Memory Superimposition by Backpropagation

Neural Networks

HOMMA Noriyasu, GUPTA Madan M.

journal or publication title
Bulletin of College of Medical Sciences, Tohoku University

volume
12

number
2

page range
111-120

year
2003-07-31

URL
http://hdl.handle.net/10097/33818
Memory Superimposition by Backpropagation Neural Networks

Noriyasu HOMMA\textsuperscript{1} and Madan M. GUPTA\textsuperscript{2}

\textsuperscript{1}Department of Radiological Technology College of Medical Sciences, Tohoku University
\textsuperscript{2}Intelligent Systems Research Laboratory College of Engineering, University of Saskatchewan

We propose a novel neural network for incremental learning tasks where networks are required to learn new knowledge without forgetting the old one. An essential core of the proposed neural learning structure is a transferring scheme from short-term memory (STM) into long-term memory (LTM) as in brains by using dynamic changing weights. As the number of LTMs increases, a new network structure is superimposed on the previous one without disturbing the past LTMs by introducing a lateral inhibition mechanism. Superiority of the proposed neural structure to the conventional backpropagation networks is proven with respect to the learning ability.

1. Introduction

Incremental learning methods add new knowledge to the networks without reexamining the past experiences. To achieve this attractive ability, many incremental learning schemes have been proposed as the conventional learning schemes are not incremental in nature\textsuperscript{1-3}. Conventionally, there are two typical incremental learning methods. In the first method, parameters are adapted by bounded modification and structural adaptation using the backpropagation (BP) network\textsuperscript{2,3}, while in the other one for the radial basis function networks\textsuperscript{4,5} it relearns the old memories. The restriction in the first method might enhance the local minima problem inherent in the BP learning. On the other hand, the second method is not incremental in a strict definition\textsuperscript{3}, although it can improve the generalization ability better than the former one.

A possible strict incremental learning scheme includes the following three strategies: (i) it may not change the trained weights (learning strategy), (ii) it may also provide new learnable connections to store the new knowledge (structural adaptation strategy), and (iii) to avoid disturbing the past knowledge due to the creation of new connections by the structural adaptation, a restoration strategy is also needed.
To integrate these strategies for the strict incremental learning tasks, we have proposed a novel neural model with a general formulation of dynamic and spatial changing weights (DSCWs)\(^6\). This model formulates not only its neural computing process (input–output relation), but also provides some strength to the above three strategies. Usefulness of the proposed model was demonstrated by a system identification (functional approximation) task.

In this paper, to reduce the computational complexity a simplified formulation of the DSCWs neural model is proposed for pattern classification problems. Superiority of the proposed neural learning structure to the BP networks is proven with respect to the learning ability. Also, it is demonstrated that the proposed neural model can add new input patterns to its memory space without disturbing and re-examining the past patterns.

2. Open-ended Incremental Learning

Synaptic weights are considered to store knowledge of the past experiences in biological brains\(^7\). In general, learning algorithms for neural networks provide how to change the weights. This changing rule for a weight \( w \) may be given by the following difference equation

\[
 w(k+1) = w(k) + \Delta w(k)
\]

(1)

![Diagram](image)

Figure 1. Two separate phases: learning phase and evaluation phase. \( x(k) \) and \( \hat{y}(k) \) denote the input and output vectors at the \( k \)-th iteration, respectively.
where $\Delta w(k)$ is a change of the weight at the $k$th iteration, $k=1, 2, \cdots, K$, calculated by the learning schemes for a given input vector $x(k)$ as in Fig. 1.

The actual changes in weights are carried out by an accumulation of changes for all the sample data during the learning phase, while there is no change of any weight in the evaluation phase as shown in Fig. 1. After learning is completed, the trained weights should form long-term (stable) memories which represent knowledge stored in the brain for a long time or permanently. However, the weights trained by the conventional BP algorithm might not form long-term memories (LTM) for incremental learning tasks because the memories may be disturbed by the additional learning of new data.

On the other hand, in the brain any stimuli are sustained temporary as short-term memories (STMs), and most of them are forgotten as new stimuli are inputted, but some of very impressive or rehearsed ones are stored as LTM$^7$. These biological neural learning processes seem to be carried out in an open-ended (not separated) learning phase, and thus the neural learning is naturally incremental. Therefore, a possible set of the three strategies for incremental learning tasks should provide how to transfer STM into LTM in the open-ended learning phase.

