Learning with weight noise, node fault and weight decay

John Sum

Institute of Technology Management
National Chung Hsing University

In collaboration with
Chi-sing Leung (CityU HK), Kevin Ho (Providence University)
and Allen Liang (NCHU)

Research supported in part by
Taiwan NSC Grants (No. 95-2221-E-040-009, 97-2221-E-005-050,
98-2221-E-005-048),
City University of Hong Kong Grant (No. 7002230),
HK Government General Research Fund (No.: CityU 116508)

OUTLINE

Fault tolerance learning

Learning with weight noise and weight decay

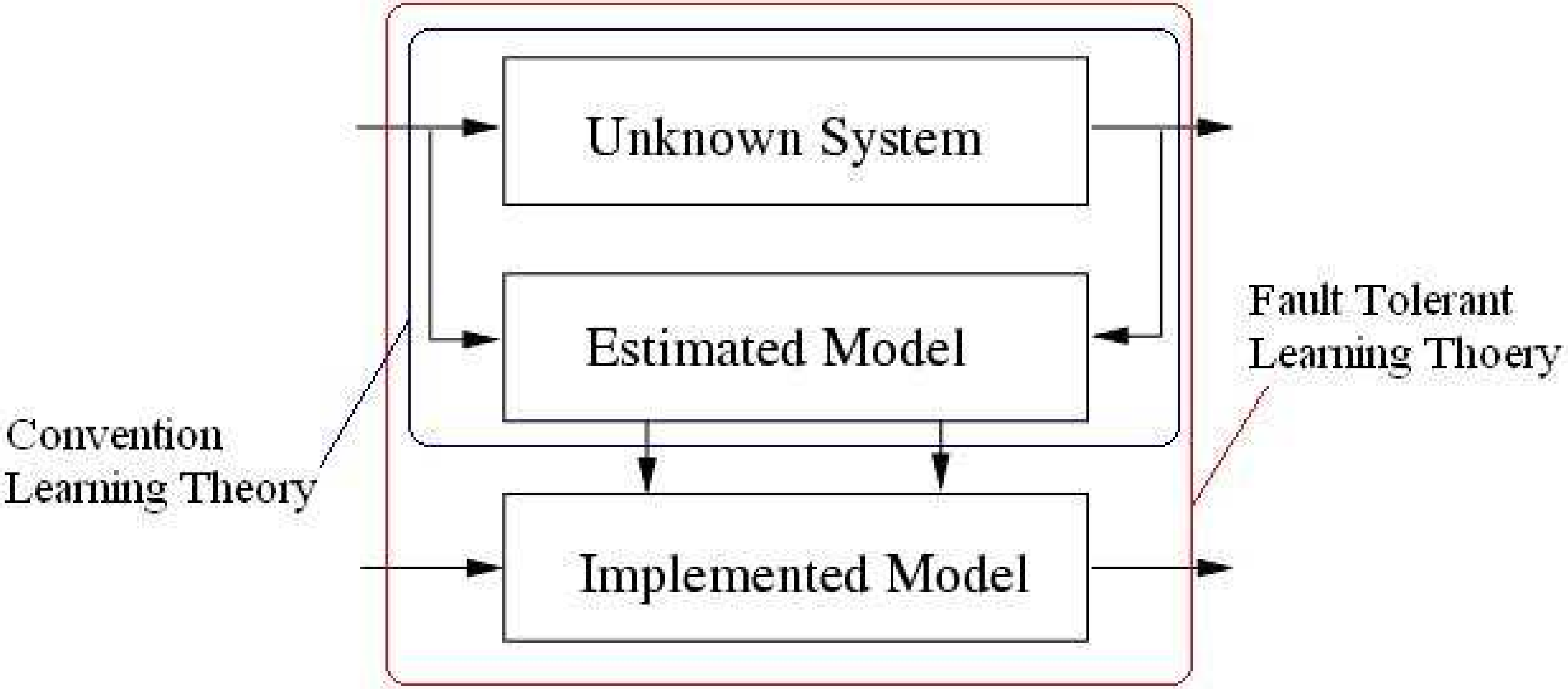
Learning with node fault and weight decay

Conclusions

PART I: Fault tolerance learning

Background: Fundamental problem

Problem



Estimated Model: Fault-free RBF

Implemented Model: Faulty RBF

Background: Fault models

- Node fault – Single node fault or multiple nodes fault

- Weight noise (weight perturbation) – Additive or multiplicative

- Input noise (input perturbation)– Additive or multiplicative

- Other perturbation – e.g. the centers and widths in RBF, and the parameter τ in the sigmoid function

- SEU – Single Event Upset, hardware bit flip due to radiation

Background: Previous approaches

<u>Approach I (Regularization)</u>

Step 1: Define an objective function

Step 2: Derive batch-mode or online-mode training algorithms

<u>Approach II (Fault injection during training)</u>

Step 1: Start with a heuristic idea

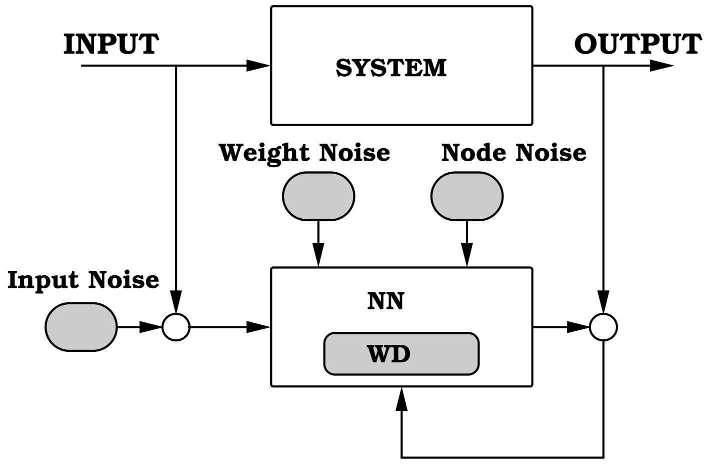
Step 2: Modify BP learning algorithm by injecting fault

