# Fault Tolerant Learning Using Kullback-Leibler Divergence

John Sum
Institute of Electronic Commerce
National Chung Hsing University
Taichung 402, Taiwan
Email: pfsum@yahoo.com.hk

Chi-sing Leung
Department of Electronic Engineering
City University of Hong Kong
Kowloon Tong, Hong Kong
Email: eeleungc@cityu.edu.hk

Lipin Hsu
Department of Information Management
Chung Shan Medical University
Taichung 402, Taiwan
Email: apple@csmu.edu.tw

***Abstract*— In this paper, an objective function for training a fault tolerant neural network is derived based on the idea of Kullback-Leibler (KL) divergence. The new objective function is then applied to a radial basis function (RBF) network that is with multiplicative weight noise. Simulation results have demonstrated that the RBF network trained in accordance with the new objective function is of better fault tolerance ability, in compared with the one trained by explicit regularization. As KL divergence has relation to Bayesian learning, a discussion on the proposed objective function and the other Bayesian type objective functions is discussed.**

## I. Introduction

Once a neural network model has been obtained numerically by a conventional computer, the model has finally be implemented by physical hardware. However, this mapping can never be perfect [8]. One imperfection is the finite bit number representation, that can be found in digital hardware [15], [18], which eventually makes the digital implementation of a neural network suffer from multiplicative weight noise. To alleviate such problem, many works have been done in the last two decades. Some of them aimed at understand the effect of such noise on a neural network. While some others aimed at training a neural network that is able to tolerate such effect.

Stevenson *et al* [25] amongst the first group gave a comprehensive analysis on the *probability of output error* of Threshold Logic Madaline model due to input and weight noise. Choi and Choi [12] from statistical sensitivity approach to derive different *output sensitivity measures* of a multilayer perceptron. Piche in [22] followed an approach from signal to noise ratio (SNR) to derive a set of measures for the output sensitivity of the Sigmoidal Madaline and applied it to develop a weight accuracy selection algorithm to determine the precision requirement for hardware implementation. Townsend and Tarassenko [26] considered a radial basis function (RBF) network with multiple outputs. They derived an output sensitivity matrix for an RBF network that is suffered from perturbations in input data, basis function centers and output weights.

Output sensitivity is just one view point to understand the effect of neural network due to noise. An alternative view point is from the actual network performance, i.e. the generalization ability. In this regard, Catala and Parra proposed a fault tolerance parameter model and studied the degradation of a RBF network if the RBF centers, widths and the corresponding weights due to multiplicative noise [9]. Following Choi & Choi's statistical sensitivity approach [12], Bernier *et al* derived the *error sensitivity measures* for MLP [2], [4] and RBF network [6]. Fontenla-Romero *et al* derived the *error sensitivity measure* for functional nets [13].

While noise can be harmful to a neural network, Murray & Edwards [20] on the other hand investigated advantages of adding noise to a neural network during training. In their paper, they have found that adding noise during training can actually improve the generalization ability of a neural network, Bishop [7] showed that adding small additive white noise to a neural network during training is equivalent to adding Tikhnov regularization. Jim *et al* [16] further noticed that adding multiplicative weight noise not just can improve the generalization ability, but also can improve the convergence ability in training a recurrent neural network.

While the above works have provided a clearer pictures on the effect of noise to the network performance, some other researchers developed training methods aiming to improve the fault tolerant ability of a neural network. Owing to reduce the magnitude of the weights, Cavalieri & Mirabella in [10] proposed a modified backpropagation learning, in which a weight magnitude control step has been added in each training epoch, for multilayer perceptron. Simon in [24] developed a distributed fault tolerance learning for optimal interpolation net, in which the learning is formulated as a nonlinear programming problem – minimizing the training error subjected to an equality constraint on weight magnitude. Extended from the their previous works, Parra and Catala in [21] demonstrated how a fault tolerant RBF network can be obtained by using a simple weight decay regularizer [19]. Bernier *et al* developed a method called explicit regularization to attain a MLP [3], [5] and RBF network [6].