In the following we will first propose a basic concept of the learning strategy to solve this problem. The detail of the three strategies will be presented in the next section.

2.1. Short-term and Long-term Memories by Dynamic Changing Weights

As a novel learning strategy for the incremental learning tasks, a simple dynamic changing weight is defined.

$$\tau \frac{dw(t)}{dt} = -w(t) + r(t) \quad (2)$$

where $\tau (>0)$ is a time constant, and $r(t)$ is a reference or target value. If biochemical changes of synaptic weights are dynamic$^8$, then this continuous-time changing rule with a reasonable time constant is more natural than the conventional discrete-time algorithm.

In this dynamic system, it is well known
that the different time constants yield memories with different durations. That is, STM becomes LTM when a small time constant is changed into larger ones as illustrated in Fig. 2. Note that, by using different time constants, the proposed method can provide the transferring scheme from STM into LTM\(^1\) which cannot be realized by conventional learning schemes including other incremental learning schemes.

2.2. Problem Definition

A training set of input vectors \(\{x_i \in \mathbb{R}^n, i = 1, 2, \ldots, N\}\) and the corresponding desired output signals \(\{y_i \in \mathbb{R}^m, i = 1, 2, \ldots, N\}\) is considered. \(N\) is the number of the training vectors, \(n\) the number of inputs, and \(m\) the number of outputs. Let \(t_k\) denote continuous time in the \(k\)-th iteration, \(kT \leq t_k < (k+1)T, \; k=0, 1, \ldots, \) Here, \(T\) is a sampling period. During the \(k\)-th iteration, the networks are required to learn a given pair of the inputs and the desired signals, \((x(t_k), y(t_k)) = (x_i, y_i), \; i \in \{1, 2, \ldots, N\}\).

For an input vector \(x(t_k) = x_i\) at the \(k\)-th iteration, a square error measure is defined as

\[
E_i(k) = \lim_{t_k \to (k+1)T} \| \hat{y}(t_k) - y_i \|
\]

where \(\hat{y} \in \mathbb{R}^m\) is the network output vector. Note that the network outputs may be changed by the dynamic weights even if the input vector is unchanged for the iteration.

Here, when the input \(x(t_k) = x_i\) is learned successfully within an error tolerance \(\gamma > 0\), i.e., \(E_i(k) \leq \gamma\), this knowledge is supposed to be stored stably as LTM in the open-ended learning phase where additional learning continues to be carried out as in the learning phase in Fig. 1. Thus the network is required to minimize the following evaluation error for LTM's

\[
J(k) = \sum_{j=1}^{N_k} E_j^{TM}(k+j)
\]

(4)

where \(N_k\) is the number of LTM's at the \(k\)-th iteration, and \(E_j^{TM}\) is the error for the input pattern corresponding to the \(j\)-th LTM, \(x_j^{TM}, j=1, 2, \ldots, N_k\). Note that, this evaluation is carried out by using the input series corresponding to the LTM's as \((x(t_{k+1}) = x_i^{TM}, \ldots, x(t_{k+N_k}) = x_j^{TM}\) in the open-ended learning phase of the incremental learning task.

3. Memory Superimposition

3.1. Back-propagated Errors as Reference Values

Three-layer feedforward neural networks are considered here, but it can be applied to the feedforward networks with more than three layers as well. The number of neurons in the input (1st), the hidden (2nd) and the output (3rd) layer are \(n, n_h, \) and \(m, \) respectively. The outputs of the input layer is equal to the neural inputs \(y_i^{(1)} = x \in \mathbb{R}^n\) and the network outputs \(\hat{y} \in \mathbb{R}^m\) are the outputs of the output layer \(y_\hat{}^{(3)} = [y_1^{(3)} \ldots y_q^{(3)} \ldots y_m^{(3)}]^T \in \mathbb{R}^m\) defined as

\[
y_\hat{}^{(3)} = f^{(3)}(\sum_{h=1}^{n_h} w_h^{(3)} f^{(2)}(\sum_{p=1}^{n} w_p^{(2)} x_p + \theta_h^{(2)}) + \theta_\hat{}^{(3)})
\]

(5)

where \(f\) and \(\theta\) represent a nonlinear function and a neural threshold or bias, respectively. In the following, for simplicity the superscripts denoting the layer number are omitted unless the layer number is important.

