Invariance of MLP Training to Input Feature De-correlation

Changhua Yu, Michael T. Manry, Jiang Li

Department of Electrical Engineering
University of Texas at Arlington, TX 76019
E-mail: changhua_yu@uta.edu; manry@uta.edu; li@wcn.uta.edu

Abstract

In the neural network literature, input feature de-correlation is often referred as one pre-processing technique used to improve the MLP training speed. However, in this paper, we find that de-correlation by orthogonal Karhunen-Loeve transform (KLT) may not be helpful to improve training. Through detailed analyses, the effect of input de-correlation is revealed to be equivalent to using a different weight set to initialize the network. Thus, for a robust training algorithm, the benefit of input de-correlation would be negligible. The theoretical results are applicable to several gradient training algorithms, i.e. back-propagation, conjugate gradient. The simulation results confirm our theoretical analyses.

Introduction

The multi-layer perceptron (MLP) is a popular tool for signal processing (Manry, Chandrasekaran and Hsieh 2001), remote sensing (Manry et al. 1994), and pattern recognition (Chen, Manry and Chandrasekaran 1999). Although its universal approximation ability has already been proved (Cybenko 1989, Hornik, Stinchcombe, and White 1989), many issues are still unsolved, such as the network topology, the initial weights setting, the generalization ability and so forth. Different contributions from statisticians and neural networks researchers attack these problems from various aspects (Battiti 1992, Parisi et al. 1996, Wang and Chen 1996, Yam and Chow 2000). As MLP training is quite data dependent, naturally, pre-processing techniques have been proposed for efficient training. LeCun (LeCun 1998) suggests de-correlating the input features before training. This strategy will make the system equations in the training algorithm easier to solve. For linearly dependent inputs, it can decrease the network size by removing redundant features. Raudys (Raudys 2001) points out that it is difficult to ensure fast convergence of the back-propagation algorithm when we have an immense difference between the smallest and largest eigenvalues of the data covariance matrix (CM). He recommends the use of a whitening transformation that scales the input features after de-correlation, to transform the CM into an identity matrix. This speeds up the convergence rate.

From these results, some readers may assume that fewer training iterations are required when de-correlated inputs are used. However, in our research, we found that, at least in the orthogonal transformation case, the de-correlation procedure may not result in fewer iterations. By analyzing the training procedure of the original network and the transformed network, which has de-correlated inputs, we discovered that if the two networks start from the same initial state, they will have exactly the same training dynamics and achieve the same results for a given training algorithm. The same initial state can be realized by a proper transformation between the corresponding initial weight sets of the networks. So the effect of de-correlation is equivalent to initializing the original network with a different set of weights. For a robust training algorithm, this would be of little help for improving training speed.

In the first section below, we introduce the notation used in the derivation and describe the Karhunen-Loeve transform (KLT). Next, we give theoretical analyses of the effects of de-correlation by tracking the training procedure in two networks. After that, simulation results are presented. Several practical issues are discussed in the last section.

MLP and KLT

Without loss of generality, we restrict our discussion to a three layer fully connected MLP with linear output activation functions. First, we describe the structure and notation, and then present one classical de-correlation technique, the KLT.

Notation for a Fully Connected MLP

The network structure is shown in Fig. 1. For clarity, the bypass weights from input layer to output layer are not shown. The training data set consists of N_v training patterns $\{(\mathbf{x}_p, \mathbf{t}_p)\}$, where the pth input row vector $\mathbf{x}_p$ and the pth desired output row vector $\mathbf{t}_p$ have dimensions N and M, respectively. For the jth hidden unit, the net function net_{pj} and the output activation O_{pj} for the pth training pattern are

$$net_{pj} = \sum_{n=1}^{N} w_{hi}(j,n) \cdot x_{pn} + T_h(j),$$
$$O_{pj} = f(net_{pj}) \qquad 1 \le j \le N_h \qquad (1)$$