In this paper, we are also interested in developing a learning algorithm to train a neural network that is able to tolerate weight noise. Specifically, a fault tolerant learning using Kullback-Leibler (KL) divergence [17] will be derived and applied to train an RBF network that is suffered from multiplicative weight noise. In the next section, an objective function based on KL divergence will be derived. Then, a learning algorithm for RBF network will be derived in Section 3. Section 4 presents a simulation results. A discussion of the new objective function and others will be presented in Section 5. Finally, the conlcusion will be presented in Section 6.

1-4244-1272-2/07/$25.00 ©2007 IEEE

## II. KLD-based Objective Function

Assume that a measured data set $\mathcal{D}$ is collected from an unknown *system* with input-output joint probability distribution $P_0(x, y)$. Without loss of generality, we assume that $x, y \in R$. We further assume that the unknown system belongs to a set of models that can be parameterized by a $M$-dimensional parametric vector $\theta$, given by $\theta = (\theta_1, \theta_2, \cdots, \theta_M)^T$. The corresponding input-output behavior is represented by $P(x, y|\theta)$. An implementation of a model $\theta$, denoted by $\tilde{\theta}$, is a random model that is generated by a probability function $P(\tilde{\theta}|\theta)$. We also call this implementation a faulty network model. Normally, the probability $P(\tilde{\theta}|\theta)$ is determined by the fault model or the noise model given in advance. Clearly, a faulty neural network model $\tilde{\theta}$ is also an unknown system to us. Not until the nominal model $\theta$ has been implemented, it is not able to check the true performance of $\tilde{\theta}$.

The marginal probability distribution will thus be given by

$$P(x, y|\theta) = \int P(x, y|\tilde{\theta}, \theta) P(\tilde{\theta}|\theta) d\tilde{\theta}$$

and treating the training data is sampled from an unknown but stochastic system $P_0(x, y)$, the performance measure of a fault tolerant neural network can naturally be quantified by Kullback-Leibler (KL) divergence [17],

$$\int \int P_0(x, y) \log \frac{P_0(x, y)}{P(x, y|\theta)} dxdy.$$

Therefore, $\hat{\theta}$ can be given by

$$\hat{\theta} = \arg\min_{\theta} \left\{ D(P_0(x, y)||P(x, y|\theta)) \right\}. \tag{1}$$

where

$$D(P_0(x, y)||P(x, y|\theta)) = \text{constant} - \mathcal{L}(\theta), \tag{2}$$

and

$$\mathcal{L}(\theta) = \int \int P_0(x, y) \log \left\{ \int P(x, y|\tilde{\theta}, \theta) P(\tilde{\theta}|\theta) d\tilde{\theta} \right\} dxdy. \tag{3}$$

Then, $\hat{\theta}$ can be obtained by

$$\hat{\theta} = \arg\max_{\theta} \left\{ \mathcal{L}(\theta) \right\}. \tag{4}$$

For $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^N$, $N$ is large and by Law of Large Number,

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{k=1}^{N} \log \int P(y_k|x_k, \tilde{\theta}, \theta) P(\tilde{\theta}|\theta) d\tilde{\theta}. \tag{5}$$

Hence,

$$\hat{\theta} = \arg\max_{\theta} \left\{ \frac{1}{N} \sum_{k=1}^{N} \log \int P(y_k|x_k, \tilde{\theta}, \theta) P(\tilde{\theta}|\theta) d\tilde{\theta} \right\}. \tag{6}$$

Which is depended on the noise model $P(\tilde{\theta}|\theta)$.