Let \(\Delta w_{\text{ad}}(kT)\) denote the changes in weights \(w_{\text{ad}}(kT) \in \{w_{\text{ad}}^{(3)}, w_{\text{ad}}^{(2)}\}\) given by the discrete-time BP algorithm at the \(k\)-th iteration, and \(w_{\text{ad}}^{(3)}(k)\) denote the resulting weights at the \(k\)-th iteration such that

\[
w_{\text{ad}}^{(3)}(k) = w_{\text{ad}}(kT) + \Delta w_{\text{ad}}(kT)
\]

(6)
Using this notation, for \( kT \leq t_k < (k+1)T \), the references in (2) are defined as
\[
\tau_{ba}(t_k) = \frac{w_{ba}^{\text{ref}}(k) - a_{ba}(t_k - kT)}{\tau_{ba}}
\]  
(7)
where \( a_{ba} \) is the unipolar unit step function. Substituting the references for (2), and solving the differential equation (2) piece-wise for \( kT \leq t_k < (k+1)T \), we have
\[
w_{ba}(t_k) = w_{ba}^{\text{ref}}(k) - \Delta w_{ba}(kT) \exp\left(-\frac{t_k - kT}{\tau_{ba}}\right)
\]
Obviously, if \( \tau_{ba} \to 0 \), this dynamic change is equivalent to the discrete-time BP scheme (Fig. 2).

The proposed updating scheme of the time constants \( \tau_{ba} \) is defined as
\[
\tau_{ba}((k+1)T) = \tau_{ba}(kT) + \Delta \tau_{ba}(kT)
\]
(8)
Here, to transfer STM into LTM for \( x \), such that \( E_i \leq \gamma \)
1. All the initial values are zero: \( \tau_{ba}(0) = 0 \);
2. The following monotonous change is used
\[
\Delta \tau_{ba}(kT) = \begin{cases} 
0, & (E_i(k) > \gamma) \\
\infty, & (E_i(k) \leq \gamma)
\end{cases}
\]
(9)
The dynamic changing rule for biases \( \theta_{ba}^{(2)} \) and \( \theta_{ba}^{(3)} \) is the same as that for the weights \( w_{ba} \).

3.2. Structural Adaptation

Since all the existing weights cannot be changed after a long-term memory is stored by using the above updating rule of the time constants, the network is required to create new learnable connections for new memories.

To reduce the computational complexity of the neuron with DSCWs, a simple structural adaptation strategy is used in this paper. For an input \( x \), if the square error \( E_i \) as the minimum value for several sets of initial values on the learnable parameter space is not less than the tolerance \( \gamma \), a new hidden layer neuron with all the available connections from the input layer and to the output layer is added to the network. That is, when a new hidden neuron is added at the end of the \( k \)-th iteration, \( n_h(k+1) = n_h(k) + 1 \), otherwise \( n_h(k+1) = n_h(k) \).

3.3. Restoration of the Past Knowledge by Lateral Inhibition

Synapses with respect to a newly added hidden neuron are needed for new memories. However, since the new synapses do not store the past LTM, the network output corresponding to one of the past LTM's may be disturbed by the output of the newly added neuron. Thus, the output of the new neuron should be inhibited only when the one of the past LTM's is recalled.

To implement this restoration mechanism, a lateral inhibition mechanism is introduced in the hidden layer. We assume that neurons cannot fire when they receive signals not greater than a negative threshold, \( N_{ba}(<0) \), whose absolute value is large but bounded, \( |N_{ba}| < \infty \). Then inhibitory weights from the \( l \)-th neuron, \( l = 1, 2, \ldots, n_h(k) \), to the \( h \)-th neuron in the hidden layer are initialized by
\[
w_{ba}^{(2)}(0) = \begin{cases} 
N_{bh}, & (l > h) \\
0, & (l \leq h)
\end{cases}
\]
(10)
and they are fixed by \( \tau_{hl} \to \infty \), i.e, \( w_{ba}^{(2)}(t) = w_{ba}^{(2)}(0) \). Therefore, when a new hidden neuron is added, all the possible inhibitory connections from the existing neurons to the new learnable neuron \( w_{ba}^{(2)} \), \( l < h \), are created.