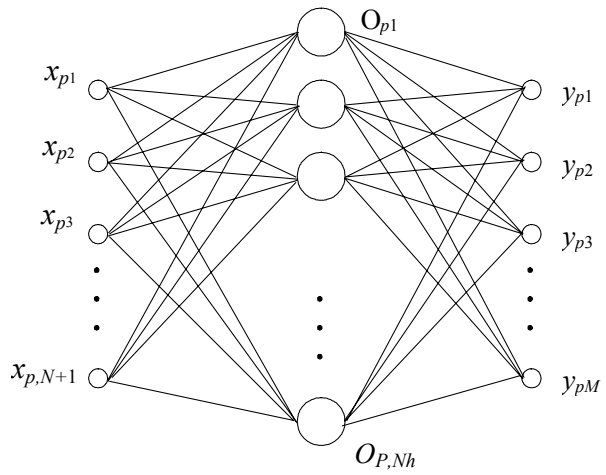

Fig. 1 MLP network structure

Here x_{pn} denotes the nth element of $\mathbf{x}_p$, $w_{hi}(j, n)$ denotes the weight connecting the nth input unit to the jth hidden unit, $T_h(j)$ is the hidden unit threshold and N_h is the number of hidden units. The activation function f is the sigmoid function

$$f\left(net_{pj}\right)=\frac{1}{1+e^{-net_{pj}}} \tag{2}$$

The kth output y_{pk} for the pth training pattern is

$$y_{pk}=\sum_{n=1}^{N} w_{oi}\left(k,n\right)\cdot x_{pn}+\sum_{j=1}^{N_h} w_{oh}\left(k,j\right)\cdot O_{pj}+T_o\left(k\right) \tag{3}$$

k=1, …, M, where $w_{oi}(k, n)$ denotes the weight connecting the nth input node to the kth output unit, $T_o(k)$ is the output threshold and $w_{oh}(k, j)$ denotes the weight connecting the jth hidden unit to the kth output unit. In batch mode training, the overall performance of a feed-forward network, can be measured by the mean square error (MSE)

$$\begin{aligned} E&=\frac{1}{N_v}\sum_{p=1}^{N_v}\sum_{m=1}^{M}\left[t_{pm}-y_{pm}\right]^2 \\ &=\frac{1}{N_v}\sum_{p=1}^{N_v}\left(\mathbf{t}_p-\mathbf{y}_p\right)\cdot\left(\mathbf{t}_p-\mathbf{y}_p\right)^T \end{aligned} \tag{4}$$

where t_{pm} denotes the mth element of the pth desired output vector. Recall that $\mathbf{t}_p$ and $\mathbf{y}_p$ are row vectors.

Karhunen-Loeve Transform (KLT)

The KL transform was originally introduced by Karhunen and Loeve. Here, the orthogonal KLT matrix $\mathbf{A}$ is used to de-correlate the original input feature vector $\mathbf{x}_p$ to $\mathbf{z}_p$:

$$\begin{bmatrix} z_{p1} \\ z_{p2} \\ \vdots \\ z_{pN} \end{bmatrix}=\mathbf{z}_P^T=\mathbf{A}\mathbf{x}_p^T=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,N} \\ a_{21} & a_{22} & \cdots & a_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N,1} & a_{N,2} & \cdots & a_{N,N} \end{bmatrix}\begin{bmatrix} x_{p1} \\ x_{p2} \\ \vdots \\ x_{pN} \end{bmatrix} \tag{5}$$

For the original training patterns, the mean vector and CM are

$$\mathbf{m}_x=\frac{1}{N_v}\sum_{p=1}^{N_v}\mathbf{x}_p \tag{6}$$

$$\mathbf{C}_x=\frac{1}{N_v}\sum_{p=1}^{N_v}\left[\left(\mathbf{x}_p-\mathbf{m}_x\right)^T\left(\mathbf{x}_p-\mathbf{m}_x\right)\right] \tag{7}$$

After applying the KLT, the mean vector and CM of the de-correlated inputs are

$$\mathbf{m}_z=\frac{1}{N_v}\sum_{p=1}^{N_v}\mathbf{z}_p=\mathbf{m}_x\cdot\mathbf{A}^T \tag{8}$$

$$\begin{aligned} \mathbf{C}_z&=\frac{1}{N_v}\sum_{p=1}^{N_v}\left[\left(\mathbf{z}_p-\mathbf{m}_z\right)^T\left(\mathbf{z}_p-\mathbf{m}_z\right)\right] \\ &=\mathbf{A}\mathbf{C}_x\mathbf{A}^T \end{aligned} \tag{9}$$

The new CM $\mathbf{C}_z$ is a diagonal matrix. The orthogonal KLT matrix $\mathbf{A}$ can be found by the singular value decomposition (SVD).

Input De-correlation and MLP Training

The effect of de-correlating the input features is analyzed in this section. We demonstrate our analyses using the full conjugate gradient algorithm (FCG) (Fletch 1987, Kim 2003).

Assume that two MLP networks are respectively trained with training data $\left\{\mathbf{x}_p,\mathbf{t}_p\right\}_{p=1}^{N_v}$ and $\left\{\mathbf{z}_p,\mathbf{t}_p\right\}_{p=1}^{N_v}$. The networks are said to be equivalent if their hidden layer activations are identical and their output vectors are identical for every training pattern.

