

Training Multilayer Perceptron By Using Optimal Input Normalization

Xun Cai

School of Computer Science & Technology
Shandong University
Jinan, Shandong, P.R.China
caixunzh@sdu.edu.cn

Kanishka Tyagi

Department of Electrical Engineering
The University of Texas at Arlington
Arlington, TX, US
kanishka.tyagi@mavs.uta.edu

Michael T.Manry

Department of Electrical Engineering,
The University of Texas at Arlington
Arlington, TX, US
manry@uta.edu

Abstract— In this paper, we propose a novel second order paradigm called *optimal input normalization* (OIN) to solve the problems of slow convergence and high complexity of MLP. By optimizing the non-orthogonal transformation matrix of input units in an equivalent network, OIN absorbs separate optimal learning factor for each synaptic weight as well as the threshold of hidden unit, leading to an improvement in the performance for MLP training. Moreover, by using a whitening transformation of negative Jacobian matrix of hidden weights, a modified version of OIN called *optimal input normalization with hidden weights optimization* (OIN-HWO) is also proposed. The Hessian matrices in both OIN and OIN-HWO are computed by using *Gauss-Newton* method. All the linear equations are solved via *orthogonal least square* (OLS). Regression simulations are performed on several real-life datasets and the results show that the proposed OIN has not only much better convergence rate and generalization ability than *output weights optimization-back propagation* (OWO-BP), *optimal input gains* (OIG) and even *Levenberg-Marquardt* (LM) method, but also takes less computational time than OWO-BP. Although OIN-HWO takes a little expensive computational burden than OIN, its convergence rate is faster than OIN and often close to or rivals LM. It is therefore suggested that OIN-based algorithms are potentially very good choices for practical applications.

Keywords- *multilayer perceptron (MLP), orthogonal least square (OLS), Gauss-Newton, output weights optimization back propagation (OWO-BP), optimal learning factor (OLF), hidden weights optimization (HWO), Schmidt procedure*

I. INTRODUCTION

The *multilayer perceptron* (MLP) has several favorable properties, such as universal approximation[1] and the ability to mimic Bayes discriminant [2] which make it a popular tool for many regression and classification applications including finance and weather predication [3,4], automatic controller [5], image and documents retrieval [6,7] and data mining [8].

A. Notation and Primary of MLP

Given a dataset with N_v training patterns $\{(\mathbf{x}_p, \mathbf{t}_p)\}$, where $p=1$ to N_v denotes the index of pattern, the typical topology of a fully-connected feed-forward MLP which consists of

three layers, namely, input layer, hidden layer and output layer is shown in Figure 1. The numbers of units in input layer, hidden layer and output layer are denoted as N , N_h and M , respectively. All these three layers are connected to each other by synaptic weights.

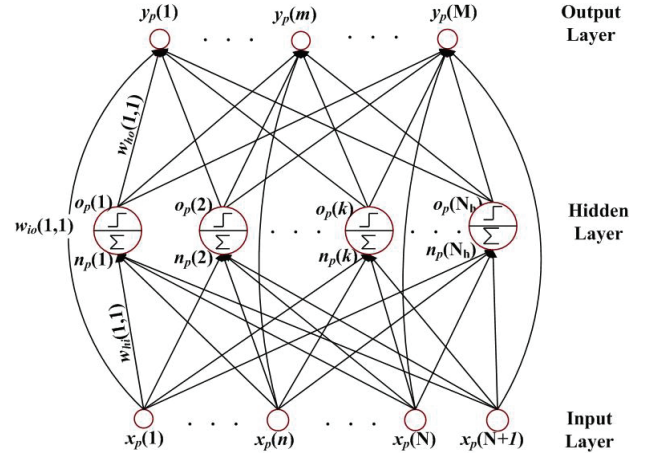


Figure 1. The architecture of a fully connected feedforward MLP

In Figure 1, $\mathbf{x}_p=[x_p(1), x_p(2), \dots, x_p(N+1)]^T$ denotes the input unit vector of p^{th} pattern which consists of $N+1$ input units, where $x_p(N+1)$ is an augmented input unit for conveniently handling the threshold of each hidden unit by setting $x_p(N+1)=1$ and $\mathbf{t}_p=[t_p(1), t_p(2), \dots, t_p(M)]^T$ denotes the corresponding desired output vector of p^{th} pattern which consists of M desired output units.

If $i=1$ to $(N+1)$, $m=1$ to M and $k=1$ to N_h , then $\mathbf{W}_{ih}=\{w_{ih}(i,k)\}$ denotes all the hidden weights linking from input units to hidden units with N by N_h dimension. $\mathbf{W}_{ho}=\{w_{ho}(k,m)\}$ represents all the output weights linking from hidden units to output units with N_h by M dimension. For full-connected feed-forward MLP, there are another kind of weights directly link from input units to output units which called bypass weights $\mathbf{W}_{io}$. All the bypass weights $\{w_{io}(i,m)\}$, are denoted as $\mathbf{W}_{io}$ with N by M dimension. Then, the net function of hidden units is denoted as $\mathbf{n}_p=[n_p(1), n_p(2), \dots, n_p(N_h)]^T$. For k^{th} hidden unit, $n_p(k)$ is denoted as

This paper is sponsored by National Natural Science Foundation of China (No. 60970047) and Innovation foundation of Shandong University (No. 2010TS001).

$$n_p(k) = \sum_{i=1}^{N+1} w_{ih}(i,k) x_p(i) = (\mathbf{W}_{ih})^T \mathbf{x}_p \quad (1)$$

