Multiple Training Concept for Back-Propagation Neural Networks for Use in Associative Memories

YEOU-FANG WANG, JOSE B. CRUZ, JR., AND J. H. MULLIGAN, JR.

University of California at Irvine
(Received 10 June 1991; revised and accepted 10 March 1992)

Abstract—The multiple training concept first applied to Bidirectional Associative Memory training is applied to the back-propagation (BP) algorithm for use in associative memories. This new algorithm, which assigns different weights to the various pairs in the energy function, is called multiple training back-propagation (MTBP). The pair weights are updated during the training phase using the basic differential multiplier method (BDMM). A sufficient condition for convergence of the training phase is that the second derivative of the energy function with respect to the weights of the synapses is positive along the paths of both synapse weights and pair weights. A simple example of the use of the algorithm is provided, followed by two simulations that show that this algorithm can increase the training speed of the network dramatically.

Keywords—Back-propagation, Multiple training, Bidirectional associative memory, Gradient descent, Basic differential multiplier method, Least-squares optimization, Weighted energy function, Training speed.

1. INTRODUCTION

Artificial neural networks have been used as associators, classifiers, and optimizers in many fields. Associators implement mappings from one field to another for the training pairs. Usually, the neural network is expected to remember all training pairs; namely, the mapping scheme has to perform mappings for all associations. This kind of application is called associative memories. In this application, usually all training associations are exact; all training pairs are desired to be learned. The presence of erroneous data in the set of training pairs will confuse the network because the associator will be expected to learn the erroneous data too. Unlike associators, classifiers usually classify the training data into several classes. Training in this application could be done in either supervised or unsupervised fashion. In supervised training, the input-output relationship has to be specified. Because the network is used for classification, noisy training data are supposed to be modified and learned through the training stage. Therefore, erroneous or overlapping data may be used for training in classifier applications. We may say that associative memories are used to learn the mapping for all data, a property not required for classifiers. An optimizer is usually the neural network itself in which the energy function represents the function to be optimized and the states of the neurons or the final synapse weights represent the solution after "learning."

In this paper, we only consider associative memory type of applications. Back-propagation (BP) neural networks (Rumelhart & McClelland, 1987; Werbos, 1974), that use superpositions of sigmoidal functions to approximate a desired function (Cybenko, 1989) are those under consideration. The training of this class of neural network is based on solving least-square optimization problems with gradient descent. The use of energy functions for optimization purposes is a common technique in evaluation of associative memories, and its extension to the analysis of back-propagation networks provides a powerful analytical method for our investigation. Because square terms of the error for all training data are used in the energy function (performance index), the energy function is well suited for associative memories. In this paper, we will modify the training scheme of the network to improve the training speed for associative memory types of application.

For associative memories, the BP network is used...
to learn \( N \) input–output pairs \( \{(A_p, B_p) | p = 1, \ldots, N\} \), where \( A_p \) is a vector in the input field of the network and \( B_p = [t_{p1}, \ldots, t_{pm}] \) is a vector in the output field of the network, such that if \( A_p \) appears in the input field, \( B_p \) is desired to be in the output field. \( O_{pj} = [o_{p1}, \ldots, o_{pm}] \) is defined to be the generated output that is the set of outputs of the neurons in the output (final) layer when \( A_p \) appears in the input field. The processing function \( f \) of each neuron is differentiable and non-decreasing. The output of neuron \( j \) (except those in the input layer) is

\[
o_{pj} = f(\text{net}_{pj}),
\]

where

\[
\text{net}_{pj} = \sum_i w_{ji} o_{pi}
\]

and \( w_{ji} \) is the weight of the synapse between neuron \( j \) and \( i \), neuron \( i \) is in the layer one layer ahead of the layer where neuron \( j \) is situated. The processing function of the input layer is linear with unity gain. Therefore, the input layer may not be treated as a layer in the sense normally used (Kung & Hwang, 1988).