## III. FT Learning for RBF

In the fault-free radial basis function (RBF) network approach, we assume that the dataset $\mathcal{D}$ is generated by a RBF network, given by

$$f(x) = \sum_{i=1}^{M} \theta_i \phi_i(x) + e \tag{7}$$

$\theta = (\theta_1, \cdots, \theta_M)^T$ is the RBF weight vector; $e$ is a zero-mean Gaussian random variables with variance $S_e$; and $\phi(x) = (\phi_1(x), \phi_2(x), \cdots, \phi_M(x))^T$ and $\phi_i(x)$ for all $i = 1, 2, \cdots, M$ are the radial basis functions given by

$$\phi_i(x) = \exp\left( -\frac{(x - c_i)^2}{\sigma} \right). \tag{8}$$

$c_i$s, for all $i = 1, 2, \cdots, M$ are the radial basis function centers and $\sigma$ is a positive parameter controlling the width of the basis functions.

A faulty implementation of $\theta$ is denoted by $\tilde{\theta}$ in which the elements of $\tilde{\theta}$ is a random variable defined as follows :

$$\tilde{\theta}_i = \theta_i + \beta_i \, \theta_i, \quad \forall \ i = 1, 2, \cdots, M.$$

$\beta_i$ is a mean zero Gaussian noise of variance $S_\beta$.

$$P(\beta_i) = \frac{1}{\sqrt{2\pi S_\beta}} \exp\left( -\frac{\beta_i^2}{2S_\beta} \right) \tag{9}$$

for all $i = 1, 2, \cdots, M$. Given an input $x$, the noise vector $\beta$ and the parametric vector $\theta$, the output of an implementation can thus be given by

$$P(y|x, \beta, \theta) = \frac{1}{\sqrt{2\pi S_e}} \exp\left( -\frac{(y - \sum_{i=1}^{M} \phi_i(x)(1 + \beta_i)\theta_i)^2}{2S_e} \right). \tag{10}$$

The marginal probability over all possible $\beta$s is then be defined as follows :

$$P(y|x, \theta) = \int P(y|x, \tilde{\theta}, \theta) P(\tilde{\theta}|\theta) d\tilde{\theta} = \int P(y|x, \beta, \theta) P(\beta) d\beta.$$

Put the definitions of $P(\beta_i)$ in Equation (9) and $P(y|x, \beta, \theta)$ in Equation (10), and integrate over all possible $\beta$,

$$P(y|x, \theta) = \frac{1}{\sqrt{2\pi S(x, \theta)}} \exp\left( -\frac{(y - \hat{f}(x, \theta))^2}{2S(x, \theta)} \right) \tag{11}$$

$$\hat{f}(x, \theta) = \int y P(y|x, \theta) dy = \phi^T(x)\theta \tag{12}$$

$$S(x, \theta) = S_e + S_\beta \sum_{i=1}^{M} \phi_i^2(x) \theta_i^2. \tag{13}$$

In sequel, $\mathcal{L}(\theta)$ in Equation (5) can then be written as follows :

$$\begin{aligned} \mathcal{L}(\theta) &= -\frac{1}{2} \log 2\pi - \frac{1}{2N} \sum_{k=1}^{N} \log S(x_k, \theta) \\ &\quad - \frac{1}{2N} \sum_{k=1}^{N} \frac{(y_k - \phi^T(x_k)\theta)^2}{S(x_k, \theta)} \end{aligned} \tag{14}$$

and $\hat{\theta}$ equals to the

$$\arg\min_{\theta}\left\{\frac{1}{2N}\sum_{k=1}^{N}\log S(x_k,\theta)+\frac{1}{2N}\sum_{k=1}^{N}\frac{(y_k-\phi^T(x_k)\theta)^2}{S(x_k,\theta)}\right\}.$$

