise) to weight (or node) during normal training, and during weight decay training ?

Connection to biological learning: What is the connection between fault-injection-based learning and biological learning in our brain ?

Table 2 summarizes a list of potential research problems in regard to the theoretical aspects of on-line fault/noise injection-based fault tolerant learning. Clearly, theoretical works for multilayer perception (MLP) are still open for further investigation.

References

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

[2] A T. and Y. Grandvalet, Adaptive noise injection for input variables relevance determination, *IEEE Transactions on Neural Networks*, 1997.

[3] Bottou L., Stochastic gradient learning in neural networks, *NEURO-NIMES'91*, EC2, Nanterre, France, 687-706, 1991.

[4] Bottou L., Online learning and stochastic approximations, in *Online Learning in Neural Networks*, David Saad (Ed), pp. 9-42, Cambridge University Press, 1999.

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

[6] Bernier J.L. *et al*, A quantitative study of fault tolerance, noise immunity and generalization ability of MLPs, *Neural Computation*, Vol.12, 2941-2964, 2000.

[7] Bernier J.L. *et al*, Improving the tolerance of multilayer perceptrons by minimizing the statistical sensitivity to weight deviations, *Neurocomputing*, Vol.31, 87-103, 2000.

[8] Bernier J.L. *et al*, Assessing the noise immunity and generalization of radial basis function networks, *Neural Processing Letter*, Vol.18(1), 35-48, 2003.

[9] Bishop C.M., Training with noise is equivalent to Tikhonov regularization, *Neural Computation*, Vol.7, 108-116, 1995.

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

[11] 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.

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

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

[14] Edwards P.J., A.F. Murray, Can deterministic penalty terms model the effects of synaptic weight noise on network fault-tolerance, *International Journal of Neural Systems*, 1995.

[15] Gladyshev E., On stochastic approximation, *Theory of Probability and its Applications*, Vol.10, 275-278, 1965.

[16] Grandvalet Y., S. Canu, A comment on noise injection into inputs in back-propagation learning, *IEEE Transactions on Systems, Man, and Cybernetics*, 1995.

[17] Grandvalet Y., S. Canu, S. Boucheron, Noise injection : Theoretical prospects, *Neural Computation*, 1997.

[18] 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.

[19] Hassibi B and D.G. Stork, Second order derivatives for network pruning: Optimal brain surgeon. In Hanson *et al.* (eds) *Advances in Neural Information Processing Systems*, 164-171, 1993.

[20] S. Himavathi, D. Anitha and A. Muthuramalingam, Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization, *IEEE Transactions on Neural Networks*, Vol.18, 880-888, 2007.

[21] Ho K., C.S. Leung, and J. Sum, On weight-noise-injection training, *Proceedings of ICONIP 2008*, Springer LNCS, 2009.

[22] 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.

[23] Kushner H.J. and D.S.Clark, *Stochastic Approximation for Constrained and Unconstrained Systems*, Springer Verlag, 1978.

[24] Lai T.L., Stochastic approximation, *Annals of Statistics*, Vol. 31, No. 2, 391-406, 2003.

[25] LeCun Y. *et al.*, Optimal brain damage, *Advances in Neural Information Processing Systems 2* (D.S. Touretsky, ed.) 396-404, 1990.

[26] Leung C.S., K.W. Wong, P.F. Sum and L.W. Chan, A pruning method for recursive least squared algorithm, *Neural Networks*, 14:147-174, 2001.

[27] Leung C.S., G.H. Young, J. Sum and W.K. Kan, On the regularization of forgetting recursive least square, *IEEE Transactions on Neural Networks*, Vol.10, 1842-1846, 1999.

[28] Leung C.S., K.W.Wong, J. Sum, and L.W.Chan, On-line training and pruning for RLS algorithms, *Electronics Letters*, Vol.32, No.23, 2152-2153, 1996.

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

[30] Leung C.S. and J. Sum Analysis on generalization error of faulty RBF networks with weight decay regularizer, in *Proceedings of ICONIP 2008*, Springer LNCS, 2009.