First, the conditions for ensuring equivalent states in the two networks are derived. Then we briefly describe the FCG algorithm. Finally, by induction, we show that the training dynamics in the two networks are exactly the same. We will use $\mathbf{u}$ to denote the weights in the transformed network and add an extra subscript 'd', representing de-correlation, to other notation for discrimination.

Conditions for Equivalent States

If the weights of the transformed network are $\mathbf{u}_{oi}$, $\mathbf{u}_{oh}$, and $\mathbf{u}_{hi}$, the thresholds are $\mathbf{T}_{do}$, $\mathbf{T}_{dh}$, then the hidden layer net functions and network outputs are:

$$\mathbf{net}_{dp}^T=\mathbf{u}_{hi}\cdot\mathbf{z}_p^T+\mathbf{T}_{dh}=\mathbf{u}_{hi}\cdot\mathbf{A}\cdot\mathbf{x}_p^T+\mathbf{T}_{dh} \tag{10}$$

$$\begin{aligned} \mathbf{y}_{dp}^T&=\mathbf{u}_{oi}\cdot\mathbf{z}_p^T+\mathbf{T}_{do}+\mathbf{u}_{oh}\cdot\mathbf{O}_{dp}^T \\ &=\mathbf{u}_{oi}\cdot\mathbf{A}\cdot\mathbf{x}_p^T+\mathbf{T}_{do}+\mathbf{u}_{oh}\cdot\mathbf{O}_{dp}^T \end{aligned} \tag{11}$$

where the hidden layer output vector $\mathbf{O}_{dp}$ is calculated by applying the sigmoid function to $\mathbf{net}_{dp}$.

From equations (10) and (11), we can see that, if the weights and thresholds in the original network satisfy:

$$\mathbf{w}_{hi}=\mathbf{u}_{hi}\cdot\mathbf{A},\ \mathbf{w}_{oi}=\mathbf{u}_{oi}\cdot\mathbf{A} \tag{12}$$

$$\mathbf{w}_{oh}=\mathbf{u}_{oh},\ \mathbf{T}_o=\mathbf{T}_{do},\ \mathbf{T}_h=\mathbf{T}_{dh} \tag{13}$$

then the two networks will have the same net functions and outputs, and consequently, the same states. After

initializing the two networks with the strategy in (12) and (13), they start from the same initial state.

The Full Conjugate Gradient Algorithm

In FCG (Fletch 1987, Kim 2003), for the kth training iteration, the weights and thresholds are updated in corresponding conjugate directions, **P,** i.e., in the transformed network

$$\mathbf{u}_{oi} \leftarrow \mathbf{u}_{oi} + B_2 \cdot \mathbf{P}_{doi}(k),\ \mathbf{u}_{hi} \leftarrow \mathbf{u}_{hi} + B_2 \cdot \mathbf{P}_{dhi}(k) \quad (14)$$

$$\mathbf{u}_{oh} \leftarrow \mathbf{u}_{oh} + B_2 \cdot \mathbf{P}_{doh}(k),\ \mathbf{T}_{do} \leftarrow \mathbf{T}_{do} + B_2 \cdot \mathbf{P}_{do}(k)$$

$$\mathbf{T}_{dh} \leftarrow \mathbf{T}_{dh} + B_2 \cdot \mathbf{P}_{dh}(k) \quad (15)$$

The weights and thresholds in the original network are updated in the same way. So, if the two networks have the same learning factors B_2 and the conjugate directions **P** satisfy:

$$\mathbf{P}_{hi} = \mathbf{P}_{dhi} \cdot \mathbf{A},\ \mathbf{P}_{oi} = \mathbf{P}_{doi} \cdot \mathbf{A} \quad (16)$$

$$\mathbf{P}_{oh} = \mathbf{P}_{doh},\ \mathbf{P}_{o} = \mathbf{P}_{do},\ \mathbf{P}_{h} = \mathbf{P}_{dh} \quad (17)$$

then they will still have the same states as long as they have the same states in the (k-1)th iteration. It is easy to verify this because updating the weights and thresholds in (14) to (17) will ensure conditions (12) and (13).