Basically, we wish to minimize the square of the difference between the desired output (target), \( B_p \), and the actual generalized output in the final layer, \( O_{pj} \). Define the energy function for pair \( p(E_p) \) to be

\[
E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2
\]

where \( t_{pj} \) is the desired output of neuron \( j \) for pair \( p \) and \( o_{pj} \) is the generated output of neuron \( j \) for pair \( p \) in the output layer.

If we do minimization on eqn (3), we can only make \( O_{pj} \) closer to \( B_p \). However, changing the weights for the next pair may affect what has been achieved by the previous pairs.

The energy function taking into account all pairs is designated \( E \) and is defined as

\[
E = \frac{1}{2} \sum_p \sum_j (t_{pj} - o_{pj})^2 = \sum_p E_p.
\]

Because we are interested in the contributions of all pairs, minimizing \( E \) is necessary. By using a gradient descent with respect to weight \( w_{ji} \), we have

\[
\delta w_{ji} = -\frac{\delta E}{\delta w_{ji}},
\]

It can be proved that

\[
\frac{\delta E}{\delta w_{ji}} = -\sum_r \delta_{pr} o_{pi},
\]

where

\[
\delta_{pr} = (t_{pj} - o_{pj})f'_j(\text{net}_{pj}),
\]

for the output (final) layer, and

\[
\delta_{pj} = f'_j(\text{net}_{pj}) \sum_k \delta_{pk} w_{ki},
\]

for middle layers, where \( k \) indicates the neurons in the layer next to the current layer and \( f'_j(\text{net}_{pj}) = df_j/d\text{net}_{pj} \). For a computer search based on the gradient descent method, we use

\[
\Delta w_{ji} = \eta \sum_p \delta_{pr} o_{pi},
\]

where \( \eta \) is a constant. One of the disadvantages of this learning algorithm is that the learning speed is too slow. Some improvement ideas have been introduced in the literature, for instance, changing the processing function of the neurons, changing the input gain to the neuron (Tawel, 1989; Kruschke & Rodriguez–Movellan, 1991), selective updates (Huang & Huang, 1990), network structure (numbers of layers and neurons or output feedback) (Mirchandani & Cao, 1989), selective training step size (Jacobs, 1988), putting momentum terms (bias) (Rumelhart & McClelland, 1987; Hanson and Pratt, 1989), etc. In the next section, a concept called multiple training will be applied to the BP algorithm. The resulting algorithm is different from the above methods.

2. MULTIPLE TRAINING FOR BACK-PROPAGATION NEURAL NETWORK

A multiple training concept was introduced by Wang, Cruz, and Mulligan (1989, 1990a, b, c). This concept was used to improve the capacity of a bidirectional associative memory (BAM) (Kosko, 1987a, b, 1988). This idea can be extended to the back-propagation neural network when used as an associative memory. Let \( \lambda_p \) be the pair weight for pair \( p \). Then the evaluation (energy) function becomes

\[
E = \sum_p \lambda_p E_p(W),
\]

where \( W \) is the synapse weight vector with weights \( w_{pq} \)'s in the entries. Basically, multiple training is an idea to use different weights on different pairs. If the \( \lambda_p \)'s are integers, then \( \lambda_p \) represents the number of times pair \( p \) is used for each training cycle. More generality, we will allow \( \lambda_p \)'s to be real numbers. This equation is similar to the weighted least squares (Franklin, Powell, & Workman, 1990) in system identification problems or the moving least squares (Lancaster & Šalkauskas, 1986) in curve fitting problems. However, we are dealing with squares of nonlinear functions in eqn (10). Notice that the energy function keeps changing during the training phase due to changes in \( \lambda_p \). Actually, we do not care about the shape of the energy surface as long as we can find the \( W \) coordinates where \( E_p(W) = \)}
0 for all \( p \). The goal (finding \( W \) coordinates for the minimum of \( E \)) is the same as that of the regular BP except that the energy surface of BP is fixed. Since \( w_q \)'s and \( \lambda_p \)'s are the variables, if we apply the gradient descent directly to minimize \( E \), then

\[
\begin{align*}
\dot{w}_q &= -\frac{\partial E}{\partial w_q}, \quad \text{and} \quad \dot{\lambda}_p &= -\frac{\partial E}{\partial \lambda_p} = -E_p. \\
\end{align*}
\]