Thus, if the corresponding output vector of hidden units which called activation function is denoted as $\mathbf{O}_p = [O_p(1), O_p(2), \dots, O_p(N_h)]^T$, then the output value of k^{th} hidden unit, $O_p(k)$ is given by

$$O_p(k) = f(n_p(k)) \quad (2)$$

where, $f(\cdot)$ denotes the nonlinear hidden layer activation function, which generally is taken as sigmoid function

$$f(n_p(k)) = \frac{1}{1 + e^{-n_p(k)}} \quad (3)$$

Finally, the output vector of MLP $\mathbf{y}_p = [y_p(1), y_p(2), \dots, y_p(M)]^T$ is summarized as,

$$\mathbf{y}_p = (\mathbf{W}_{io})^T \mathbf{x}_p + (\mathbf{W}_{ho})^T \mathbf{O}_p \quad (4)$$

For batch mode training of MLP, the typical error function $\mathbf{E}$ is *mean squared error* (MSE) which is defined as

$$\mathbf{E} = \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{m=1}^M [t_p(m) - y_p(m)]^2 \quad (5)$$

where $t_p(m)$ denotes the desired value and actual value of m^{th} output unit, respectively. If $\mathbf{W}$ denotes all the synaptic weight matrix, then the output vector, $\mathbf{y}_p$ is the function of $\mathbf{W}$ then the general main purpose of training MLP is to optimize the weight matrix, $\mathbf{W}$, by minimizing the MSE function, $\mathbf{E}$, based on some optimal training rules. In such case, $\mathbf{E}$ can also be represented as $\mathbf{E}(\mathbf{W})$.

B. Gradient Descent Algorithm

One of the earliest training algorithms for optimizing weights is gradient descent back propagation algorithm [10]. In such case, the hidden weight matrix, $\mathbf{W}_{ih}$, is updated as

$$\mathbf{W}_{ih} \leftarrow \mathbf{W}_{ih} + z \cdot \mathbf{G}_{ih} \quad (6)$$

where $\mathbf{G}_{ih} = \{g_{ih}(n, k)\}$ is a $(N+1)$ by N_h negative Jacobian matrix of hidden weights which is given by

$$\mathbf{G}_{ih} = \frac{1}{N_v} \sum_{p=1}^{N_v} \mathbf{x}_p (\delta_p)^T \quad (7)$$

where $\delta_p = [\delta_p(1), \delta_p(2), \dots, \delta_p(N_h)]^T$. $\delta_p(k)$ is the delta function corresponding to for k^{th} hidden unit and is formulated as

$$\delta_p(k) = f'(n_p(k)) \cdot \sum_{m=1}^M \delta_o(m) \cdot w_{ho}(k, m) \quad (8)$$

where $[\delta_o(1), \delta_o(2), \dots, \delta_o(M)]^T$ is denoted as δ_o which represent the delta function for output units and is normally given by

$$\delta_o = 2(\mathbf{t}_p - \mathbf{y}_p) \quad (9)$$

Let $\mathbf{W}_o = [\mathbf{W}_{io}, \mathbf{W}_{ho}]$ denote the combined output weight matrix with $(N+1+N_h)$ by M dimension, then the equation for updating $\mathbf{W}_o$ is given by

$$\mathbf{W}_o \leftarrow \mathbf{W}_o + z_1 \mathbf{G}_o \quad (10)$$

where z_1 is the learning factor and $\mathbf{G}_o$ is the gradient matrix of error function with respect to $\mathbf{W}_o$.

If $\mathbf{X}_p = [\mathbf{x}_p; \mathbf{O}_p]$ with $(N+N_h+1)$ by 1 dimension, then $\mathbf{G}_o$ is formulated as

$$\mathbf{G}_o = \frac{1}{N_v} \sum_{p=1}^{N_v} \mathbf{X}_p (\delta_o)^T \quad (11)$$

From (6) and (10), we can easily find that the learning factors, z and z_1 have the directly effects on the convergence rate of MLP.

To choose appropriate z and z_1 , many researchers have suggested that learning factor should be optimized separately for different synaptic weights [11]~[14]. However, for many MLP training algorithms, the definitions of learning factor z and z_1 were either assigned to be the fixed values or tuned empirically. Although these methods can show fast convergence rates for some specific applications, they are suffered from lots of efforts to find out the proper learning factors. Also, these methods have some error oscillations during training.

C. Output Weights Optimization Back Propagation (OWO-BP)

The OWO-BP algorithm is a fully optimal learning method which convergences fast without any error oscillation [15]. The OWO-BP algorithm includes two stages: the *output weights optimization* (OWO) and *back propagation* (BP).

In OWO stage, since the output units have linear activation functions, the OWO procedure is equal to solve a set of linear equations. In this case, the output weight vector $\mathbf{W}_o$ is given as follows

$$\mathbf{W}_o = \mathbf{R}_o \mathbf{C}_o^T \quad (12)$$

Here, $\mathbf{R}_o$ is the autocorrelation matrix of $\mathbf{X}_p$ with $(N+N_h+1)$ by $(N+N_h+1)$ dimension and $\mathbf{C}_o$ is the correlation matrix of between output vector $\mathbf{y}_p$ and $\mathbf{X}_p$ with dimension of M by $(N+N_h+1)$. The calculations of $\mathbf{R}_o$ and $\mathbf{C}_o$ are given respectively as

$$\begin{cases} \mathbf{C}_o = \frac{1}{N_v} \sum_{p=1}^{N_v} \mathbf{y}_p \mathbf{X}_p^T \\ \mathbf{R}_o = \frac{1}{N_v} \sum_{p=1}^{N_v} \mathbf{X}_p \mathbf{X}_p^T \end{cases} \quad (13)$$

In this case, (12) can be directly solved by using orthogonal least square (OLS) which is achieved by Schmidt procedure[15].