In FCG, the conjugate directions evolve from iteration to iteration in the following way:

$$\mathbf{P}(k) = -\mathbf{g}(k) + B_1 \mathbf{P}(k-1) \quad (18)$$

Here **g** is the gradient matrix for the weights and thresholds. In the transformed network, for certain training iterations, the gradient matrix of $\mathbf{u}_{oi}$ has the form:

$$\mathbf{g}_{doi} = \frac{\partial E}{\partial \mathbf{u}_{oi}} = \begin{bmatrix} \frac{\partial E}{\partial u_{oi}(1,1)} & \cdots & \frac{\partial E}{\partial u_{oi}(1,N)} \\ \vdots & \vdots & \vdots \\ \frac{\partial E}{\partial u_{oi}(M,1)} & \cdots & \frac{\partial E}{\partial u_{oi}(M,N)} \end{bmatrix}$$

$$= \frac{-2}{N_v} \cdot \sum_{p=1}^{N_v} \begin{bmatrix} e_{dp1} \cdot z_{p1} & \cdots & e_{dp1} \cdot z_{pN} \\ \vdots & \vdots & \vdots \\ e_{dpM} \cdot z_{p1} & \cdots & e_{dpM} \cdot z_{pN} \end{bmatrix} \quad (19)$$

$$= \frac{-2}{N_v} \sum_{p=1}^{N_v} \begin{bmatrix} e_{dp1} \\ \vdots \\ e_{dpM} \end{bmatrix} \cdot \mathbf{z}_p = \frac{-2}{N_v} \left(\sum_{p=1}^{N_v} \mathbf{e}_{dp} \cdot \mathbf{x}_p \right) \cdot \mathbf{A}^T$$

where

$$e_{dpm} = t_{pm} - y_{dpm} \qquad m = 1, \cdots M \quad (20)$$

Similarly, the gradient matrix of $\mathbf{u}_{oh}$ is:

$$\mathbf{g}_{doh} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{e}_{dp} \cdot \mathbf{O}_{dp} \quad (21)$$

The gradient matrix of $\mathbf{T}_{do}$ is:

$$\mathbf{g}_{do} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{e}_{dp} \quad (22)$$

For the hidden weights $\mathbf{u}_{hi}$, the gradient matrix is:

$$\mathbf{g}_{dhi} = \frac{\partial E}{\partial \mathbf{u}_{hi}} = \begin{bmatrix} \frac{\partial E}{\partial u_{hi}(1,1)} & \cdots & \frac{\partial E}{\partial u_{hi}(1,N)} \\ \vdots & \vdots & \vdots \\ \frac{\partial E}{\partial u_{hi}(N_h,1)} & \cdots & \frac{\partial E}{\partial u_{hi}(N_h,N)} \end{bmatrix}$$

$$= \frac{-2}{N_v} \sum_{p=1}^{N_v} \begin{bmatrix} \left(\sum_{m=1}^{M} e_{dpm} \frac{\partial y_{dpm}}{\partial O_{dp1}} \right) z_{p1} & \cdots & \left(\sum_{m=1}^{M} e_{dpm} \frac{\partial y_{dpm}}{\partial O_{dp1}} \right) z_{pN} \\ \vdots & \vdots & \vdots \\ \left(\sum_{m=1}^{M} e_{dpm} \frac{\partial y_{dpm}}{\partial O_{dpN_h}} \right) z_{p1} & \cdots & \left(\sum_{m=1}^{M} e_{dpm} \frac{\partial y_{dpm}}{\partial O_{dpN_h}} \right) z_{pN} \end{bmatrix}$$

$$= \frac{-2}{N_v} \sum_{p=1}^{N_v} \begin{bmatrix} \left(\sum_{m=1}^{M} e_{dpm} u_{oh}(m,1) \right) \\ \vdots \\ \left(\sum_{m=1}^{M} e_{dpm} u_{oh}(m,N_h) \right) \end{bmatrix} \cdot \mathbf{z}_p = \frac{-2}{N_v} \left(\sum_{p=1}^{N_v} \mathbf{D}_{dp} \mathbf{x}_p \right) \cdot \mathbf{A}^T \quad (23)$$

Similarly, the gradient matrix for the hidden unit threshold vector $\mathbf{T}_{dh}$:

$$\mathbf{g}_{dh} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{D}_{dp} \quad (24)$$

Applying similar analyses to the original network, we can write the corresponding gradient matrices:

$$\mathbf{g}_{oi} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{e}_p \cdot \mathbf{x}_p,\ \mathbf{g}_{hi} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{D}_p \cdot \mathbf{x}_p \quad (25)$$

$$\mathbf{g}_{oh} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{e}_p \cdot \mathbf{O}_p,\ \mathbf{g}_{o} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{e}_p,\ \mathbf{g}_{h} = \frac{-2}{N_v} \sum_{p=1}^{N_v} \mathbf{D}_p \quad (26)$$

B_1 in (18) is the ratio of the gradient energies between the current iteration and the previous iteration:

$$B_1 = \frac{\|\mathbf{g}(i)\|}{\|\mathbf{g}(i-1)\|} \quad (27)$$

Dynamics of the Training Procedure

In the following, we show by induction that the above two networks have the same dynamics for full conjugate gradient training.