By using the idea of gradient descent, a training algorithm can thus be obtained by the following recursive equation :

$$\theta(t+1)=\theta(t)-\mu\frac{\partial}{\partial\theta}\,\mathcal{L}(\theta(t)), \tag{15}$$

where $\mu$ is a small positive value corresponding to the step size and

$$\begin{aligned}&\frac{\partial\mathcal{L}(\theta)}{\partial\theta}\\ =\;&\frac{S_\beta}{N}\sum_{k=1}^{N}\left(\frac{1}{S(x_k,\theta)}-\frac{(y_k-\phi^T(x_k)\theta)^2}{S^2(x_k,\theta)}\right)G(x_k)\theta\\ -\;&\frac{1}{N}\sum_{k=1}^{N}\frac{(y_k-\phi^T(x_k)\theta)}{S(x_k,\theta)}\phi(x_k),\end{aligned} \tag{16}$$

where

$$G(x_k)=\mathbf{diag}\left\{\phi_1^2(x_k),\phi_2^2(x_k),\cdots,\phi_M^2(x_k)\right\}.$$

The initial condition $\theta(0)$ is set to be a small random vector close to null.

## IV. Simulations

To study the performance of our proposed objective function, a simulated experiment based on sinc function approximation have been conducted. Sinc function is a common benchmark example [11], [27] in many neural network researches. The data is generated by a function of following form :

$$y=\mathrm{sinc}(x)+e, \tag{17}$$

where the noise term $e$ is a mean zero Gaussian noise with variance $S_e$. The RBF network to be studied has 37 RBF nodes and their centers are located in $\{-4.5,-4.25,\cdots,4.25,4.5\}$. $\sigma$ is set to 0.1.

In the experiment, eight different values of $S_e$ have been examined :

$$\{0.0025,0.010,0.015,0.020,0.025,0.030,0.035,0.040\}.$$

Then eight training data sets each consists of 100 samples for different $S_e$ are generated. Figure 1 shows a data set for which $S_e$ equals to 0.010. A testing data set consisting of another 1001 noise-free samples, for which $x$ equals to $-5,-4.99,\cdots,4.99,5$, are generated for validation.

For each $S_e$, two sets of RBF networks are trained. The first set is trained by Bernier *et al* explicit regularization method :

$$J(\theta)=\frac{1}{N}\sum_{k=1}^{N}(y_k-\phi^T(x_k)\theta)^2+\frac{S_\beta}{N}\theta^T\sum_{k=1}^{N}G(x_k)\theta \tag{18}$$

While the second set is trained by our method, Equation (14). In each set of networks, 11 different RBFs corresponding to 11 different values of $S_\beta$ as listed below are trained.

$$\{0,0.01^2,0.02^2,\cdots,0.09^2,0.1^2\}.$$

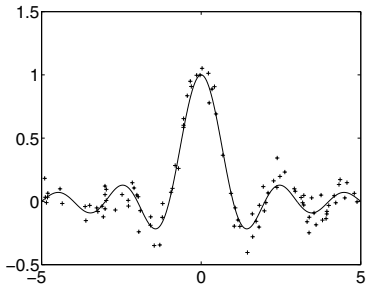


Fig. 1. The training and testing datasets for the sinc function experiment.

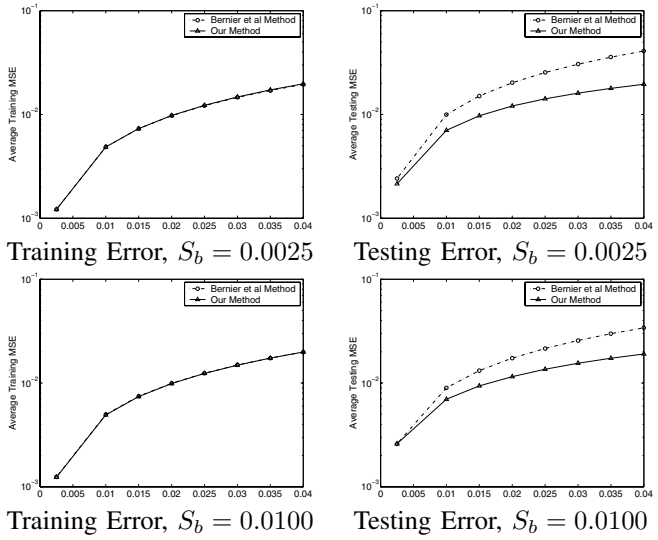


Fig. 2. The average training and testing MSE against $S_e$.