<u>Assessment</u>

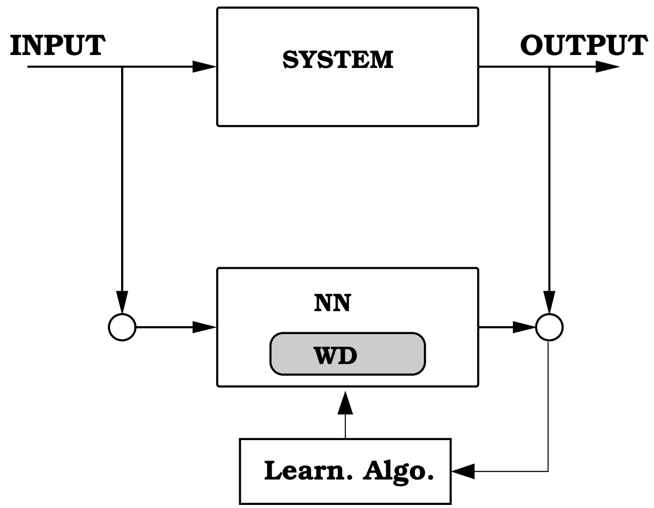
(a) Training and testing error

(b) Prediction MSE versus *fault level*

Background: Previous approaches

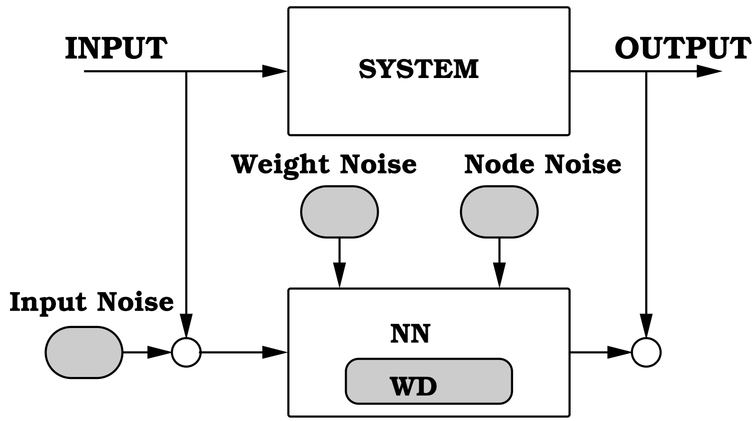


Noise injection–based online training



Objective function–based online training

NN: NEURAL NETWORK
WD: WEIGHT DECAY



Noise injection after training

Background: Motivated Questions

Question 1: Objective functions for FTL

- Not all existing FTL algorithms are defined based upon objective functions. Some of these are designed by heuristic.
- Is it possible to find an objective function for them ?

Question 2: About weight decay

- Algorithm like weight decay has also been applied in training a NN of good fault tolerance and generalization.
- Does it mean that weight decay should be an universal technique for NN learning ?

Question 3: Connection to conventional (batch & online) learning

- If the objective functions are found, what are their similarities, differences and relationships with those defined in conventional learning ?

Question 4: Theoretical framework

- Some research articles in the literature have summarized the previous works in regard to fault tolerant neural networks.
- But, little theoretical work has been done and almost no previous work has been done along the statistical learning point of view.

Previous works (1): NN learning algorithms

Sequin & Clay (1991); Bolt (1992)
– Modified backpropagation learning
– MLP
– Inject random node fault (stuch-at fault) during training

Edward & Murray (1993, 1994, 1996)
– Modified backpropagation learning
– MLP
– Weight noise injection during training

Chiu (1994)
– Modified backpropagation learning
– MLP
– Inject random node fault with random node deletion and addition

Cavalieri & Mirabella in (1999)
– Modified back-propagation learning
– MLP
– Weight magnitude control step

Parra and Catala (2000)
– RBF network
– Weight decay regularizer

Bernier *et al* (2000, 2003)
– Explicit regularization
– MLP & RBF

Leung & Sum (2005, 2008)
– FT regularizer
– RBF
– Batch-mode learning, node fault

Leung & Sum (2007)
– KL-Divergence based objective function
– RBF
– Batch-mode learning, MWN

Sum & Leung (2009)
– Fault tolerant regularizer
– RBF
– Batch-mode learning, MWN

Previous works (2): NN generation methods

Emmerson & Damper (1993); Phatak & Koren (1995)
– Network generation method
– MLP
– Adding network redundancy

Simon (2001)
– Distributed fault tolerance learning
– Optimal interpolation net
– Nonlinear programming problem

Nonlinear MinMax optimization (Single node fault)
– Max of the mean square errors over all fault models, i.e. l_2-norm (Neti *et al* 92)
– Max square error over all fault models, i.e. l_∞-norm (Deodhare *et al* 98)

Previous works (3): NN performance analysis

Ref.	Fault	NN	Work
[48]	Any weight noise	Madaline	Probability of output error
[15]	Any noise	Any	Output sensitivity measure
[43]	Any noise	Madaline	Precision requirement
[10]	Mul. weight noise	RBF	Generalizaton ability
[54]	Any noise	RBF	Output sensitivity matrix
[2]	Any weight noise	MLP	Output sensitivity measure
[41]	-	-	Relationship between FT, generalization and VC dim.
[4]	Any weight noise	MLP	Generalization ability
[19]	Any weight noise	FN[a]	Error sensitivity measure