In the first training iteration, for both networks,

$$\mathbf{g}_d(0) = \mathbf{g}(0) = \mathbf{0},\ \mathbf{P}_d(0) = \mathbf{P}(0) = \mathbf{0} \quad (28)$$

In order to calculate B_1, we just set

$$\|\mathbf{g}_d(0)\| = \|\mathbf{g}(0)\| = 1 \quad (29)$$

Due to the initialization strategy, at the beginning of the first iteration, we have

$$\mathbf{y}_{dp} = \mathbf{y}_p,\ \mathbf{e}_{dp} = \mathbf{e}_p,\ \mathbf{O}_{dp} = \mathbf{O}_p,\ \mathbf{D}_{dp} = \mathbf{D}_p \quad (30)$$

By observing the gradient matrices (19) to (26), it is obvious that during the first training iteration, the gradients in the two networks satisfy

$$\mathbf{g}_{doi} = \mathbf{g}_{oi} \cdot \mathbf{A}^T,\ \mathbf{g}_{dhi} = \mathbf{g}_{hi} \cdot \mathbf{A}^T \quad (31)$$

$$\mathbf{g}_{doh} = \mathbf{g}_{oh},\ \mathbf{g}_{do} = \mathbf{g}_{o},\ \mathbf{g}_{dh} = \mathbf{g}_{h} \tag{32}$$

Then the energy of gradients in the transformed network for the first training iteration is:

$$\begin{aligned}\left\|\mathbf{g}_d(1)\right\| &= \sum_{j=1}^{M}\mathbf{g}_{doi}^{(j)}\cdot\left(\mathbf{g}_{doi}^{(j)}\right)^T + \sum_{j=1}^{N_h}\mathbf{g}_{dhi}^{(j)}\cdot\left(\mathbf{g}_{dhi}^{(j)}\right)^T \\ &+ \sum_{j=1}^{M}\mathbf{g}_{doh}^{(j)}\cdot\left(\mathbf{g}_{doh}^{(j)}\right)^T + \left(\mathbf{g}_{do}\right)^T\cdot\mathbf{g}_{do} + \left(\mathbf{g}_{dh}\right)^T\cdot\mathbf{g}_{dh} \\ &= \sum_{j=1}^{M}\mathbf{g}_{oi}^{(j)}\mathbf{A}^T\cdot\left(\mathbf{g}_{oi}^{(j)}\mathbf{A}^T\right)^T + \sum_{j=1}^{N_h}\mathbf{g}_{hi}^{(j)}\mathbf{A}^T\cdot\left(\mathbf{g}_{hi}^{(j)}\mathbf{A}^T\right)^T \\ &+ \sum_{j=1}^{M}\mathbf{g}_{oh}^{(j)}\cdot\left(\mathbf{g}_{oh}^{(j)}\right)^T + \left(\mathbf{g}_{o}\right)^T\cdot\mathbf{g}_{o} + \left(\mathbf{g}_{h}\right)^T\cdot\mathbf{g}_{h} \\ &= \left\|\mathbf{g}(1)\right\|\end{aligned} \tag{33}$$

where superscript (j) denote the jth row vector in the corresponding matrix. From equation (27), (29) and (33), we conclude that B_1 is the same in the two networks. Based on equations (28), (31), (32), and substituting them into (18), the new conjugate directions in the first training iteration satisfy conditions (16) and (17).

In order to prove that the two networks have same state after the first training iteration, we still need to verify that the learning factor B_2 is the same in the two networks. The learning factor B_2 is found by solving:

$$\frac{\partial E}{\partial B_2} = 0 \tag{34}$$

In the transformed network, the output vector after a training iteration is:

$$\begin{aligned}\mathbf{y}_{dp}^T &= \left(\mathbf{u}_{oi} + B_2\mathbf{P}_{doi}\right)\cdot\mathbf{z}_p^T + \left(\mathbf{T}_{do} + B_2\mathbf{P}_{do}\right)\mathbf{T}_{do} \\ &+ \left(\mathbf{u}_{oh} + B_2\mathbf{P}_{doh}\right)\cdot\mathbf{f}\left(\left(\mathbf{u}_{hi} + B_2\mathbf{P}_{dhi}\right)\cdot\mathbf{z}_p^T + \left(\mathbf{T}_{dh} + B_2\mathbf{P}_{dh}\right)\right)\end{aligned} \tag{35}$$