In BP phase, the learning factor z is optimized which introduces the concept of *optimal learning factor* (OLF) z and the optimization of z for all the hidden weight is given by Newton's method as

$$z = -\frac{\frac{\partial \mathbf{E}}{\partial z}}{\frac{\partial^2 \mathbf{E}}{\partial z^2}} \quad (14)$$

where $\frac{\partial \mathbf{E}}{\partial z}$ and $\frac{\partial^2 \mathbf{E}}{\partial z^2}$ are the first order and the second order derivative of error function, with respect to the learning factor, z , which are computed, respectively as,

$$\begin{cases} \frac{\partial \mathbf{E}}{\partial z} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M (t_p(i) - y_p(i)) \cdot \frac{dy(i)}{dz} \\ \frac{\partial^2 \mathbf{E}}{\partial z^2} = \frac{2}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M \left[\frac{dy(i)}{dz} \right]^2 \end{cases} \quad (15)$$

where $\frac{dy(i)}{dz}$ is derived by following equation

$$\frac{dy(i)}{dz} = \sum_{k=1}^{N_h} w_{oh}(i, k) f'(n(k)) \sum_{n=1}^{N+1} g(k, n) x_p(n) \quad (16)$$

As the training of output weights is equivalent to solving linear equations and the learning factor z is optimized by using second order Newton's method, OWO-BP performs much better than gradient descent based BP with much less computations than LM. However, as OWO-BP only computes single OLF for all hidden weights, it cannot guarantee it is the global optimal factor for the entire error surface. Hence, OWO-BP doesn't exhibit faster convergence rate than LM.

D. The Plain Levenberg-Marquardt Algorithm

The plain *Levenberg-Marquardt* (LM) algorithm [17]~[20] is a sub-optimal learning factor method which updates all synaptic weights by following formulation

$$\mathbf{W} \leftarrow \mathbf{W} - (\mathbf{H} + \lambda \cdot \mathbf{I})^{-1} \nabla \mathbf{f}(\mathbf{W}) \quad (17)$$

where $\mathbf{H}$ is Hessian matrix of error function with respect to $\mathbf{W}$ with $(N+N_h+1)$ by $(N+N_h+1)$ dimension, λ is a controlling factor for tuning the direction of method towards first order or second order method, $\nabla \mathbf{f}(\mathbf{W})$ is gradient matrix of a quadratic function of $\mathbf{W}$ and $\mathbf{I}$ is identity matrix. As LM algorithm optimizes all the synaptic weights by using the second order Newton's method, this algorithm converges

significantly better than many other MLP training algorithms. However, due to the huge computations for calculate Hessian matrix $\mathbf{H}$ and its inversion matrix, which costs $O(N^3)$, as well as lots of efforts to choose proper value of λ , LM suffers from huge computations leading to impractical application.

E. Optimal Input Gains algorithm

Compared with LM and OWO-BP, *optimal input gains* (OIG) algorithm is a fully optimal learning algorithm which embeds optimal learning factors into so-called optimal input gains [21]. OIG introduces a novel way to update hidden weights by optimizing an input unit transformation matrix in an equivalent MLP network. This transformation matrix is shown as

$$\mathbf{A} = \begin{bmatrix} a(1) & 0 & \cdots & 0 & 0 \\ 0 & a(2) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a(N) & 0 \\ 0 & 0 & \cdots & 0 & a(N+1) \end{bmatrix} \quad (18)$$

where $[a(1), a(2), \dots, a(N+1)]$ is called input gain vector and is denoted as $\mathbf{a}$. Hence, the input transformation equation is given by

$$\bar{\mathbf{x}}_p = \mathbf{A} \mathbf{x}_p \quad (19)$$

By optimizing input gain vector $\mathbf{a}$, the hidden weight can be updated as,

$$\mathbf{W}_{ih} \leftarrow \mathbf{W}_{ih} + \mathbf{a} \cdot \mathbf{G}_{ih} \quad (20)$$

Compared (20) with (6), it can be easily seen that the input gain vector, $\mathbf{a}$, in fact takes the role of learning factor for training hidden weight matrix $\mathbf{W}_{ih}$.

Due to absorbing optimal learning factors into optimal input gains instead of obtaining optimal learning factor z directly, OIG converges much faster than OWO-BP with much less computation burden than LM.

In this paper, we proposed a more generalized optimal transformation matrix of input units. By optimizing this transformation matrix, a new algorithm which called *optimal input normalization* (OIN) improves the convergent rate and generalization than OIG.

The rest of paper is organized as follows: in the section II, we present the motivation for OIN and its mathematic derivation as well as the improved version of OIN which called OIN-HWO. In section III, the algorithm procedure of OIN as well as OIN-HWO is represented. In section IV, the simulation results on several benchmarks are analyzed. In section V, the conclusions and some suggestions on future enhancements are present. Acknowledgements and references are given in section VI and section VII, respectively.