[a] Functional net

Previous works (4): NN convergence analysis

Edward & Murray (1994)
– Analysis on injecting weight noise during training
– Multiplicative/additive
– MLP
– Prediction error
– No objective function
– No convergence analysis

An (1996)
– Analysis on injecting noise during training
– Input noise, weight noise, additive
– MLP
– Objective functions
– No convergence analysis

Ho, Leung & Sum (2008, 2010)
– Convergence analysis, objective functions
– RBF
– Injecting noise/fault during training, weight or node
– With/without weight decay
– Theoretical analysis

Ho, Leung & Sum (2009)
– Divergence of pure weight noise injection
– MLP
– Simulations

Ho, Leung & Sum (2010)
– Convergence analysis, objective functions
– MLP
– Injecting noise/fault during training, weight or node
– With weight decay
– Theoretical analysis

PART II: Learning with weight noise and weight decay

Multilayer Perceptron (MLP)

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

$\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_l] \in R^{m \times l}$ 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

$\mathbf{c} \in R^m$ is the input to hidden bias vector.

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

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

Algorithms

Pure weight noise injection

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t(y_t - f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)). \quad (2)$$

$$\tilde{\mathbf{w}}(t) = \mathbf{w}(t) + \mathbf{b} \otimes \mathbf{w}(t). \quad \text{(multi. noise)} \quad (3)$$

$$\tilde{\mathbf{w}}(t) = \mathbf{w}(t) + \mathbf{b}. \quad \text{(additive noise)} \quad (4)$$

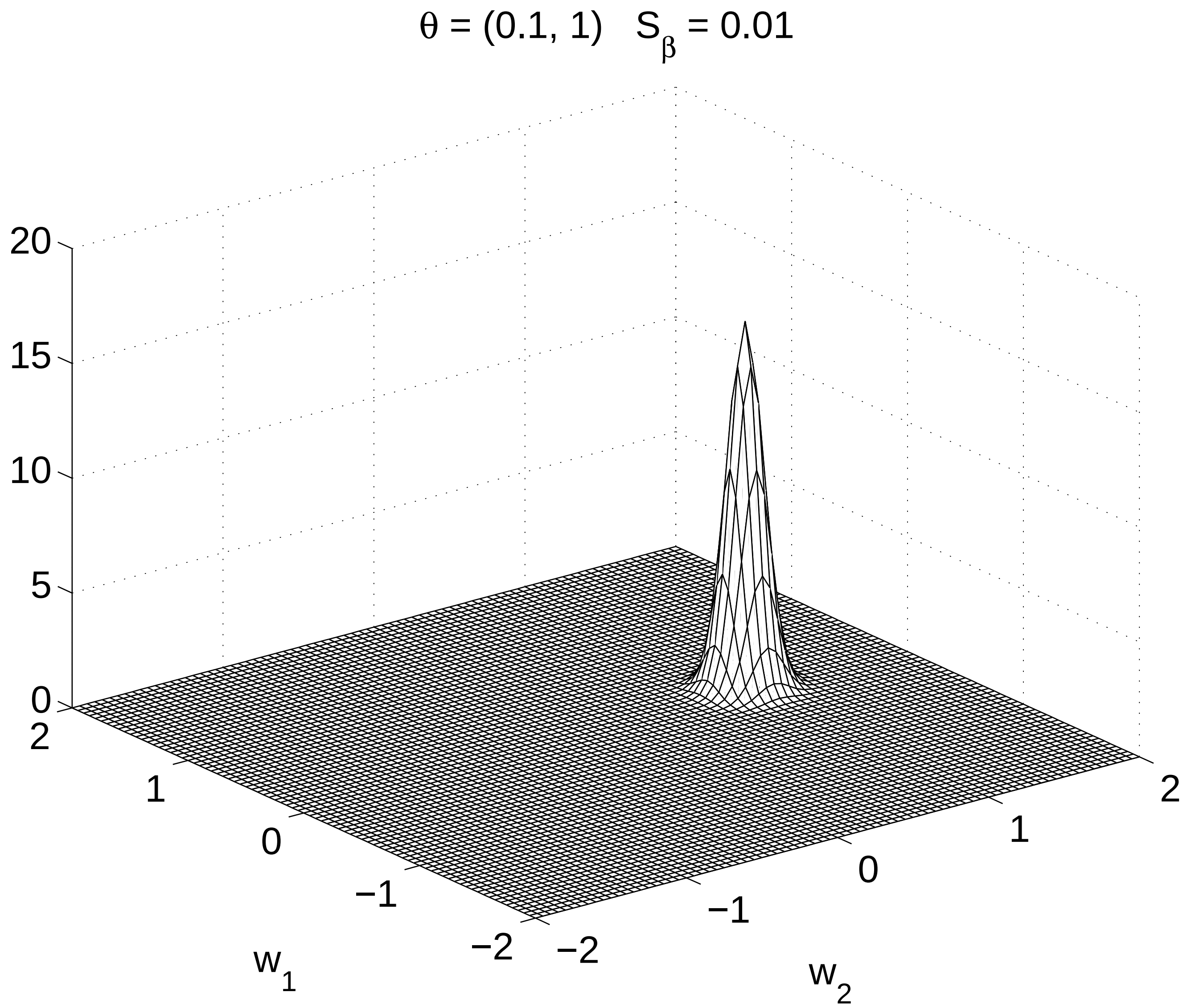
Here $\mathbf{b} \otimes \mathbf{w} = (b_1 w_1, b_2 w_2, \cdots, b_M w_M)^T$ and b_i, for all i, is a mean zero Gaussian distribution with variance S_b.