where **f** denotes the vector form of the sigmoid function. The current weights and thresholds satisfy conditions (12), (13), and the updating directions satisfy (16) and (17), thus equation (35) can be expressed as:

$$\begin{aligned}\mathbf{y}_{dp}^T &= \left(\mathbf{w}_{oi} + B_2\mathbf{P}_{oi}\right)\mathbf{A}^T\cdot\mathbf{A}\mathbf{x}_p^T + \left(\mathbf{T}_o + B_2\mathbf{P}_o\right) \\ &+ \left(\mathbf{w}_{oh} + B_2\mathbf{P}_{oh}\right)\mathbf{f}\left(\left(\mathbf{w}_{hi} + B_2\mathbf{P}_{hi}\right)\mathbf{A}^T\mathbf{A}\mathbf{x}_p^T + \left(\mathbf{T}_h + B_2\mathbf{P}_h\right)\right) \\ &= \mathbf{y}_p^T\end{aligned} \tag{36}$$

Hence after the first training iteration, the outputs in the two networks are exactly the same. Substituting them into (4) will result in the same expression for E in the two networks. Thus any practical methods for solving (34) will yield the same B_2 for the two networks.

From the above analyses, it is clear that the two networks will have exactly the same state after the first training iteration.

We assume for the nth iteration, that the two networks still reach the same state after training. This implies that in the (n+1)th iteration, equation (30) is satisfied, current weights and thresholds satisfy conditions (12) and (13), so (33) becomes

$$\left\|\mathbf{g}_d(n)\right\| = \left\|\mathbf{g}(n)\right\| \tag{37}$$

The derivation for the (n+1)th iteration is similar to the analyses in the first iteration. From (30), conditions (31) and (32) are met again. Just as in (33), we have

$$\left\|\mathbf{g}_d(n+1)\right\| = \left\|\mathbf{g}(n+1)\right\| \tag{38}$$

So once again B_1 is the same in the two networks. Using this result, combined with equations (18), (31) and (32), it is easy to show that conditions (16) and (17) are valid in the (n+1)th iteration. As equation (35) is true for an arbitrary iteration, from conditions (16), (17), (12) and (13), equation (36) is satisfied in the (n+1)th iteration. Based on exactly the same reason as in the first iteration, B_2 will be same in the two networks. As a result, conditions (12) and (13) are proved to be true for the new updated weights and thresholds after the (n+1)th training iteration. That is to say, the two networks will still have same state after the (n+1)th training iteration.

From the above induction, the training dynamics of the two networks are proved to be exactly the same. Actually, for other gradient-related algorithms, i.e. output weight optimization-backpropagation (OWO-BP), output weight optimization-hidden weight optimization (OWO-HWO), we can prove similar conclusions as long as learning factors are optimal, meaning that equation (34) is satisfied. Without the sufficient condition of equation (34), the training errors of two equivalent networks may diverge during training.

Simulation

The results in the last section are verified using four training data sets. Our simulations were carried out on a 733Mhz Pentium III, running Windows 2000 and using the Visual C++ 6.0 compiler.

Training data set *Twod.tra* contains simulated data based on models from backscattering measurements. This training file is used in the task of inverting the surface scattering parameters from an inhomogeneous layer above a homogeneous half space, where both interfaces are randomly rough. The parameters to be inverted are the effective permittivity of the surface, the normalized *rms* height, the normalized surface correlation length, the optical depth, and single scattering albedo of an inhomogeneous irregular layer above a homogeneous half space from back scattering measurements.

The data set has 8 inputs, 7 outputs, and 1768 patterns. We use 10 hidden units in a three-layer MLP. The networks are trained using FCG for 200 iterations. From the simulation results of figure 2, it is clear that the two networks have exactly the same dynamics.