II. OPTIMAL INPUT NORMALIZATION (OIN)

A. Motivation for OIN

The motivation for OIN comes from the well-known fact that if the input units are normalized to zero-mean, then the training error will be improved by non-orthogonal transformation of input units [22]. Thus, the non-orthogonal transformation matrix $\mathbf{A}$ is expressed as

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \cdots & 0 & -m(1) \\ 0 & 1 & \cdots & 0 & -m(2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -m(N) \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad (21)$$

where $[m(1), (2) \cdots m(N)]$ is the mean vector of the input units. Hence, the n^{th} transformed input unit $\bar{x}_p(n)$ is given by

$$\bar{x}_p(n) = x_p(n) + m(n) \quad (22)$$

where $n=1 \cdots N$. This leads to the following lemma for the transformation of input unit to an equivalent network which is defined as [20].

Lemma: Consider a nonlinear network with input unit vector $\mathbf{x}_p \in R^{N+1}$, with the restriction $\mathbf{x}_p(N+1) = 1$ and output vector $\mathbf{y} \in R^M$ and a second network in which the input unit vector $\bar{\mathbf{x}}_p = \mathbf{A} \cdot \mathbf{x}_p$ has corresponding output vector $\bar{\mathbf{y}}_p \in R^M$ then two networks are strongly equivalent if $\forall \mathbf{x}_p \in R^{N+1}, \bar{\mathbf{y}}_p = \mathbf{y}_p$.

Based on the above theory, the negative Jacobian matrix of equivalent transformed network, $\bar{\mathbf{G}}_{ih}$ is given by derived from the original network, $\mathbf{G}_{ih}$, where

$$\bar{\mathbf{G}}_{ih} = \frac{1}{N_v} \sum_{p=1}^{N_v} \bar{\mathbf{x}}_p (\delta_p)^T = \frac{1}{N_v} \sum_{p=1}^{N_v} (\mathbf{A} \mathbf{x}_p) (\delta_p)^T = \mathbf{A} \mathbf{G}_{ih} \quad (23)$$

If we map the transformed input units back to the original network, then the relation between the negative Jacobian matrix of mapped MLP network, which is denoted as $\bar{\mathbf{G}}_{ih}$, and $\mathbf{G}_{ih}$ is described as

$$\bar{\mathbf{G}}_{ih} = (\mathbf{A}^T \mathbf{A}) \mathbf{G}_{ih} = \bar{\mathbf{A}} \mathbf{G}_{ih} \quad (24)$$

where $\bar{\mathbf{A}} = \mathbf{A}^T \mathbf{A}$ is a new transformed matrix with $(N+1)$ by $(N+1)$ dimension.

From (24), we can infer that only if $\mathbf{A}$ is non-orthogonal matrix, then $\bar{\mathbf{A}}$ could be useful for transformation from $\mathbf{G}_{ih}$ to $\bar{\mathbf{G}}_{ih}$. In this case, the mapped back MLP will not be equal to the original MLP. Hence, by optimizing the non-orthogonal $\mathbf{A}$, the training error of the mapped back MLP could be decreased more rapidly than that of the original one. This motivates us to propose the current OIN model.

Compared to OIG, the transformation matrix $\mathbf{A}$ of our proposed paradigm OIN has a more generalized form as

$$\mathbf{A} = \begin{bmatrix} a(1) & 0 & \cdots & 0 & b(1) \\ 0 & a(2) & \cdots & 0 & b(2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a(N) & b(N) \\ 0 & 0 & \cdots & 0 & a(N+1) \end{bmatrix} \quad (25)$$

Here, $[a(1), a(2), \dots, a(N+1)]$ and $[b(1), b(2), \dots, b(N)]$ are called input gain vector which denoted as $\mathbf{a}$ and input bias vector which denoted as $\mathbf{b}$, respectively. In these case, the $\bar{\mathbf{A}}$ is derived into the following symmetrical formation

$$\begin{aligned} \bar{\mathbf{A}} &= \mathbf{A}^T \mathbf{A} \\ &= \begin{bmatrix} a(1)a(1) & 0 & \cdots & 0 & a(1)b(1) \\ 0 & a(2)a(2) & \cdots & 0 & a(2)b(2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a(N)a(N) & a(N)b(N) \\ b(1)a(1) & b(2)a(2) & \cdots & b(N)a(N) & \sum_{n=1}^N a(n)b(n) \end{bmatrix} \\ &= \begin{bmatrix} \bar{a}(1) & 0 & \cdots & 0 & \bar{b}(1) \\ 0 & \bar{a}(2) & \cdots & 0 & \bar{b}(2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \bar{a}(N) & \bar{b}(N) \\ \bar{b}(1) & \bar{b}(2) & \cdots & \bar{b}(N) & \bar{a}(N+1) \end{bmatrix} \end{aligned} \quad (26)$$

By substituting $\mathbf{G}_{ih}$ with $\bar{\mathbf{G}}_{ih}$ in (6), the hidden weights of the mapped back MLP, $\bar{w}_{ih}(n, k)$ and $\bar{w}_{ih}(N+1, k)$, where $n=1$ to $N, k=1$ to N_h , are updated, respectively, as

$$\begin{cases} \bar{w}_{ih}(n, k) \leftarrow \bar{w}_{ih}(n, k) + z \cdot [\bar{a}(n)g_{ih}(n, k) + \bar{b}(n) \cdot g_{ih}(N+1, k)] \\ \bar{w}_{ih}(N+1, k) \leftarrow \bar{w}_{ih}(N+1, k) + z \cdot [\bar{a}(N+1)g_{ih}(N+1, k) + \sum_{j=1}^N \bar{b}(j) \cdot g_{hi}(j, k)] \end{cases} \quad (27)$$

If $z \cdot \bar{a}(n)$ and $z \cdot \bar{b}(n)$ are denoted as $\bar{\bar{a}}(n)$ and $\bar{\bar{b}}(n)$ respectively, then by comparing (27) with (6), it can be easily inferred that $\bar{\bar{a}}(n)$ and $\bar{\bar{b}}(n)$ absorb the learning factor z for each hidden weight of hidden unit as well as the threshold of hidden units. This results into the *optimal input normalization* (OIN) algorithm which gives more concern for thresholds of hidden units more than OIG.