On the other hand, to decide whether the input is corresponding to the LTM or not, the inner product of the weights and the inputs, \( s_i = \mathbf{w}_i \cdot \mathbf{x} \), is used. The inner product corresponding to LTM, \( s^i \), is stored by the same dynamic rule as that of the weights
\[
t_{s_i} \frac{ds^i(t_k)}{dt} = -s^i(t_k) + s_i(kT)a_{ba}(t_k - kT)
\]
(11)
where $l=1, 2, \cdots, n_H(k)$. The updating rule is similar to that of $\Delta t_0$: $\Delta t_0 \rightarrow \infty$ when a LTM is formed ($E_r(k) \leq \gamma$), and otherwise $\Delta t_0 = 0$.

Then the lateral inhibitory signals, $s_{lh}$, are defined as

$$s_{lh}(t_k) = \begin{cases} N_{rh}, & (l < h \text{ and } |s_{lh}(t_k) - s_{lh}(t_k) - s_{lh}(t_k)| < \epsilon) \\ 0, & (\text{otherwise}) \end{cases}$$

where $\epsilon > 0$ is a sensitivity parameter. The recall decision is more sensitive to the input noise for smaller $\epsilon$, while for larger $\epsilon$ the decision is more robust to the noise. The robustness often causes incorrect decision, but contributes to the generalization ability of the network.

These inhibitory signals are summed as an external signal to the neuron in the hidden layer. Using this inhibitory signal, newly added hidden neurons cannot fire when the past LTMs are recalled. Thus a new memory is superimposed on the current network structure without disturbing the past LTMs by the newly added neurons.

4. Learning Ability

We evaluate learning ability of the memory superimposition neural network (MS-NN) compared with the conventional BP neural network (BP-NN). For this evaluation, let $\{x\}_k \triangleq \{x(t_0), x(t_1), \cdots, x(t_k)\}$ denote an input series, and $\{x\}_k$, be a homogeneous input series consisting of only one vector $x_t$, i.e., $x(t_0) = \cdots x(t_k) = x_t$.

We consider a BP-NN trained with the homogeneous input series $\{x\}_k$, and denote such BP-NN as $NN^{BP}(\{x\}_k)$. On the other hand, let $NN^{MS} \triangleq NN^{MS}(\{x\}_k)$ denote a proposed MS-NN trained with an arbitrary (heterogeneous) input series.

The ability to superimpose a new LTM on the past LTMs will now be discussed. Let $x^{LTM}_j$, $j \in \{1, 2, \cdots, N_k\}$ denote an input vector corresponding to a LTM. The evaluation error by a BP-NN trained with the homogeneous input corresponding to a LTM, $NN^{BP}(\{x^{LTM}\}_k)$, $j \in \{1, 2, \cdots, N_k\}$, is defined as

$$J_{BP} \triangleq E_j^{LTM}(k_j + 1, NN^{BP}(\{x^{LTM}\}_k))$$

The sum of such evaluation errors for all the LTMs by a set of the corresponding BP-NNs, $\{NN^{BP}(\{x^{LTM}\}_k), j \in \{1, 2, \cdots, N_k\}\}$, is given as

$$J_{BP} \triangleq \sum_{j=1}^{N_k} J_{BP}$$

Suppose that the evaluation error by the MS-NN, $J(k, NN^{MS})$, is equal to the above sum of evaluation errors by the set of BP-NNs, $J_{BP}$. Then it implies that the MS-NN truly includes all the LTMs in the set of BP-NNs in its single MS-NN structure, and that the accuracy of each LTM in the MS-NN is equal to that of the corresponding BP-NN. Under the condition of the error tolerance that $\gamma_j = J_{BP}$, the capability to achieve this strict incremental learning is summarized as follows.

**Theorem 1** (Memory superimposition theorem)

$$\forall \{NN^{BP}(\{x^{LTM}\}_k), j \in \{1, 2, \cdots, N_k\}\}, \exists NN^{MS} \text{ such that}$$

$$J(k, NN^{MS}) \rightarrow J_{BP}, \left(\min J^{LTM} \rightarrow \infty \right)$$

where $M^{LTM}$ is the number of learning iterations for the input $x_j^{LTM}$, and the tolerance is given as $\gamma_j = J_{BP}$.