[31] Leung C.S., P.F. Sum and T.T.Wong, Analysis on Bidirectional Associtive Memories with multiplicative weight noise, *Neural Information Processing*, Springer LNCS, 2007.

[32] Ljung L., Analysis of recursive stochastic algorithms, *IEEE Transaction on Automatic Control*, Vol. 22, pp.551-575, 1977.

[33] Lo Z.P., Y.Yu and B.Bavarian, Analysis of the convergence properties of Topology Preserving Neural Networks, *IEEE Transactions on Neural Network*, Vol. 4, No. 2, pp.207 – 220, 1993.

[34] Mendel J.M. (Ed), *A Prelude to Neural Networks: Adaptive and Learning Systems*, Prentice Hall, New Jersey, 1994.

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

[36] Murata N., 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(6), pp.865-872, 1994.

[37] 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.

[38] 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.

[39] 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.

[40] Parra X. and A. Catala, Fault tolerance in the learning algorithm of radial basis function networks, *Proc. IJCNN 2000*, Vol.3, 527-532, 2000.

[41] Pedersen M.W., L.K. Hansen and J. Larsen. Pruning with generalization based weight saliencies: γOBD, γOBS. *Advances in Information Processing Systems 8* 521-528, 1996.

[42] 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.

[43] Phatak D.S. and E. Tcherner, Synthesis of fault tolerance neural networks, *Proc. IJCNN'02*, 1475-1480, 2002.

[44] Reed R., Pruning algorithms – A survey, *IEEE Transactions on Neural Networks*, Vol.4(5), 740-747, 1993.

[45] Reed R., R.J. Marks II & S. Oh, Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter, *IEEE Transactions on Neural Networks*, Vol.6(3), 529-538, 1995.

[46] Antony W. Savich, Medhat Moussa and Shawki Areibi, The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study, *IEEE Transactions on Neural Networks*, Vol.18, 240-252, 2007.

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

[48] Sher B.Y. and W.S. Hsieh, Fault tolerance training of feedforward neural networks. *Proceedings of the National Science Council, Republic of China (A),* Vol-23, No.5, pp. 599-608, 1999.

[49] Sugiyama, M. and Ogawa, H., Optimal design of regularization term and regularization parameter by subspace information criterion, *Neural Networks*, Vol.15, 349-361, 2002.

[50] Sum J., On a multiple nodes fault tolerant training for RBF: Objective function, sensitivity analysis and relation to generalization, *Proceedings of TAAI'05*, Tainan, ROC, 2005.

[51] Sum J. and K. Ho, On-line estimation of the final prediction error via recursive least square method, *Neurocomputing*, Vol. 69, 2420-2424, 2006.

[52] Sum J., Towards an objective function based framework for fault tolerant learning, in *Proceedings of TAAI'2007.*

[53] Sum J. and C.S. Leung, Prediction error of a fault tolerant neural network, *Neurocomputing*, 2008.

[54] Sum J., C.S. Leung, L. Hsu, Y. Huang, An objective function for single node fault RBF learning, in *Proceedings of TAAI'2007.*

[55] Sum J., C.S. Leung and L. Hsu, Fault tolerant learning using Kullback-Leibler Divergence, in *Proc. TENCON'2007* Taipei, 2007.

[56] 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*, 2008.

[57] Sum J., Fault tolerant learning for neural networks: Survey, framework and future work, to appear in *International Journal of Advanced Information Technologies*, 2008.

[58] Sum J., C.S. Leung, and K. Ho, On node-fault-injection training an RBF network, in *Proceedings of ICONIP 2008*, Springer LNCS, 2009.

[59] Takase H., H. Kita and T. Hayashi, A study on the simple penalty term to the error function from the viewpoint of fault tolearnt training, *Proc. IJCNN 2004*, 1045-1050, 2004.

[60] Velazco R. *et al*, SEU fault tolerance in artificial neural networks, *IEEE Transactions on Nuclear Science*, Vol. 42 (6), 1856-1862, 1995.