Weight noise injection with weight decay

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \mu_t \left\{ (y_t - f(\mathbf{x}_t, \tilde{\mathbf{w}}(t)))\mathbf{g}(\mathbf{x}_t, \tilde{\mathbf{w}}(t)) - \alpha \mathbf{w}(t) \right\}. \quad (5)$$

$$\tilde{\mathbf{w}}(t) = \mathbf{w}(t) + \mathbf{b} \otimes \mathbf{w}(t). \quad \text{(multi. noise)} \quad (6)$$

$$\tilde{\mathbf{w}}(t) = \mathbf{w}(t) + \mathbf{b}. \quad \text{(additive noise)} \quad (7)$$



Probability density function of $\mathbf{b} \otimes \mathbf{w}$ when $\mathbf{w} = (0.1, 1)^T$ and $S_b = 0.01$.

Objective function

Pure multiplicative weight noise

$$\begin{aligned}\mathbf{V}(\mathbf{w}) &= \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2 + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} w_j^2 g_j(\mathbf{x}_k,\mathbf{w})^2\right\} \\ &\quad -\frac{S_b}{N}\sum_{k=1}^{N}\int_{\mathbf{w}_0}^{\mathbf{w}} \mathbf{u}(\mathbf{x}_k,\mathbf{r})\mathbf{d}r, \qquad (8)\end{aligned}$$

where

$$\mathbf{u}(\mathbf{x}_k,\mathbf{w}) = (w_1 g_1(\mathbf{x}_k,\mathbf{w})^2, \cdots, w_M g_M(\mathbf{x}_k,\mathbf{w})^2)^T. \qquad (9)$$

Mutilplicative weight noise with weight decay

$$\begin{aligned}\mathbf{V}(\mathbf{w}) &= \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2 + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} w_j^2 g_j(\mathbf{x}_k,\mathbf{w})^2\right\} \\ &\quad -\frac{S_b}{N}\sum_{k=1}^{N}\int_{\mathbf{w}_0}^{\mathbf{w}} \mathbf{u}(\mathbf{x}_k,\mathbf{r})\mathbf{d}r + \frac{\alpha}{2}\|\mathbf{w}\|^2. \qquad (10)\end{aligned}$$

Pure additive weight noise

$$\mathbf{V}(\mathbf{w}) = \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2 + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} g_j(\mathbf{x}_k,\mathbf{w})^2\right\}. \quad (11)$$

Additive weight noise with weight decay

$$\mathbf{V}(\mathbf{w}) = \frac{1}{2}\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2 + \frac{S_b}{N}\sum_{k=1}^{N}\sum_{j=1}^{M} g_j(\mathbf{x}_k,\mathbf{w})^2 + \alpha\|\mathbf{w}\|^2\right\}. \quad (12)$$

Convergence

Theorem 1 *For the algorithm based on (5) and (6), if (i)* $\alpha > 0$*, (ii)* $S_b < 1$ *and (iii)* $0 < \mu(t)(\alpha + (1 - \sqrt{S_b})m) < 1$ *for all* $t \geq 0$*, then* $\lim_{t\to\infty} \mathbf{w}(t) = \mathbf{w}^*$ *exists and its elements are finite with probability one,*

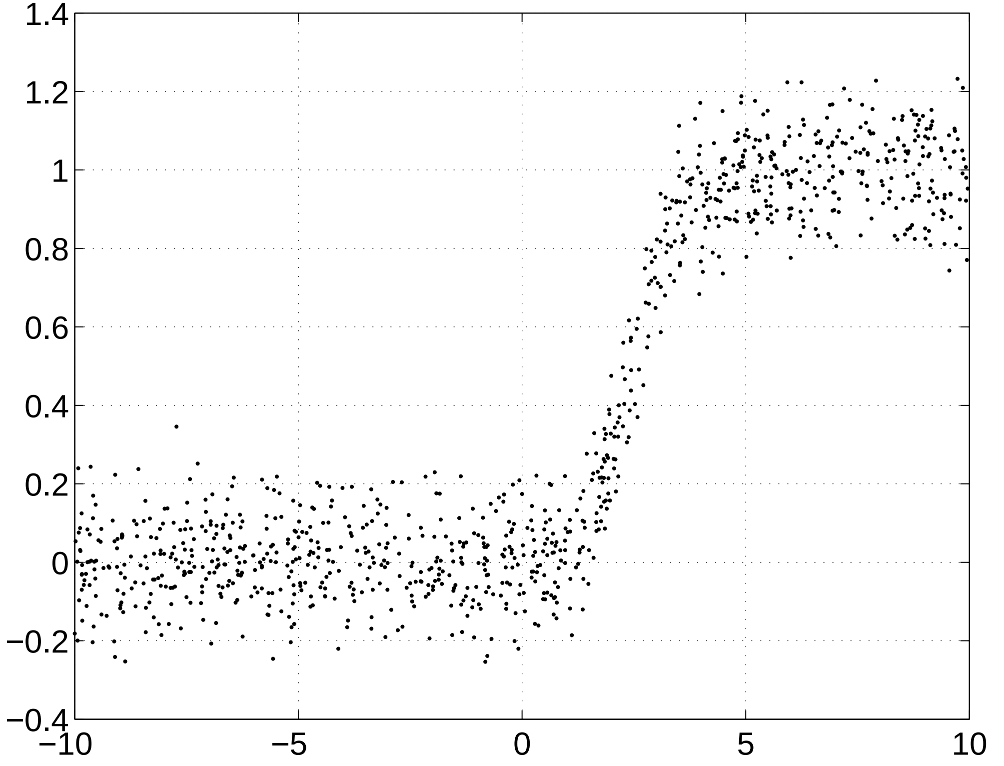
Theorem 2 *For the algorithm based on (5) and (6), if (i)* $\alpha > 0$*, (ii)* $S_b \ll 1$*, (iii)* $\mu(t) \to 0$ *for all* $t \geq 0$ *and (iv)* $\sum_{\tau=t}^{\infty} \mu(t) = \infty$ *for any* $t \geq 0$*, then* $\mathbf{w}(t)$ *converges to the location in which*