The error tolerance $J_{BP}$ can be very small since the BP-NN learns only one training set, and thus the sum of evaluation errors by the set of BP-NNs, $J_{BP}$, could be smaller than that of a single BP-NN trained simultaneously with all the inputs $x^{LTM}_j$, $j \in \{1, 2, \cdots, N_k\}$. Note that we can have a MS-NN satisfying Theorem 1 as proved in the Appendix. In this sense, the
learning ability of the MS–NN for LTMs is superior to that of the BP–NN.

5. Simulation Example

The proposed MS–NN could directly implement the pattern classification in the strict learning tasks. In the following simulation study, an input vector \( \mathbf{x}_i \in \{0, 1\}^{64}, \ i \in \{1, 2, \ldots, 26\} \), consists of \( 64 (=8 \times 8) \) binary pixels of a capital letter pattern as shown in Fig. 3. The desired output vectors \( \mathbf{y}_i = [y_i^1, y_i^2, y_i^3, y_i^4, y_i^5]^T \in \{0, 1\}^5 \) are the corresponding binary numbers from 1 to 26, i.e., \((y_i^1, y_i^2, y_i^3, y_i^4, y_i^5) = (2^i, 2^{i-1}, 2^{i-2}, 2^{i-3}, 2^{i-4})\) for example, for pattern “A” (= \( \mathbf{x}^1 \)) the desired vector is \( \mathbf{y}_1 = [0, 0, 0, 0, 1]^T \) and \( \mathbf{y}_{26} = [1, 1, 0, 1, 0]^T \) for pattern “Z” (= \( \mathbf{x}_{26} \)). Then, the MS–NN structure was initially composed of \( 64 \times 1 \times 5 \) neurons in the input–hidden–output layers, respectively. The number of hidden neurons increased as they were needed. The nonlinear functions for all the neurons were the unipolar sigmoidal function, \( f_s(x) = 1/(1 + \exp(-x)) \).

For the \( k \)-th iteration, one of the 26 patterns is randomly chosen as the input \( \mathbf{x}(t_k) \). There is no restriction on the input selection, thus some patterns are allowed to be chosen repeatedly regardless of whether the patterns are memorized as LTM or not. Indeed, to transfer STM into LTM, corresponding input vectors should be rehearsed for enough durations as in the brain \(^7\).

Note that, it is not a very difficult task for BP–NNs if they were trained with all these 26 patterns simultaneously. However, in the open-ended learning phase, unlike in a conventional classification task, the network continues to “learn” the input pattern series of LTMs during the calculation of the evaluation error. Thus, since past LTMs in a BP–NN will be disturbed by the additional learning, the evaluation error will not be kept small even after each input pattern is learned completely. On the other hand, as proved in the Appendix the evaluation error of the MS–NN is kept small since LTMs are restored correctly regardless of whether learning of any patterns, including the patterns corresponding to LTMs, would be carried out or not. This advantage of the MS–NN is confirmed experimentally as shown in Fig. 4 where the bold and light solid lines show, respectively, the evaluation error of the MS–NN and that of the conventional BP–NN.

![Figure 3. 26 alphabet patterns of capital letters.](image-url)
6. Conclusions

In this paper, a memory superimposition (MS) method for incremental learning tasks has been developed using a novel learning, structural adaptation and restoration strategies. By using a lateral inhibition mechanism, a new network structure is superimposed on the previous structure without forgetting the past knowledge. Although the proposed MS-NN requires more complex structure and more memories, the MS-NN never forgets the past knowledge. This is a strict incremental learning that cannot be achieved by the existing incremental learning schemes.

Possible extensions to the radial basis function networks, dynamic networks with feedback, and self-learning methods such as Hebbian rule should lead to some interesting investigations in the future work.

References

6) N. Homma and M.M. Gupta: Incremental Neu-


A. **Proof for Theorem 1**

In this appendix we will prove Theorem 1 discussed in Section 4. We assume that the initial weights can be suitable to eliminate the effects of the initial conditions on the learning errors. We use the following notations: \( NN \triangleq NN((k+1)T) = NN(\{x\}_k) : \) Neural network trained with an arbitrary input series \( \{x\}_k = \{x(t_0), x(t_1), \ldots, x(t_N)\} ; N \) and \( N_L \) : The number of training sets and long-term memories.