However, if eqn (11) is used, \( \lambda_p \) will become smaller and smaller since \( \lambda_p < 0 \), so that \( E_p \) is no longer critical to \( E \) eventually. Namely, it is not necessary that the generated output \( O_{Wt} \) be close to the target output \( B_p \) in order to get small \( E \). Therefore, eqn (11) is not suitable.

Platt and Barr (1987) introduced a method called the basic differential multiplier method (BDMM) for constrained optimization. The basic idea of BDMM is the following:

Suppose we want to

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad g(x) = 0.
\end{align*}
\]

Construct

\[
T(x) = f(x) + \lambda g(x).
\]

then use the gradient descent

\[
\dot{x} = -\frac{\partial T}{\partial x}
\]

and the gradient ascent

\[
\dot{\lambda} = \frac{\partial T}{\partial \lambda} = g(x)
\]

to find the optimal solution.

The BDMM method is applied to the multiple training back-propagation (MTBP) network problem. In eqn (10),

\[
E = \sum_p \lambda_p E_p(W),
\]

similar to eqn (13) as \( f(x) = 0 \). Namely, the feasible solution \( g(x) = 0 \) is the only concern. That is, we want \( E_p = 0 \) if possible. Notice that

\[
E_p = \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2.
\]

If \( E_p \) is close to 0, then \( O_{Wt} \) is close to \( B_p \).

Applying BDMM, the MTBP training phase becomes

\[
\begin{align*}
\dot{w}_q &= -\frac{\partial E}{\partial w_q} = -\sum_p \lambda_p \frac{\partial E_p}{\partial w_q} \\
\dot{\lambda}_p &= \frac{\partial E}{\partial \lambda_p} = E_p.
\end{align*}
\]

Equation (17) makes sense, because we want to emphasize the pair that has larger error \( E_p \) by putting a larger weight \( \lambda_p \) on it. From eqn (17), if \( E_p \) is large, namely, pair \( p \) is not well learned, then \( \lambda_p \) is modified to be relatively larger in the positive direction such that \( w_q \) can be modified to decrease \( E_p \) more in eqn (16). Generally speaking, because the gradient descent is applied to \( W \) to decrease \( E \) and \( \lambda_p \) is an increasing positive number, \( E_p \) is most likely to drop as time increases.

Intuitively, if there exists a \( \bar{W} \) such that \( E_p = 0 \) for all \( p \), then in the neighborhood of \( \bar{W} \), \( w_q \approx 0 \) and \( \lambda_p \approx 0 \). Furthermore, the process yields a global minimum. It can be proved that one sufficient condition for convergence is that the second derivative of \( E \) with respect to \( W \), \( H_{MT} \), is positive definite along the \( W \) and \( \Lambda = \{ \lambda_p \} \) paths.

This is seen from the following. From eqn (16),

\[
\begin{align*}
\dot{w}_q &= -\sum_p \left( \lambda_p \frac{\partial E_p}{\partial w_q} + \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \right) \\
&= -\sum_p \left( \frac{\partial E_p}{\partial w_q} + \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \right) \\
&= -\sum_p \frac{\partial E_p}{\partial w_q} - \sum_p \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \dot{\lambda}_p. \tag{18}
\end{align*}
\]

Intuitively, if there exists a \( \bar{W} \) such that \( E_p = 0 \) for all \( p \), then in the neighborhood of \( \bar{W} \), \( w_q \approx 0 \) and \( \lambda_p \approx 0 \). Furthermore, the process yields a global minimum. It can be proved that one sufficient condition for convergence is that the second derivative of \( E \) with respect to \( W \), \( H_{MT} \), is positive definite along the \( W \) and \( \Lambda = \{ \lambda_p \} \) paths.