$$\nabla_{\mathbf{w}} V(\mathbf{w}^*) = \lim_{t\to\infty} \nabla_{\mathbf{w}} V(\mathbf{w}(t)) = \mathbf{0}, \tag{13}$$

where $V(\mathbf{w})$ *is a scalar function given by (10).*

Simulations

<u>Training data</u>



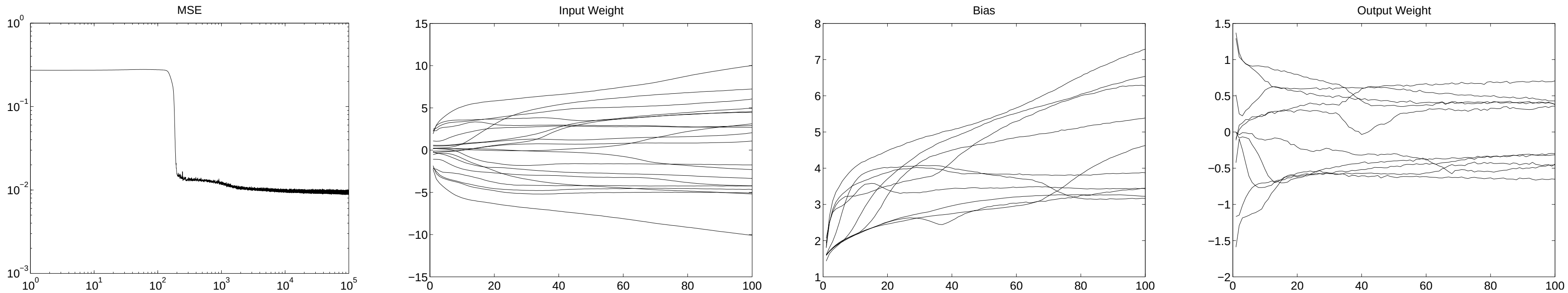
Network structure: 1 input node, 10 hidden nodes, 1 output node