B. Derivation of OIN

If $[\bar{a}(1), \bar{a}(2), \dots, \bar{a}(N+1)]^T$ is denoted as $\mathbf{a}$ and $[\bar{b}(1), \bar{b}(2), \dots, \bar{b}(N+1)]^T$ is denoted as $\mathbf{b}$, then $[\mathbf{a}; \mathbf{b}]$ is denoted as $\mathbf{ab}$ which is a vector with $(2N+1)$ by 1 dimension. Here, we use Gauss-Newton's method to optimize $\mathbf{ab}$, which is formulated as

$$\mathbf{ab} = \mathbf{H}_{ab}^{-1} \mathbf{G}_{ab} \quad (28)$$

where $\mathbf{H}_{ab} = \begin{bmatrix} \mathbf{h}_{aa} & \mathbf{h}_{ab} \\ \mathbf{h}_{ba} & \mathbf{h}_{bb} \end{bmatrix}$ denotes the Hessian matrix with dimension of $(2N+1)$ by $(2N+1)$ and $\mathbf{G}_{ab} = [\mathbf{G}_a : \mathbf{G}_b]$ denotes the negative Jacobian matrix with dimension of $(2N+1)$ by 1, where $\mathbf{G}_a$ and $\mathbf{G}_b$ are expressed, respectively, as

$$\begin{cases} \mathbf{G}_a = \frac{\partial \mathbf{E}}{\partial \mathbf{a}} = -\frac{2}{N_v} \cdot \left[\frac{\partial \mathbf{y}_p}{\partial \mathbf{a}} \right]^T [\mathbf{t}_p - \mathbf{y}_p] \\ \mathbf{G}_b = \frac{\partial \mathbf{E}}{\partial \mathbf{b}} = -\frac{2}{N_v} \cdot \left[\frac{\partial \mathbf{y}_p}{\partial \mathbf{b}} \right]^T [\mathbf{t}_p - \mathbf{y}_p] \end{cases} \quad (29)$$

From $m=1$ to $(N+1)$, $u=1$ to N , then the each element of $\mathbf{h}_{ab}$, $\mathbf{h}_{ba}$, $\mathbf{h}_{aa}$ and $\mathbf{h}_{bb}$ are denoted as $h_{ab}(m, u)$, $h_{ba}(u, m)$, $h_{aa}(m, m)$ and $h_{bb}(u, u)$, respectively and are calculated as,

$$\begin{cases} h_{ab}(m, u) = h_{ba}(u, m) = \frac{\partial^2 \mathbf{E}}{\partial \bar{a}(m) \partial \bar{b}(u)} \approx 2 \cdot \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M \frac{\partial y_p(i)}{\partial \bar{a}(m)} \cdot \frac{\partial y_p(i)}{\partial \bar{b}(u)} \\ h_{aa}(m, m) = \frac{\partial^2 \mathbf{E}}{\partial \bar{a}(m)^2} \approx 2 \cdot \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M \frac{\partial y_p(i)}{\partial \bar{a}(m)} \cdot \frac{\partial y_p(i)}{\partial \bar{a}(m)} \\ h_{bb}(u, u) = \frac{\partial^2 \mathbf{E}}{\partial \bar{b}(u)^2} \approx 2 \cdot \frac{1}{N_v} \sum_{p=1}^{N_v} \sum_{i=1}^M \frac{\partial y_p(i)}{\partial \bar{b}(u)} \cdot \frac{\partial y_p(i)}{\partial \bar{b}(u)} \end{cases} \quad (30)$$

The net function, $n_p(k)$ in (1) is rewritten as

$$n_p(k) = \sum_{n=1}^N \bar{w}_{hi}(n, k) \cdot x_p(n) + \bar{w}_{hi}(N+1, k) \quad (31)$$

By substituting (31) and (27) into (4) and differentiating the first order of $\mathbf{y}_p$ with respect of $\mathbf{a}$ and $\mathbf{b}$, we obtain

$$\frac{\partial y_p(i)}{\partial \bar{a}(m)} \text{ and } \frac{\partial y_p(i)}{\partial \bar{b}(u)} \text{ by following equations}$$

$$\begin{cases} \frac{\partial y_p(i)}{\partial \bar{a}(m)} = \sum_{k=1}^{N_h} w_{ho}(k, i) \cdot O'_p(k) \cdot [g_{ih}(m, k) \cdot x_p(m)] \\ \frac{\partial y_p(i)}{\partial \bar{b}(u)} = \sum_{k=1}^{N_h} w_{ho}(k, i) \cdot O'_p(k) \cdot [g_{ih}(N+1, k) \cdot x_p(u) + g_{ih}(u, k)] \end{cases} \quad (32)$$

where $O'_p(k) = f'(n(k))$.

As $\mathbf{W}_{ho}$ can be obtained by using OWO from (12) shows, $\mathbf{H}_{ab}$ and $\mathbf{G}_{ab}$ in (28) can be calculated by (29)(30) and (32).

Finally, to obtain optimal $\mathbf{ab}$ in (28), here we use the *orthogonal least square* (OLS) method which is performed by using Schmidt procedure [15].