For the Bernier *et al* regularization method, For our method, as the objective function is not quadratic, the model is obtained by the gradient descent algorithm defined in Equation (15) and $\mu$ is set to 0.01.

Suppose a network model trained for a specified $S_e$ and $S_\beta$ has been obtained and the weight vector is given by $\hat{\theta}$, 10000 random networks are generated in accordance with $\hat{\theta}$ and the following equation.

$$\tilde{\theta}=\left((1+\beta_1)\hat{\theta}_1,(1+\beta_2)\hat{\theta}_2,\cdots,(1+\beta_M)\hat{\theta}_M\right)^T,$$

where $\beta_i\sim\mathcal{N}(0,S_\beta)$ for all $i=1,\cdots,M$. Their training mean square errors (MSE) and testing mean square errors are measured by fitting the network model $\tilde{\theta}$ to the training and testing datasets corresponding to the specific $S_e$.

To highlight the performance differences, the average MSE the values $S_e$ are plotted in Figure 2. The dot-dash lines with circles are corresponding to the results obtained by using Bernier *et al* method. While the solid lines with triangles are corresponding to the results obtained by our method. Owing to the page limit, we only show the case when $S_b$ equals 0.0025 and 0.01. It is clear from Figure 2, both methods can generate neural networks fitting well to the training data. Both methods give almost identical performance. But their performances in the testing datasets are different. Our method works better than the one obtained by the Bernier *et al* method.

## V. Discussion

Before making a conclusion of the paper, a few remarks should be added. First, the objective function derived here is similar but not identical to the one Amari derived for a stochastic perceptron learning in p.1398 of [1]. In [1], the noise is an additive white noise which corrupts on the output of hidden nodes and output nodes. While the noise we consider here is a multiplicative weight noise corrupted only on the output weights.

Second, the objective function derived in this paper can equally be applied to simple additive weight noise. For the weights that are corrupted by additive white noise of mean zero variance $S_\beta$, $S(x_k, \theta)$ will reduce to $S_e + S_\beta \theta^T \theta$. Then, applying gradient descent method, a training algorithm similar to the one presented in this paper can be accomplished. If furthermore we assume $S_\beta \ll S_e$, it can readily be shown that the objective function in Equation (14) can be reduced to the well known weight decay objective function [19].

Third, our objective function is in fact related to some other Bayesian type objective functions. In our paper, we have assumed that there is no prior information on the distribution of $\theta$. If $P(\theta)$ is known in advance and assumed that the dataset $\mathcal{D}$ is generated by a set of fault models with conditional probability $P(\tilde{\theta}|\theta)$, the estimation of $\theta$ can thus be obtained by Bayesian Modeling Averaging (BMA) technique [14], [23]. BMA is essentially an extension of Bayesian inference to estimate a model with parameters uncertainty.

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \quad (19)$$

$$P(\mathcal{D}|\theta) = \int P(\mathcal{D}|\tilde{\theta}, \theta)P(\tilde{\theta}|\theta)d\tilde{\theta}. \quad (20)$$

In such case, an RBF can be obtained by maximizing the $P(\theta|\mathcal{D})$, i.e.

$$\log P(\theta) - \log P(\mathcal{D}) + \sum_{k=1}^{N} \log \left( \int P(x_k, y_k|\tilde{\theta}, \theta)P(\tilde{\theta}|\theta)d\tilde{\theta} \right),$$

where $N$ is the total number of data in the set $\mathcal{D}$. Compare the above objective function with ours, the difference just in the *a priori* term $\log P(\theta)$. Therefore, our method can be treated as a special case of BMA method in which $P(\theta)$ is assumed to be a constant.