Training data set *mattrn.dat* provides the data for inversion of random two-by-two matrices. Each of the 2000 patterns consists of 4 input features and 4 output features. The input features, which are uniformly distributed between 0 and 1, represent matrix elements and the four

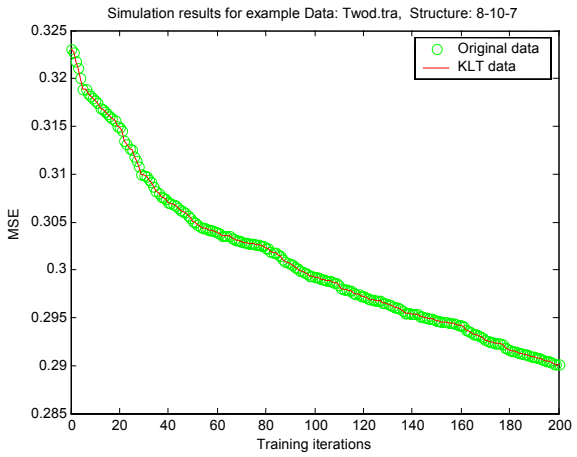

Fig. 2 Simulation results for example 1 data: *Twod.tra*, Structure: 8-10-7

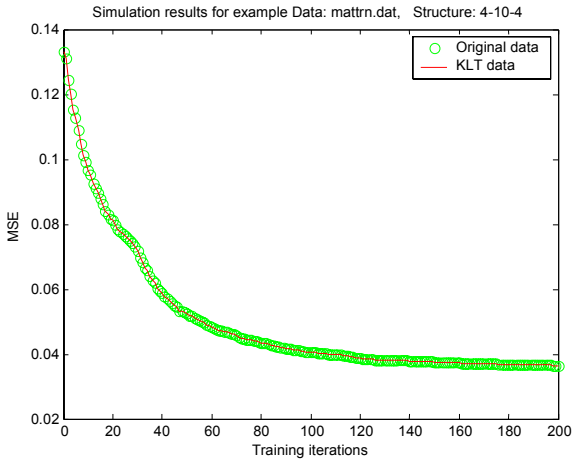

Fig. 3 Simulation results for example 2 data: *mattrn.dat*, Structure: 4-10-4

output features are elements of the corresponding inverse matrix. The determinants of the input matrices are constrained to be between 0.3 and 2.

We chose the MLP structure 4-10-4 and trained the networks for 200 iterations. The simulation results of figure 3 show the same dynamics for the two networks.

Training data *fmtrain.dat* is used to train a neural network to perform demodulation of an FM (frequency modulation) signal containing a sinusoidal message. It has 5 inputs, 1 output, and 1024 patterns. The data are generated from the equation

$$r(n) = C_{amp} \bullet \cos[2\pi \bullet n \bullet C_{freq} + M_{amp} \bullet \sin(2\pi \bullet n \bullet M_{freq})]$$

where C_{amp} is the Carrier Amplitude, M_{amp} is the Message Amplitude, C_{freq} is the normalized Carrier frequency, M_{freq} is the normalized message frequency. In this data set, C_{amp} = .5, C_{freq} = .1012878, M_{freq} = .01106328, and M_{amp}=5. The five inputs are r(n-2), r(n-1), r(n), r(n+1), and r(n+2). The desired output is $\cos(2\pi \bullet n \bullet M_{freq})$. In each consecutive pattern, n is incremented by 1.

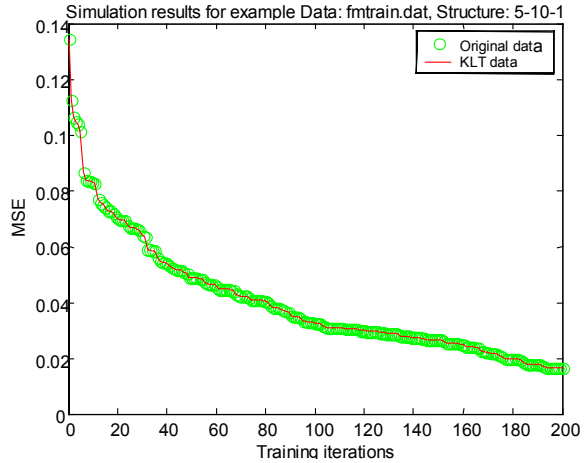

Fig. 4 Simulation results for example 3 data: *fmtrain.dat*, Structure: 5-10-1

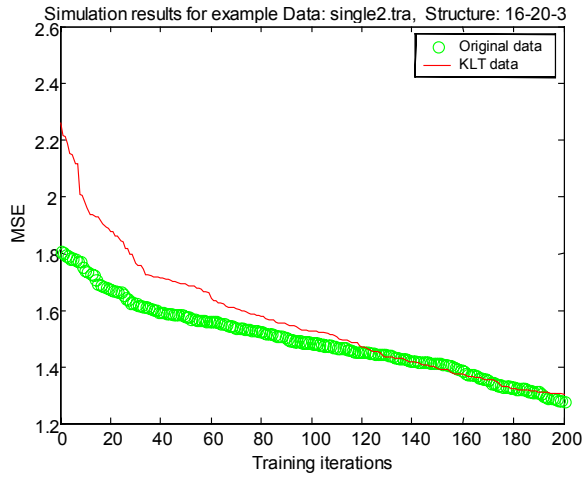

Fig. 5 Simulation results for example 4 data:*Single2.tra*, Structure: 16-20-3

We use 10 hidden units in the three-layer MLP. The networks are trained using FCG for 200 iterations. Figure 4 shows the result.