C. OIN-HWO

The concept behind HWO method [16] is that it uses separate error function for training each hidden weight. In such case, the error function of j^{th} hidden unit, $E_\delta(j)$, where j is from 1 to N_h , in HWO is expressed as

$$E_\delta(j) = \sum_{p=1}^{N_v} \left[\delta_p(j) - \sum_{n=1}^{N+1} g_{hwo}(j, n) x_p(n) \right]^2 \quad (33)$$

$\mathbf{G}_{hwo}$ is derived from Newton's method as

$$\mathbf{G}_{hwo} \mathbf{R} = \mathbf{G}_{ih} \quad (34)$$

where $\mathbf{R}$ is the input auto-correlation matrix of input unit vector $\mathbf{x}_p$. By this whitening transformation of $\mathbf{G}_{ih}$, the input dependence can be eliminated, which makes the $\mathbf{H}_{ab}$ is non-singular and the optimization $\mathbf{ab}$ is available. The set of linear equations in (34) is solved by using OLS. Thus, by substituting $\mathbf{G}_{ih}$ into $\mathbf{G}_{hwo}$, we can get a modified version of OIN called OIN-HWO.

III. PROCEDURE

The procedure of OIN in batch mode training is shown as follows:

Step1: Initialize input weight matrix $\mathbf{W}_{ih}$ by using normal distribution function and net control.

Step2: Use OWO by (12) to get the optimal output weight matrix, $\mathbf{W}_o$, initially at first iteration.

Step 3: If the iteration does not end, then use (7) and (8) to get the negative Jacobian matrix $\mathbf{G}_{ih}$ for OIN or use (7), (8) and (34) to get $\mathbf{G}_{hwo}$ for OIN-HWO; else go to Step 10.

Step 4: Solve (28) ~ (32) to obtain optimal input normalization vector, $\mathbf{ab}$, by using Schmidt procedure.

Step 5: Update the hidden weights of mapped back MLP by (27), $\bar{\mathbf{W}}_{ih}$, for OIN based on $\mathbf{G}_{ih}$ or $\mathbf{G}_{hwo}$.

Step 7: Use these new hidden weights to get new output values of hidden units, $\mathbf{O}_p$, by (2).

Step 8: Calculate new optimal output weight matrix $\mathbf{W}_o$ in (12).

Step 9: Combine new $\bar{\mathbf{W}}_{ih}$ and $\mathbf{W}_o$ to get actual output vector $\mathbf{y}_p$ in (4) and use (5) to get new training MSE, $\mathbf{E}$.

Step 10: Return to Step 3 and repeat the above steps until get to the end of iterations.

IV. EXPERIMENTAL RESULTS

We perform our simulations for two versions of OIN on several datasets and show the performance by comparing the results with OWO-BP and LM. All the simulations run on Intel Core2 Duo P8700. The number of training iterations for each simulation is fixed to 100, namely, $N_{it}=100$. For better comparison, all the synaptic weights are initialized using the same method which is given by section III Step (1)~(2). Thus, the training iteration will start after OWO and the first training error will start at the same point.

A. Description of Datasets

The datasets which we use are called "Twod", "Single2", "Mattn" and "Concrete", respectively. All these datasets used for simulation are publicly available and the input units

of all these datasets have been normalized to be zero-mean and unit variance..

The first three are from the Image Processing and Neural Networks Lab repository [23] and the last one is from UCI Machine Learning Repository [24,25]. Among these datasets, “Twod” is used to invert the surface scattering parameters from an inhomogeneous layer above a homogeneous half space, where both interfaces are randomly rough. “Mattrn” is used for inversion of random two-by-two matrices; “Single2” is used for inversion of surface permittivity, the normalized surface rms roughness, and the surface correlation length found in back scattering models from randomly rough dielectric surfaces. “Concrete” dataset is used to approximate the nonlinear function of age and ingredients of concrete compressive strength.

All of the numbers of patterns, input units, output units and hidden units of these datasets are denoted as N_v , N , M , and N_h , respectively, as listed in TABLE I. In order to learn how each algorithm deals with the co-linearity of each dataset, all the co-linearity of datasets are also listed in Table I.

TABLE I. THE STRUCTURE OF MLP AND CO-LINEARITY FOR EACH DATASET

Datasets	N_v	N	M	N_h	Co linearity
Twod	1768	8	7	10	high
Single2	10000	16	3	8	high
Mattrn	2000	4	4	12	high
Concrete	1030	8	1	15	low

Under the above training conditions, the training MSEs of OIN, OIN-HWO, OIG, OWO-BP and LM are plotted version the time spent after 100 iterations are shown in Figure 2, respectively.

B. Training Error Versus Computational Time

From Figure 2(a)~(d), we can see that OIN shows faster converging rates than OWO-BP and OIG, even rivals LM in (c) as well as OIG does. The total training time spent by OIN is a little longer than OIG but shorter than OWO-BP.

Also, it can be easily seen that OIN-HWO always shows better converge rate than OIN even close to or rivals LM as (a)~(c) shown. OIN-HWO takes a little longer training time than OIN, OIG and OWO-BP. However, compared with LM, the training time spent by OIN-HWO is much less.

C. K-fold Cross Validation

Furthermore, we use k-fold cross validation procedure to evaluate the generalization abilities of OIN and OIN-HWO with comparison with OIG, OWO-BP and, LM. Here, k=10. The procedure of k-fold cross validation is described as follows.

We split each dataset randomly into 10 non-overlapping parts of equal size and combine 9 parts of 10 for training leaving the remaining one part for testing. This procedure is repeated until we have exhausted all 10 combinations. Then, by training all these combinational datasets and then testing on the remaining dataset, we obtain the training MSE and validation MSE of each algorithm on each dataset. The results are listed in Table II.