## VI. Conclusion

In this paper, we have derived from a new approach – KL Divergence – an objective function for training a RBF network that is able to tolerate multiplicative weight noise. Suppose a RBF network is suffered from multiplicative output-weight noise, simulation results have indicated that the RBF trained by the new objective function outperforms the one trained by using Bernier *et l*'s explicit regularizer. As KL divergence has very close relation to Bayesian learning, a discussion has been included in the end of the paper to elucidate the relations amongst our proposed objective function and the other Bayesian type objective functions.


## Acknowledgement

The work presented in this paper is supported by a research grant from the City University of Hong Kong under Project (No. 7001850) and a grant from NSC Taiwan Project Grant (No. NSC 95-2221-E-040-009).


## References


[1] Amari S.I., Information geometry of the EM and em algorithms for neural networks, *Neural Networks*, Vol.8(9), 1379-1408, 1995.

[2] Bernier J.L. *et al*, An accurate measure for multiplayer perceptron tolerance to weight deviations, *Neural Processing Letters*, Vol.10(2), 121-130, 1999.

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

[4] 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.

[5] 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.

[6] 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.

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

[8] Burr J.B., Digital neural network implementation, in *Neural Networks: Concepts, Applications, and Implementations*, Volume 2, P. Antognetti and V. Milutinovic (eds.), 237-285, Prentice Hall, 1991.

[9] Catala M. A. and X.L. Parra, Fault tolerance parameter model of radial basis function networks, *IEEE ICNN'96*, Vol.2, 1384-1389, 1996.

[10] Cavalieri S. and O. Mirabella, A novel learning algorithm which improves the partial fault tolerance of multilayer neural networks, *Neural Networks*, Vol.12, 91-106, 1999.

[11] Chen S., Local regularization assisted orthogonal least square regression, *Neurocomputing*, 559-585, 2006.

[12] Choi J.Y. and C.H. Choi, Sensitivity of multilayer perceptrons with differentiable activation functions, *IEEE Transactions on Neural Networks*, Vol. 3, 101-107, 1992.

[13] Fontenla-Romero O. *et al*, A measure of fault tolerance for functional networks, *Neurocomputing*, Vol.62, 327-347, 2004.

[14] Hoeting J.A. *et al*, Bayesian model averaging : A tutorial, *Statistical Science*, Vol.14, No.4, 382-417, 1999.

[15] Holt J. and J. Hwang, Finite precision error analysis of neural networks hardware implementations, *IEEE Transactions on Computers*, Vol.42, 281-290, 1993.

[16] 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.

[17] Kullback S., *Information Theory and Statistics*, Wiley, 1959.

[18] Lam P.M., C.S. Leung and T.T. Wong, Noise-resistant fitting for spherical harmonics, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 12(2), 254 - 265, March-April 2006

[19] 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.

[20] 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.

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

[22] Piche S.W., The selection of weight accuracies for Madalines, *IEEE Transactions on Neural Networks*, Vol.6, 432-445, 1995.

[23] Santafe G., J.A. Lozano and P. Larranaga, Bayesian model averaging of naive bayes for clustering, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 36, No. 5, 1149-1161, 2006.

[24] Simon D., Distributed fault tolerance in optimal interpolative nets, *IEEE Transactions on Neural Networks*, Vol.12(6), 1348-1357, 2001.

[25] Stevenson M., R. Winter and B. Widrow, Sensitivity of feedfoward neural networks to weight errors, *IEEE Transactions on Neural Networks*, Vol.1, 71-80, 1990.

[26] Townsend N.W. and L. Tarassenko, Estimations of error bounds for neural network function approximators, *IEEE Transactions on Neural Networks*, Vol.10(2), 217-230, 1999.

[27] Vapnik V. and S. Golowich and A. Smola, Support vector method for function approximation, regression estimation, and signal processing, *Neural Information Processing Systems*, Vol. 9, 281-287", 1997.