Single2.tra consists of 16 inputs, 3 outputs and 5992 patterns. It represents the training set for inversion of surface permittivity, the normalized surface *rms* roughness, and the surface correlation length found in the back scattering models from randomly rough dielectric surfaces.

The first 16 inputs represent the simulated back scattering coefficient measured at 10, 30, 50 and 70 degrees at both vertical and horizontal polarization. The remaining 8 are various combinations of ratios of the original eight values. These ratios correspond to those used in several empirical retrieval algorithms (Fung, Li and Chen 1992).

We chose the MLP structure 16-20-3 and trained the networks for 200 iterations. From the simulation results of figure 5, we can see that there are obvious differences between the two curves. This phenomenon is caused by

problems in the matrix **A**. For data sets *Twod.tra*, *mattrn.dat*, and *fmtrain.dat*, we found that **A** calculated by the SVD is orthogonal, i.e. $\mathbf{A}^T\mathbf{A} = \mathrm{I}$, so the analyses in the last section are satisfied. However, for the training data file *Single2.tra*, **A** is not orthogonal because of problems in the SVD. Our analyses are not valid for this case and the two networks are not equivalent, as seen in figure 5.

Conclusion

In this paper, we analyzed the effect of orthogonally transformed input vectors on MLP training. Using the concept of equivalent states, we have shown that this de-correlation procedure has no effect on the error per iteration for the FCG training algorithm. Of course, de-correlation is still useful for compression purposes and as a means for reducing the number of operations per iteration in some training algorithms. We have derived our results for networks that have bypass weights. These results are still valid when the bypass weights are set to zero, or equivalently, removed.

Although the analyses in this paper are based upon FCG, they apply equally well to BP, and other gradient-based training algorithms when optimal learning factors are used.

Acknowledgement

This work was supported by the Advanced Technology Program of the state of Texas, under grant number 003656-0129-2001.

References

Battiti, R. 1992. First- and Second-order Methods for Learning: between Steepest Descent and Newton's Method. *Neural Computation* 4 (2): 141-166.

Chen, H. H.; Manry, M. T.; and Chandrasekaran, H. 1999. A Neural Network Training Algorithm Utilizing Multiple Sets of Linear Equations. *Neurocomputing* 25: 55-72.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems* 2: 303-314.

Fletch, R. 1987. *Practical Methods of Optimization*. John Wiley & Sons.

Fung, A.K.; Li, Z.; and Chen, K.S. 1992. Back Scattering from a Randomly Rough Dielectric Surface. *IEEE Transactions on Geoscience and Remote Sensing* 30 (2): 356-369.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2: 359-366.

Kim, T. H.; Manry, M. T.; and Maldonado, F. J. 2003. New Learning Factor and Testing Methods for Conjugate Gradient Training Algorithm. In *Proceedings of 2003 International Joint Conference on Neural Networks*, 2011-2016. Portland, OR: International Joint Conference on Neural Networks.

LeCun Y., et al. 1998. Efficient BackProp. In Orr, G. B. and Muller, K. R. eds., *Neural Networks*: *Tricks of the Trade*. Springer-Verlag.

Manry, M. T., et al. 1994. Fast Training of Neural Networks for Remote Sensing. *Remote Sensing Reviews* 9: 77-96.

Manry, M. T.; Chandrasekaran, H.; and Hsieh, C. H. 2001. *Signal Processing Using the Multiplayer Perceptron*. CRC Press.

Parisi, R.; Claudio, E. D.; Orlandi G.; and Rao B. D. 1996. A Generalized Learning Paradigm Exploiting the Structure of Feedforward Neural Networks. *IEEE Transactions on Neural Networks* 7 (6): 1450-1460.

Raudys, Š. 2001. *Statistical and Neural Classifiers: An Integrated Approach to Design*. Springer-Verlag.

Wang, G. J., and Chen C. C. 1996. A Fast Multilayer Neural-network Training Algorithm Based on the Layer-by-layer Optimizing Procedures. *IEEE Transactions on Neural Networks* 7 (3): 768-775.

Yam, Y. F., and Chow, W. S. 2000. A Weight Initialization Method for Improving Training Speed in Feedforward Neural Network. *Neurocomputing* 30: 219-232.