From Table II, we can see that both training error and validation error of OIN are smaller than OIG and OWO-BP, even smaller than LM on “Single 2” dataset. OIN-HWO shows much better training and validation errors than OIN, even rivals LM on these four datasets. That means our proposed paradigm OIN and OIN-HWO can be seemed as a potential robust approach for practical applications.

V. CONCLUSION

In this paper, we present a novel MLP training algorithm called optimal input normalization (OIN) and its improved version OIN-HWO. Several simulations on real-life datasets and the k-fold cross validation show that the proposed OIN and OIN-HWO greatly improve the converging rates and generalization abilities of OIG, OWO-BP and even LM, respectively. Moreover, as they spend much less computational time than LM, OIN and OIN-HWO are potentially good choice for real life applications.

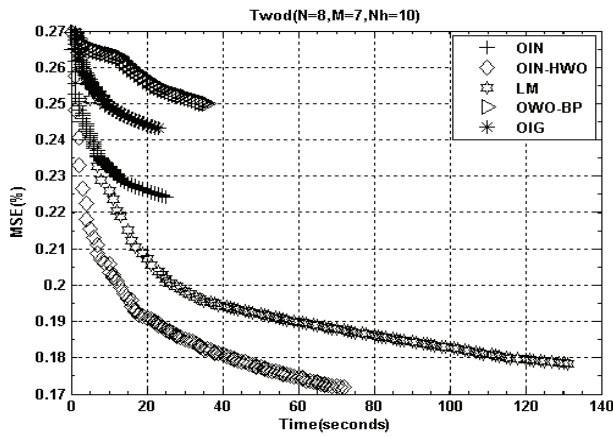
Future work will give more detailed analysis the effects of co-linearity of different dataset on OIN performance and develop the novel OIN algorithm combined with other network such as RBF, SVM etc..

VI. ACKNOWLEDGMENTS

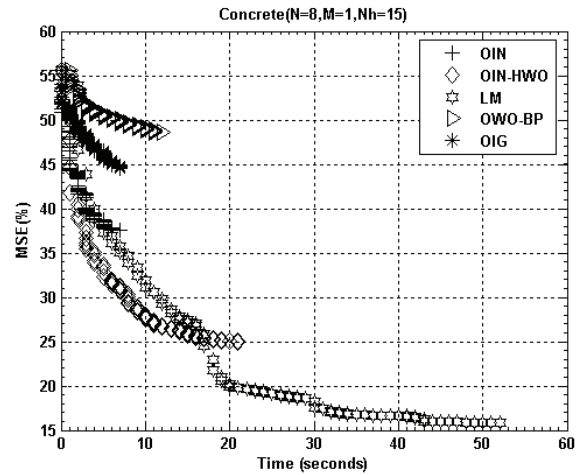
Our sincere thanks are given to the Image Processing and Neural Network Lab in the Department of Electrical Engineering at the University of Texas at Arlington.

TABLE II. K-FOLD CROSS VALIDATION

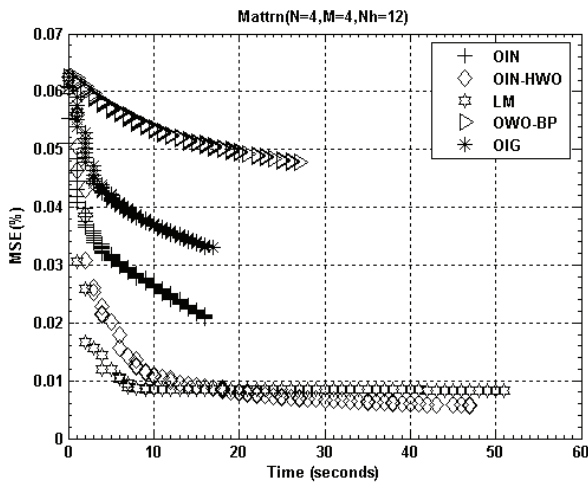
Datasets	T&V	OIG	OIN	OIN-HWO	LM	OWO-BP
Twod	T	0.240	0.224	0.165	0.200	0.248
	V	0.249	0.235	0.179	0.212	0.259
Single2	T	0.452	0.168	0.104	0.582	1.445
	V	0.588	0.432	0.130	0.780	1.479
Mattrn	T	0.021	0.017	0.004	0.014	0.039
	V	0.021	0.016	0.009	0.013	0.037
Concrete	T	41.93	38.268	13.813	19.600	50.228
	V	49.39	48.564	20.444	30.239	59.399



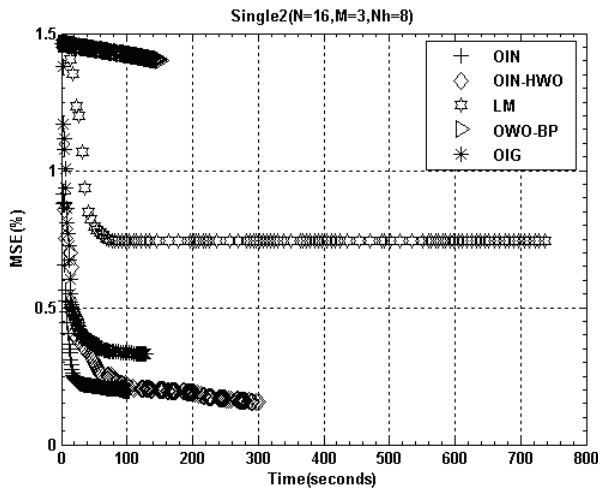
(a) "Twod"