During training: $S_b = 0.01$, $\alpha = 0.00001$

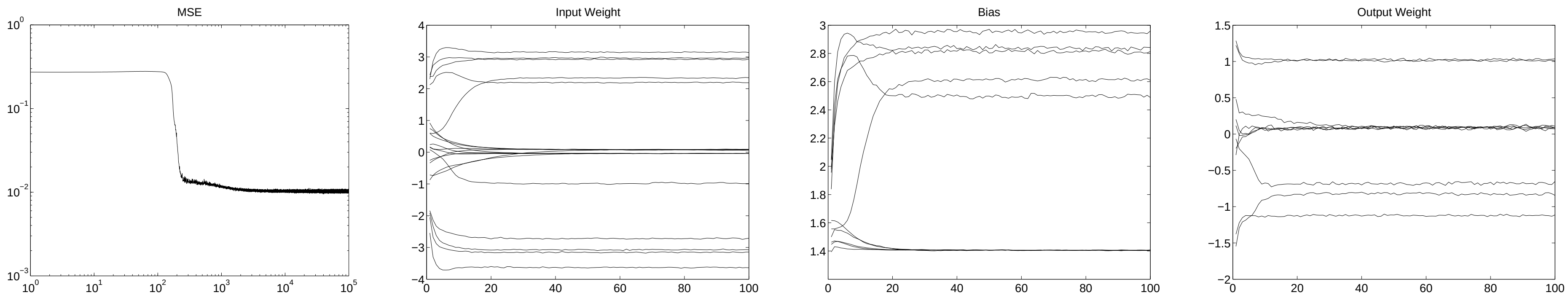
During testing: $S_b \in [0, 0.04]$

Multiplicative weight noise

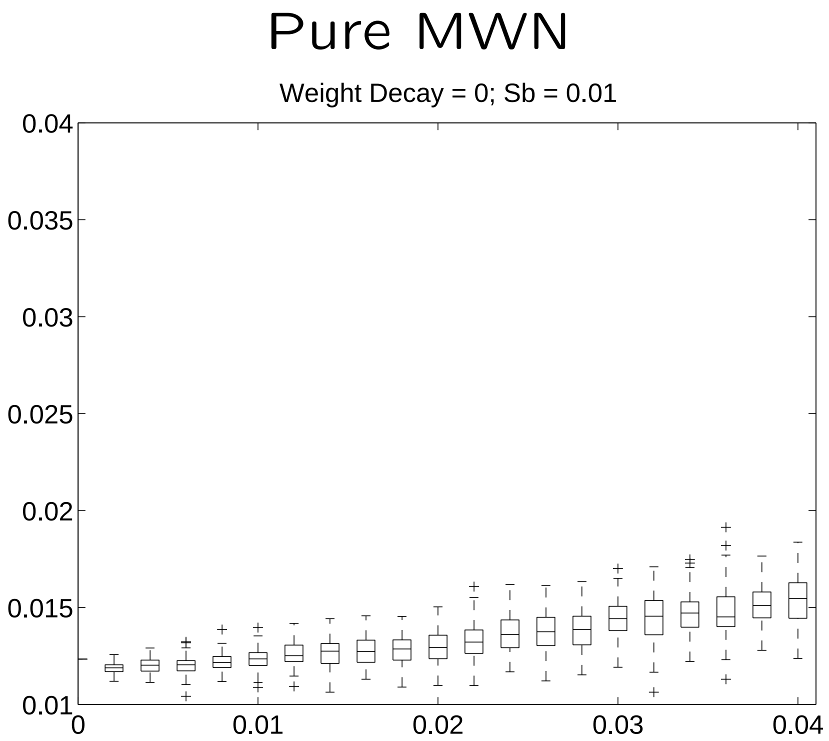
Pure MWN

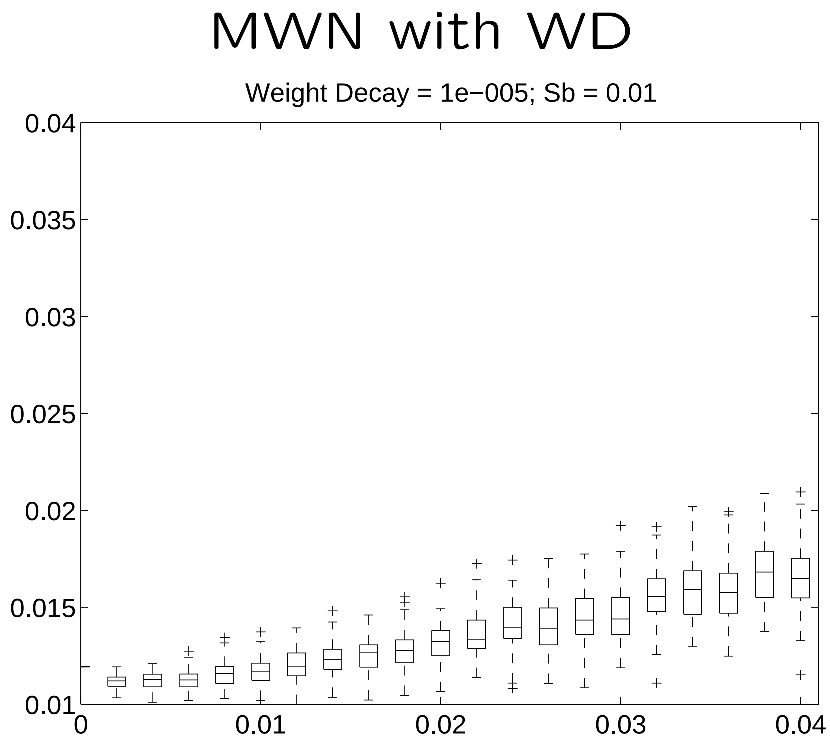


MWN with WD



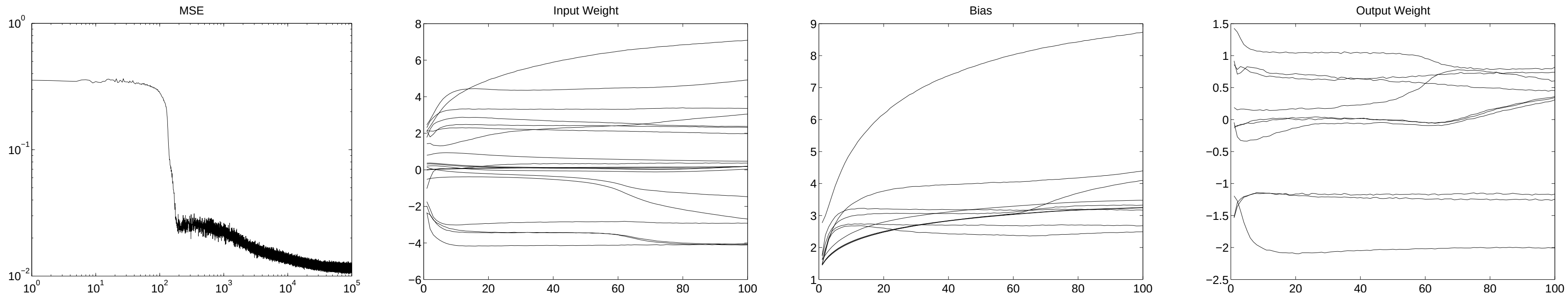
Testing error



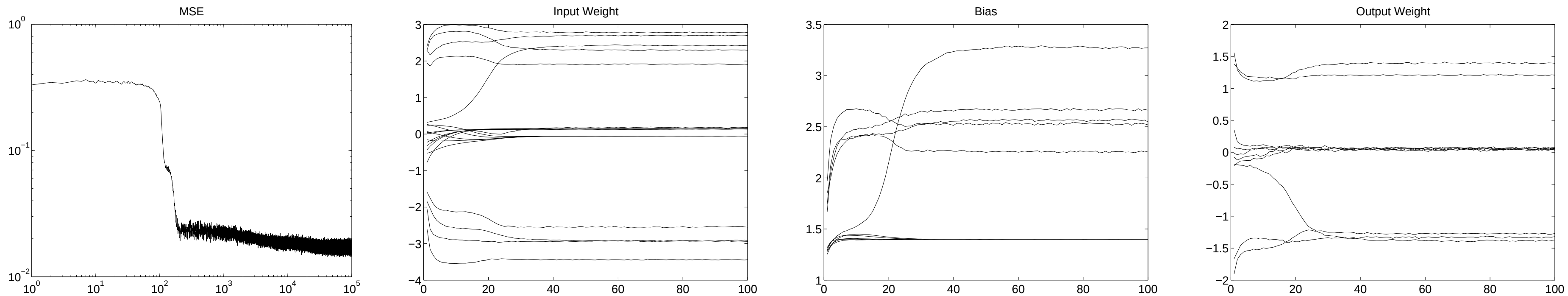


Additive weight noise

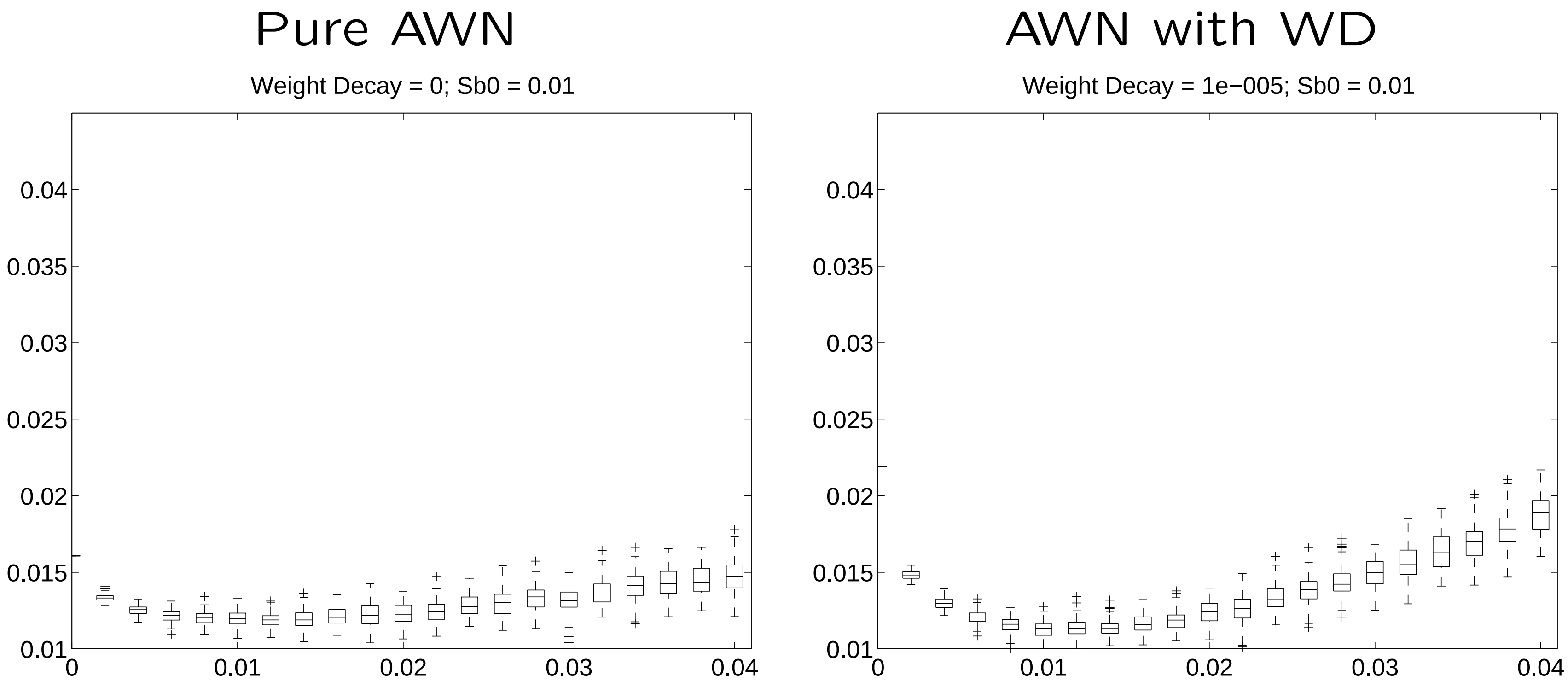
Pure AWN



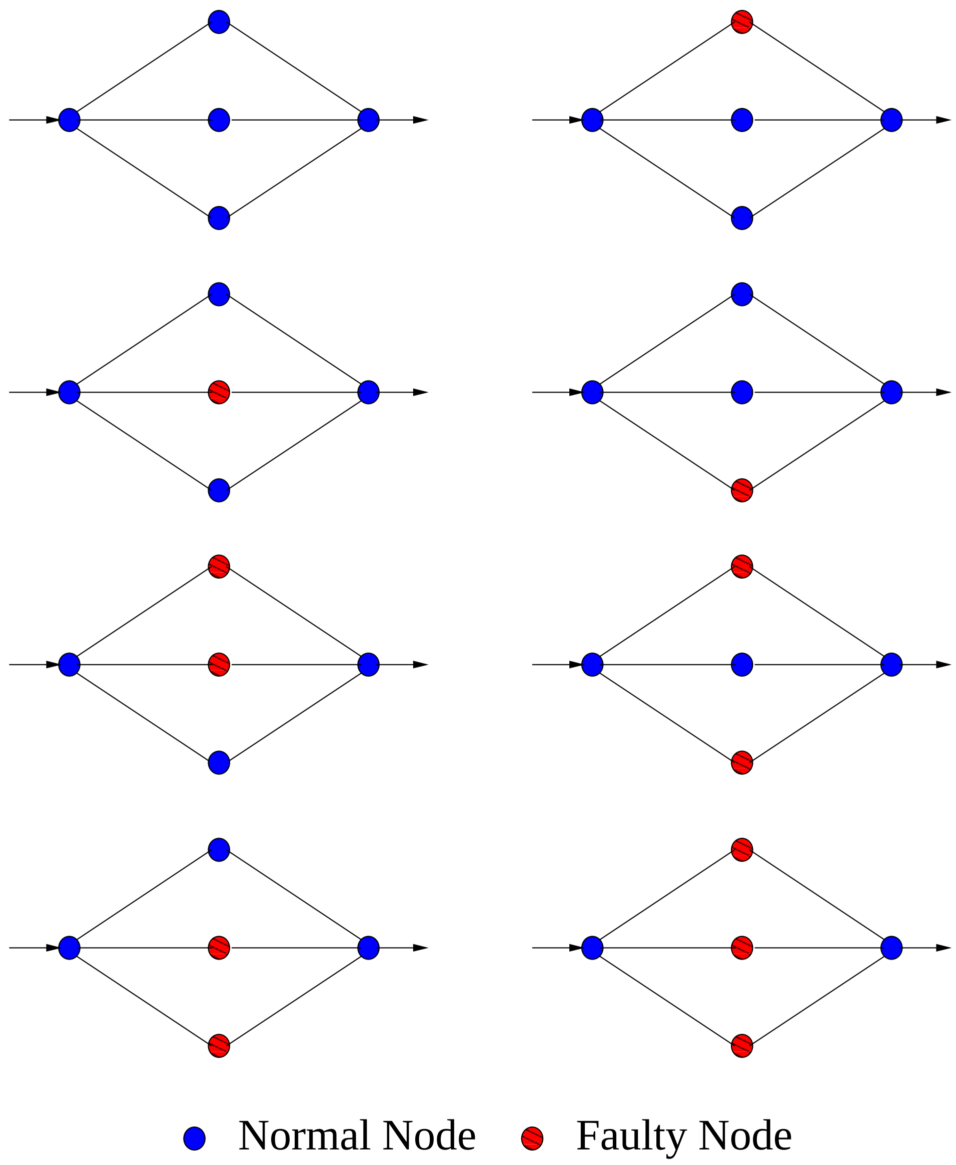
AWN with WD



Testing error



PART III: Learning with node fault and weight decay



For a MLP with 3 hidden nodes, there are 1 fault-free and 7 faulty structures.

Fault model

Let $\mathbf{b}(t) = (b_1(t), b_2(t), \cdots, b_m(t))^T \in \{0, 1\}^m$.

$$b_i(t) = \begin{cases} 1 & \text{if the } i^{th} \text{ hidden node is normal,} \\ 0 & \text{if the } i^{th} \text{ hidden node is faulty.} \end{cases} \tag{14}$$

$$P(b_i(t)) = \begin{cases} 1 - p & \text{if } b_i(t) = 1 \\ p & \text{if } b_i(t) = 0. \end{cases} \tag{15}$$

For all i, j $(i \neq j)$, t, t' $(t \neq t')$, the random variables $b_i(t), b_i(t'), b_j(t)$ are all identical and independent.

Algorithm