This is seen from the following. From eqn (16),

\[
\begin{align*}
\dot{w}_q &= -\sum_p \left( \lambda_p \frac{\partial E_p}{\partial w_q} + \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \right) \\
&= -\sum_p \left( \frac{\partial E_p}{\partial w_q} + \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \right) \\
&= -\sum_p \frac{\partial E_p}{\partial w_q} - \sum_p \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \dot{\lambda}_p. \tag{18}
\end{align*}
\]

From eqn (18),

\[
\dot{w}_q + \sum_p \lambda_p \frac{\partial E_p}{\partial w_q} = -\sum_p \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \dot{\lambda}_p.
\]

Therefore,

\[
\frac{\partial E_{MT}}{\partial t} = \sum_q \sum_r \dot{w}_q \left( -\sum_p \lambda_p \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} \right) \dot{\lambda}_p.
\]

\[
= -\dot{\lambda} \frac{\partial^2 E_p}{\partial w_q \partial \lambda_p} W
\]

\[
= \begin{bmatrix}
\sum_q \lambda_p \frac{\partial^2 E_p}{\partial w_q 1} & \ldots & \sum_q \lambda_p \frac{\partial^2 E_p}{\partial w_q m} \\
\vdots & \ddots & \vdots \\
\sum_q \lambda_p \frac{\partial^2 E_p}{\partial w_m 1} & \ldots & \sum_q \lambda_p \frac{\partial^2 E_p}{\partial w_m m}
\end{bmatrix} W
\]
\[ \begin{align*}
&= -\hat{W}^T \sum_p \lambda_p \begin{bmatrix}
\frac{\partial^2 E_p}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E_p}{\partial w_1 \partial w_m} \\
\vdots & & \vdots \\
\frac{\partial^2 E_p}{\partial w_m \partial w_1} & \cdots & \frac{\partial^2 E_p}{\partial w_m \partial w_m}
\end{bmatrix} \hat{W} \\
&= -\hat{W}^T \sum_p \lambda_p H_p \hat{W} \\
&= -\hat{W}^T H_{MT} \hat{W}.
\end{align*} \tag{21} \]

If \( \hat{W}^T H_{MT} \hat{W}^T > 0 \), \( E_{MT} \) drops as time increases. One sufficient condition is that the second derivative \( H_{MT} \) is positive definite along the \( W \) and \( \Lambda = \{ \lambda_p \} \) paths. As some authors have noticed (Brady, RagHAVAN, & Slawny, 1989), the second derivative is preferred to be positive definite in BP net. However, the second derivative of pattern \( p \), \( H_p \), is not positive definite in general. This can be verified by checking the determinant of the second principal minor
\[ \Delta_2 = \begin{bmatrix}
\frac{\partial^2 E_p}{\partial w_1 \partial w_1} & \frac{\partial^2 E_p}{\partial w_1 \partial w_2} \\
\frac{\partial^2 E_p}{\partial w_2 \partial w_1} & \frac{\partial^2 E_p}{\partial w_2 \partial w_2}
\end{bmatrix} \]
(see Figure 1).
\[ \begin{align*}
\frac{\partial E_p}{\partial w_1} &= \frac{\partial}{\partial w_1} \left[ \frac{1}{2} \sum_j (t_{pj} - o_{pj})^2 \right] \\
&= -(t_{p0} - o_{p0}) \frac{\partial o_{p0}}{\partial w_1} \\
&= -(t_{p0} - o_{p0}) \frac{\partial o_{p0}}{\partial \text{net}_{p0}} \cdot o_{p1}
\end{align*} \]
\[ \frac{\partial^2 E_p}{\partial w_1 \partial w_1} = -(t_{p0} - o_{p0}) \frac{\partial o_{p0}}{\partial \text{net}_{p0}} \cdot o_{p1} \]
\[ \frac{\partial^2 E_p}{\partial w_1 \partial w_2} = \frac{\partial o_{p0}}{\partial \text{net}_{p0}} \cdot o_{p1} - (t_{p0} - o_{p0}) \left( \frac{\partial^2 o_{p0}}{\partial \text{net}_{p0} \partial w_1} \cdot o_{p1} \right) \]
\[ \frac{\partial^2 E_p}{\partial w_2 \partial w_1} = \left( \frac{\partial o_{p0}}{\partial \text{net}_{p0}} \right)^2 \cdot o_{p1} - (t_{p0} - o_{p0}) \left( \frac{\partial^2 o_{p0}}{\partial \text{net}_{p0} \partial w_1} \cdot o_{p1} \right) \]
\[ \frac{\partial^2 E_p}{\partial w_2 \partial w_2} = o_{p1}^2 \left( \frac{\partial o_{p0}}{\partial \text{net}_{p0}} \right)^2 - (t_{p0} - o_{p0}) \frac{\partial^2 o_{p0}}{\partial \text{net}_{p0}^2} \]