(d) "Concrete"



(b) "Mattm"



(c) "Single2"

Figure 2. The training error versus time spent on various datasets

VII. REFERENCES

- [1] G. Cybenko, "Approximations by superpositions of a sigmoidal function," Mathematics of Control, Signals, and Systems (MCSS), vol. 2, pp. 303-314, 1989.
- [2] Dennis W. Ruck et al., "The multilayer perceptron as an approximation to a bayes optimal discriminant function," IEEE Transactions on Neural Networks, vol. 1, No. 4, pp.256-298, 1990.
- [3] Meesomsarn, K., Chaisrichaen, R., Chipipop, B. and Yooyativong, T. "Forecasting the effect of stock repurchase via an artificial neural network," ICROS-SICE International Joint Conference 2009, pp.2573 - 2578, Aug , 2009.
- [4] Jao-Hong Cheng, Huei-Ping Chen and Yi-Min Lin, "A hybrid forecast marketing timing model based on probabilistic neural network rough set and C4," Expert Systems with Applications, vol. 37, n.4, pp. 1814 - 1820, 2010.
- [5] Rong-Jong, Wai and Zhi-Wei, Yang, " Design of fuzzy-neural-network tracking control with only position feedback for robot manipulator including actuator dynamics," IEEE International Conference on Systems, Man and Cybernetics. (Taipei, China, Oct12-15, 2008), pp. 2349 - 2354, Oct., 2008,
- [6] Marinai, S., Gori, M., Soda, G., "Artificial neural networks for document analysis and recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, n.1, pp. 23-35, 2005.
- [7] Jyothi, B.V., Eswaran, K., "Comparative study of neural networks for image retrieval," International Conference on Intelligent Systems, Modelling and Simulation (Liverpool, UK, Jan 27-29, 2010), pp. 199 - 203, Jan . 2010.
- [8] .K.-L. Du. "Clustering: A neural network approach," Neural Networks, vol. 23, n. 1, pp. 89-107, 2010,.
- [9] Melacci, S.; Maggini, M.and Sarti, L., "Semi-supervised clustering using similarity neural networks," International Joint Conference on Neural Networks, (Atlanta, GA, June 14-19, 2009), pp.2065 - 2072, June 2009.
- [10] Rumelhart, D.E., Hinton, G.E.and Williams, R.J., "Learning internal representations by error propagation," Parallel distributed processing: explorations in the microstructure of cognition, Mit Press Computational Models Of Cognition And Perception Series, Cambridge, MA, USA, vol. 1: foundations.1, pp. 318 - 362, 1986.
- [11] Robert, A. Jacobs, "Increased rates of convergence through learning rate adaptation," Neural Networks. vol.1, n.4, pp.295-307, 1988.
- [12] Sang-Hoon Oh and Soo-Young Lee., "Optimal learning rates for each pattern and Neuron in gradient descent training of multilayer

- perceptrons,"International Joint Conference on Neural Networks (Washington, DC , USA 10 Jul 1999),pp. 1635 – 1638, Jul 1999.
- [13] Y. LeCunn, L.Bottou, G.B.Orr, K.R.Muller, "Efficient BackProp," Lecture Notes In Computer Science, vol. 1524, Springer-Verlag, London, pp. 9-50, 1998.
 - [14] Hestenes M., R., and Steifel, E., "Methods of conjugate gradients for solving linear systems." Journal of Research of the National Bureau of Standards. vol. 49, n.6, pp. 409-436, 1952.
 - [15] F. J. Maldonado, M. T. Manry and Tae-Hoon Kim,. "Finding optimal neural network basis function subsets using the Schmidt procedure" In Proceedings of the International Joint Conference on Neural Networks,(Portland, Oregon,2003),1, pp.444 – 449, July , 2003.
 - [16] H-H Chen, M.T. Manry, and H. Chandrasekaran, "A neural network training algorithm utilizing multiple sets of linear equations", Neurocomputing, vol. 25, pp. 1-3, 55-72, 1999,.
 - [17] Fun., M.H., and Hagan., M.T., " Levenberg-marquardt training for modular networks," Proc. Of IEEE International Conference on Neural Networks'96, Washington DC,pp.468-473, 1996.
 - [18] Hagan., M. T., and Mehna., M.B., "Training feedforward networks with the marquardt algorithim," IEEE Trans. Of Neural Networks, vol.5, pp. 989-993, Nov.1994.
 - [19] Levenberg, K.. " A method for the solution of certain problems in least squares," Quart. Appl. Math.,vol. 2, pp. 164.168, 1944,.
 - [20] Marquardt, D., " An algorithm for least-squares estimation of nonlinear parameters," SIAM J. Appl. Math., vol. 11, 431-441, 1963.
 - [21] Sanjeev S. Malalur, Manry, M. T., " Feed-forward network training using optimal input gains," Proceedings of International Joint Conference on Neural Network.(Atlanta, Georgia, USA, June 14-19, 2009),pp. 1958-1960, June 14-19, 2009
 - [22] Y. LeCun, "Efficient Learning and Second-Order Methods," A Tutorial at NIPS 93, Denver 1993.
 - [23] http://www-ee.uta.edu/ee/b/tp/training_data_files.htm
 - [24] <http://archive.ics.uci.edu/ml/>
 - [25] Cheng Yeh., " Modeling of strength of high performance concrete using artificial neural networks," Cement and Concrete Research,vol. 28,n.12, pp.1797-1808,1998,