A.1. **Proof for the First LTM**

First, we consider a learnable parameter space \( \mathcal{O}(NN) \) where the errors as functions of the network configuration, \( E(k, NN) \) and \( J(k, NN) \), exist. In the case of the first long-term memory \( x^{LT}_M \), for any BP-NN trained with only this input \( x^{LT}_M \), \( NN^{BP}(\{x^{LT}_M\}) \), we can have a proposed MS-NN, \( NN^{MS}(k'T) \), \( k' \in \{1, 2, \ldots\} \), which has the same space (the same number of learnable hidden neurons) as that of the conventional BP networks. That is, \( \forall NN^{BP}(\{x^{LT}_M\}) \), \( \exists NN^{MS}(k'T) \) such that

\[
\mathcal{O}(NN^{BP}(\{x^{LT}_M\})) = \mathcal{O}(NN^{MS}(k'T))
\]

where \( k_{j+1} \geq k' \). In other words, for each \( NN^{BP}(\{x^{LT}_M\}) \), a single proposed \( NN^{MS} \) can have the same learnable parameter space at the corresponding time \( k' \). Then, on the basis of the same reason for the proof for the first LTM, the MS-NN can reach any local minimum as \( M^{LTM}(j) \rightarrow \infty \) if each conventional network \( NN^{BP}(\{x^{LT}_M\}) \) can.

A.2. **Proof for a Special Type of \( NN^{BP} \)**

We consider a BP-NN, \( NN^{BP-\theta_q} \), which has a restriction of changes in biases of neurons in the output layer. For \( N \geq 1, w_{ao} \) and \( \theta_q \) storing LTM in the MS-NN yield fixed dendrite potentials or biases on output neurons \( \theta_q(i) \) for each input \( x_i \). Thus the MS-NN is equivalent to a network with the fixed biases of the output neurons, \( \theta_q(i) \). That is, \( \forall \{NN^{BP}(\{x^{LTM}_M\}), j = 1, 2, \ldots, N_L \}, \exists \theta_q^{LTM}(j) \) and \( \exists NN^{MS}(k'T) \) such that

\[
\mathcal{O}(NN^{BP-\theta_q^{LTM}}(\{x^{LTM}_M\})) = \mathcal{O}(NN^{MS}(k'T))
\]

where \( k_{j+1} \geq k' \). In other words, for each \( NN^{BP-\theta_q^{LTM}}(\{x^{LTM}_M\}) \), a single proposed \( NN^{MS} \) can have the same learnable parameter space at the corresponding time \( k' \). Then, on the basis of the same reason for the proof for the first LTM, the MS-NN can reach any local minimum as \( M^{LTM}(j) \rightarrow \infty \) if each conventional network \( NN^{BP-\theta_q^{LTM}}(\{x^{LTM}_M\}) \) can.

A.3. **Complete Proof**

The bias vector \( \theta_q(i) \) for each input \( x_i \) is a disadvantage of the MS-NN, but the fact that the proposed method is a strict incremental learning scheme can be a significant advantage itself. As described in Section 4, it does not mean that the conventional BP-NNs are superior to the MS-NNs because in general, a conventional BP-NN doesn't satisfy the theorem for any \( N \geq 1 \).

However, a complete solution is also avail-
able if we introduce a self inhibition mechanism for the biases of output neurons.

\[ \theta(t_k) = \begin{cases} \theta^i(t_k), & (|s^i(t_k) - \bar{s}^i(t_k)| < \varepsilon) \\ \theta^o(t_k), & \text{(otherwise)} \end{cases} \] (15)

Here \( s^i \) and \( \theta^i \) are, respectively, the inner product and the bias for the input corresponding to the \( j \)-th LTM. The \( s^i \) and \( \theta^i \) can be memorized by the same dynamics of the inner product \( s^f \) described in Subsection 3.3.

If we use this extra mechanism, Theorem 1 is satisfied completely by the MS-NN, although this requires more complicated structures and memories. The MS-NNs with the extra mechanism are then superior clearly than any conventional BP-NNs in the sense of the learning ability described in Theorem 1. \( \square \)