For convenience
\[ C_p = \left[ \begin{bmatrix}
\frac{\partial o_{p0}}{\partial \text{net}_{p0}}^2 - (t_{p0} - o_{p0}) \frac{\partial^2 o_{p0}}{\partial \text{net}_{p0}^2} 
\end{bmatrix} \right] \]

Then
\[ \frac{\partial^2 E_p}{\partial w_1 \partial w_1} = o_{p1}^2 \cdot C_p \]
\[ \frac{\partial^2 E_p}{\partial w_1 \partial w_2} = o_{p1} \cdot o_{p2} \cdot C_p \]
\[ \frac{\partial^2 E_p}{\partial w_2 \partial w_2} = o_{p2}^2 \cdot C_p \]

Therefore
\[ |\Delta_2| = C_p \begin{vmatrix}
o_{p1} & o_{p1} \cdot o_{p2} \\
o_{p2} \cdot o_{p1} & o_{p2}^2
\end{vmatrix} = 0 \]
and \( \Delta_2 \) is not positive definite.

However, this does not mean that \( H_{MT} \) cannot be positive definite. Also, even if \( H_{MT} \) is not positive definite, the process can still converge as long as \( \hat{W}^T H_{MT} \hat{W} \) is positive along \( W \) and \( \Lambda = \{ \lambda_p \} \) paths. This can be seen from the following example:

**EXAMPLE 1. Convergence of MTBP:** Suppose we want to train a pair \([1, 0], 1 \) to a two layer MTBP. The structure of the network is shown in Figure 2. The processing function is chosen to be
\[ o_{p0} = f_0(\text{net}_{p0}) = \frac{1}{1 + e^{-\text{net}_{p0}}} \] \tag{22}

Then
\[ \frac{\partial f_0}{\partial \text{net}_{p0}} = o_{p0}(1 - o_{p0}) \] \tag{23}

\[ H_{MT} = \lambda_p H_p = \lambda_p \begin{bmatrix}
\frac{\partial^2 E_p}{\partial w_1 \partial w_1} & \frac{\partial^2 E_p}{\partial w_1 \partial w_2} \\
\frac{\partial^2 E_p}{\partial w_2 \partial w_1} & \frac{\partial^2 E_p}{\partial w_2 \partial w_2}
\end{bmatrix} \]
\[ = \lambda_p C_p \begin{bmatrix}
o_{p1}^2 & o_{p1} \cdot o_{p2} \\
o_{p2} \cdot o_{p1} & o_{p2}^2
\end{bmatrix} = \lambda_p C_p \begin{bmatrix}1 & 0 \\
0 & 0
\end{bmatrix}. \]
Obviously, $H_{MT}$ is not positive definite. But, if initial $w_i > -\ln 2$,

$$
C_p = \left( \frac{\partial o_{p0}}{\partial net_{p0}} \right)^2 - (t_{p0} - o_{p0}) \left( \frac{\partial^2 o_{p0}}{\partial net_{p0}^2} \right)
$$