$$\begin{aligned} \tilde{z}_i(t) &= b_i(t) z_i(t) && (16)\\ \tilde{f}(\mathbf{x}_t, \mathbf{w}(t)) &= \sum_{i=1}^{m} d_i(t) \tilde{z}_i(t) && (17)\\ \tilde{\mathbf{g}}_i(\mathbf{x}_t, \mathbf{w}(t)) &= \begin{bmatrix} \tilde{z}_i(t) \\ d_i(t)\tilde{z}_i(t)(1 - \tilde{z}_i(t))\mathbf{x}_t \\ d_i(t)\tilde{z}_i(t)(1 - \tilde{z}_i(t)) \end{bmatrix}. && (18) \end{aligned}$$

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \mu(t) \left\{ (y_t - \tilde{f}(\mathbf{x}_t, \mathbf{w}(t)))\tilde{\mathbf{g}}_i(\mathbf{x}_t, \mathbf{w}(t)) - \alpha \mathbf{w}_i(t) \right\}, \quad (19)$$

where $\mu(t) > 0$ is the step size at the t^{th} step and $\alpha > 0$ is called the decay constant.

Theorem 3 *The objective function of algorithm (19) is given by*

$$V(\mathbf{w}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - f(\mathbf{x}_k,\mathbf{w}))^2 + \frac{p}{N}\sum_{k=1}^{N}\mathbf{d}^T\left(\mathbf{G}(\mathbf{x}_k,\mathbf{w}) - \mathbf{H}(\mathbf{x}_k,\mathbf{w})\right)\mathbf{d} + \frac{\alpha}{(1-p)}\|\mathbf{w}\|_2^2. \quad (20)$$

The matrices $\mathbf{G}(\mathbf{x}_k,\mathbf{w})$ *and* $\mathbf{H}(\mathbf{x}_k,\mathbf{w})$ *in (20) are given by*

$$\mathbf{G}(\mathbf{x}_k,\mathbf{w}) = \textbf{diag}\{z_1^2(\mathbf{x}_k,\mathbf{w}),\cdots,z_m^2(\mathbf{x}_k,\mathbf{w})\}, \quad (21)$$