$$
= o_{p0} (1 - o_{p0})^2 - (1 - o_{p0}) o_{p0} (1 - o_{p0}) (1 - 2 o_{p0})
$$

$$
= o_{p0} (1 - o_{p0})^2 (3 \cdot o_{p0} - 1)
$$

$$
> o_{p0} (1 - o_{p0})^2 \left( \frac{3 \cdot 1}{1 + e^{-w_i}} - 1 \right)
$$

$$
= o_{p0} (1 - o_{p0})^2 \left( \frac{3 \cdot \frac{1}{3} - 1}{1 + e^{-w_i}} - 1 \right)
$$

so that

$$
\hat{W}^T H_{MT} \hat{W} = \lambda_p C_p \hat{w}_i^2 > 0, \text{ if } \hat{w}_i \neq 0.
$$

In order to prove that the process converges, we need to prove that $C_p$ remains positive during the period of updating $\hat{W}$. Because

$$
C_p = o_{p0} (1 - o_{p0})^2 (3 \cdot o_{p0} - 1)
$$

and $0 < o_{p0} < 1$, if we can prove $\dot{o}_{p0} > 0$, then $C_p$ remains positive. Since

$$
o_{p0} = \frac{1}{1 + e^{-w_i}},
$$

then

$$
\dot{o}_{p0} = -(1 + e^{-w_i})^{-2} e^{-w_i} \cdot (-\hat{w}_i)
$$

$$
= (1 + e^{-w_i})^{-2} e^{-w_i} \left( \frac{\partial E}{\partial w_i} \right)
$$

$$
= (1 + e^{-w_i})^{-2} e^{-w_i} \lambda_p (t_{p0} - o_{p0}) o_{p0} (1 - o_{p0}) o_{p1}
$$

$$
= (1 + e^{-w_i})^{-2} e^{-w_i} \lambda_p (1 - o_{p0})^2 o_{p0} > 0. \quad (24)
$$

Namely, even if $H_{MT}$ is not positive definite, the training still converges in this case. However, if $H_{MT}$ is positive definite, the training will be guaranteed to converge.

Now, suppose we want to train the same network with one more pair $([0, 1], 1)$. Namely, $o_{11} = 1$, $o_{12} = 0$, $o_{21} = 0$, $o_{22} = 1$, $t_{10} = t_{20} = 1$. Then,

$$
H_{MT} = \lambda_1 H_1 + \lambda_2 H_2
$$

$$
= \lambda_1 C_1 \begin{bmatrix}
\dot{u}_{11} & u_{11} \cdot u_{12} \\
\dot{u}_{12} \cdot u_{11} & \dot{u}_{12}^2
\end{bmatrix}
$$

$$
+ \lambda_2 C_2 \begin{bmatrix}
\dot{u}_{21}^2 & u_{21} \cdot u_{22} \\
\dot{u}_{22} \cdot u_{21} & \dot{u}_{22}^2
\end{bmatrix}.
$$

In order to have $H_{MT}$ positive definite, we need

$$
\lambda_1 C_1 \dot{u}_{11} + \lambda_2 C_2 \dot{u}_{11}^2 > 0
$$

(25)

and

$$
(\lambda_1 C_1 \dot{u}_{11} + \lambda_2 C_2 \dot{u}_{11}^2) (\lambda_1 C_1 \dot{u}_{11} + \lambda_2 C_2 \dot{u}_{11}^2)
$$

$$
-(\lambda_1 C_1 \dot{u}_{11} + \lambda_2 C_2 \dot{u}_{11}^2)^2 > 0
$$

or

$$
\lambda_1 \lambda_2 C_1 C_2 (\dot{u}_{11} \dot{u}_{12} - \dot{u}_{11} \dot{u}_{22})^2 > 0
$$

(26)

where

$$
C_1 = o_{10} (1 - o_{10})^2 (3 \cdot o_{10} - 1)
$$

$$
= o_{10} (1 - o_{10})^2 \left( \frac{3 \cdot \frac{1}{3} - 1}{1 + e^{-w_i}} - 1 \right)
$$

$$
C_2 = o_{20} (1 - o_{20})^2 (3 \cdot o_{20} - 1)
$$

$$
= o_{20} (1 - o_{20})^2 \left( \frac{3 \cdot \frac{1}{3} - 1}{1 + e^{-w_i}} - 1 \right).
$$

One sufficient condition is $C_1 > 0$ and $C_2 > 0$. If the initial $w_i$ and $w_2$ are both chosen to be greater than ($-\ln 2$), then the initial $C_1$ and $C_2$ are both greater than 0. To prove $\dot{o}_{10} > 0$ and $\dot{o}_{20} > 0$ such that $C_1$ and $C_2$ remain positive during training is exactly the same as eqn (24). Intuitively, both $t_{10} = 1$ and $t_{20} = 1$, because the $W$ path is chosen to make $o_{p0}$ close to $t_{p0}$. $o_{p0}$ should be increased, namely, $o_{p0} > 0$. Therefore, $H_{MT}$ is positive definite along the $W$ and $A$ path even though both $E_p$ are not positive definite.

The relation between the positive definiteness of $H_{MT}$ and the number of layers and neurons for a specific data set remains open. According to our experience, the process described about converges in most of the cases if there exists $E_p = 0$ for all $p$. However, a maximum iteration bound should be added to the algorithm, to avoid long computation runs.

The MTBP provides a scheme to adjust the step size for searching during gradient descent procedures. Because the direction of the $W$ path is to decrease $E$ while $\lambda_p$ increases, the $E_p$ is expected to decrease faster. In the computer simulations for the following examples, we found that the MTBP can increase the training speed dramatically.

### 3. COMPUTER SIMULATIONS

The multiple training back-propagation algorithm is shown in Figure 3. Based on the gradient method, we used the following equations for computer algorithm where $\eta$ is a step size

$$
\Delta w_p = \eta \sum_p \lambda_p \delta_p o_{p1}, \text{ and }
$$

$$
\Delta \lambda_p = \eta \left[ \frac{1}{2} \sum_j (t_{p0} - o_{p0})^2 \right].
$$

(27)
4. CONCLUSIONS

The multiple training concept, which assigns different weights to different pairs, is introduced to the BP network intended for use as associative memories. The pair weights are updated dynamically using gradient ascent suggested by BDMM. The multiple training only affects the learning phase, and thus the speed advantage in training should be achieved without significant effect on the operation of the BP network as an associative memory when in the decoding phase. A sufficient condition for the process of multiple training of the backpropagation network to converge is also discussed, followed by an illustrative example. Two simulations, XOR and stochastic test, showed the improvement in the learning speed of MTBP.

REFERENCES


**NOMENCLATURE**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_p$</td>
<td>Input for pair $p$</td>
</tr>
<tr>
<td>$B_p$</td>
<td>Desired output for pair $p$</td>
</tr>
<tr>
<td>$O_{p}$</td>
<td>Actual output in the final layer for pair $p$</td>
</tr>
<tr>
<td>$o_{pj}$</td>
<td>Output of neuron $j$ for pair $p$</td>
</tr>
<tr>
<td>$net_{pj}$</td>
<td>Input to neuron $j$ for pair $p$</td>
</tr>
<tr>
<td>$w_{ji}$</td>
<td>Synapse weight between neuron $j$ and $i$</td>
</tr>
<tr>
<td>$\Delta w_{ji}$</td>
<td>Weight change for $w_{ji}$</td>
</tr>
<tr>
<td>$E_p$</td>
<td>Target error for pair $p$</td>
</tr>
<tr>
<td>$E$</td>
<td>$\sum E_p$</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Step size for training</td>
</tr>
<tr>
<td>$\lambda_p$</td>
<td>Pair weight for pair $p$</td>
</tr>
</tbody>
</table>