$$\mathbf{H}(\mathbf{x}_k,\mathbf{w}) = \mathbf{z}(\mathbf{x}_k,\mathbf{w})\mathbf{z}(\mathbf{x}_k,\mathbf{w})^T. \quad (22)$$

Theorem 4 *For arbitrarily given* $\mathbf{w}(0)$*,* $\mathbf{w}(t)$ *defined by (19) converges to* $\{\mathbf{w}|\left(\frac{\partial}{\partial\mathbf{w}}V(\mathbf{w})\right)^T\bar{\mathbf{M}}(\mathbf{w}) = 0\}$ *with probability 1.*

PART IV: Implication and conclusions

Implications (It is just my thought. It could be wrong!)

IF

– Input noise, weight noise, node noise $\rightarrow$ Brain noise

– Node fault $\rightarrow$ Neuronal cell death, synapse re-connections

– Weight decay $\rightarrow$ Forgetting

THEN

– Brain noise alone $\rightarrow$ brain state instability

– Synapse re-connections $\rightarrow$ increase redundacy

– Forgetting $\rightarrow$ alleviates instability due to brain noise

Proof: Sum *et al* (Might appear 10 years later.)

Conclusions

Survey on researches in fault tolerance learning

Recent results on online weight noise injection algorithms

Recent results on online node fault injection algorithms

Implication about the importance of weight decay

THANK YOU

John Sum: pfsum@nchu.edu.tw