Bull. Coll. Med. Sci. Tohoku Univ. **12**(2) : 111～120, 2003

Memory Superimposition by Backpropagation Neural Networks

Noriyasu HOMMA[1] and Madan M. GUPTA[2]

[1]*Department of Radiological Technology College of Medical Sciences, Tohoku University*
[2]*Intelligent Systems Research Laboratory College of Engineering, University of Saskatchewan*

誤差逆伝播ネットワークによる重ね書き記憶

本間経康[1], Madan M. GUPTA[2]

[1]東北大学医療技術短期大学部　診療放射線技術学科
[2]サスカッチワン大学工学部　知的システム研究所

Key words: Neural networks, incremental learning, pattern classification and long-term memory

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[1]. 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[2),3)], while in the other one for the radial basis function networks[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[2)], 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

$$w(k+1)=w(k)+\Delta w(k) \tag{1}$$

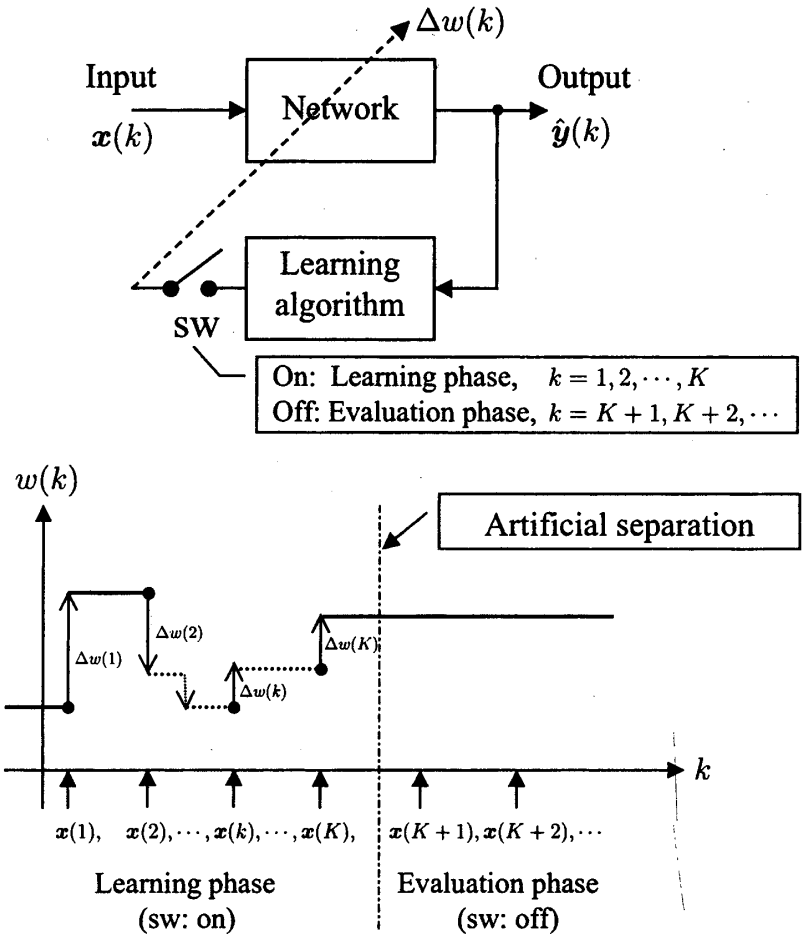

Figure 1. Two separate phases: learning phase and evaluation phase. $\boldsymbol{x}(k)$ and $\hat{\boldsymbol{y}}(k)$ denote the input and output vectors at the k-th iteration, respectively.

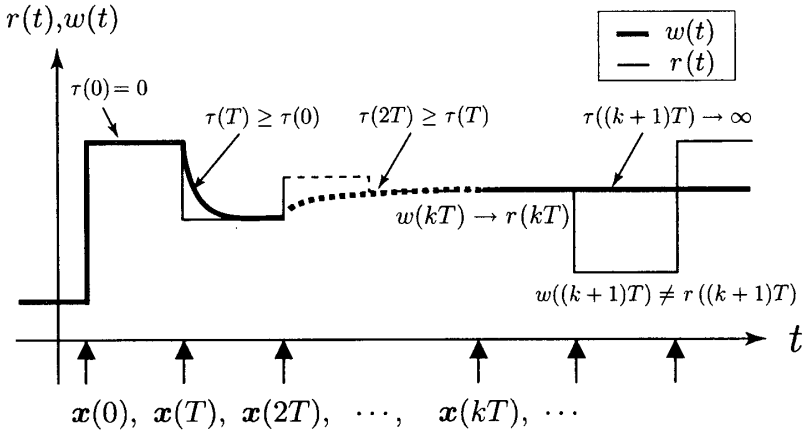

Figure 2. A dynamic changing weight by a piece-wise constant reference in the *open-ended learning phase*. Using *monotonous* increment of the time constant, no change in the weight takes place for a very large time constant (*e.g*, $\tau \rightarrow \infty$) even if the references for new inputs are not equal to the weight $w(t)$, $t > kT$. The reference $r(kT)$ is stored as LTM in this example.

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 $\boldsymbol{x}(k)$ 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 (LTMs) 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 LTMs[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.

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

where τ (>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 $\{\boldsymbol{x}_i \in \Re^n,\ i=1, 2, \cdots, N\}$ and the corresponding desired output signals $\{\boldsymbol{y}_i \in \Re^m,\ i=1, 2, \cdots, N\}$ 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, $kT \leq t_k < (k+1)T,\ k=0, 1, \cdots$. 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, $(\boldsymbol{x}(t_k), \boldsymbol{y}(t_k))=(\boldsymbol{x}_i, \boldsymbol{y}_i),\ i \in \{1, 2, \cdots, N\}$.

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

$$E_i(k)=\lim_{t_k \to (k+1)T} \| \hat{\boldsymbol{y}}(t_k)-\boldsymbol{y}_i \| \qquad (3)$$

where $\hat{\boldsymbol{y}} \in \Re^m$ 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 $\boldsymbol{x}(t_k)=\boldsymbol{x}_i$ 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 LTMs

$$J(k)=\sum_{j=1}^{N_k} E_j^{LTM}(k+j) \qquad (4)$$

where N_k is the number of LTMs at the k-th iteration, and E_j^{LTM} is the error for the input pattern corresponding to the j-th LTM, $\boldsymbol{x}_j^{LTM}$, $j=1, 2, \cdots, N_k$. Note that, this evaluation is carried out by using the input series corresponding to the LTMs as $\{\boldsymbol{x}(t_{k+1})=\boldsymbol{x}_1^{LTM}, \cdots, \boldsymbol{x}(t_{k+N_k})=\boldsymbol{x}_{N_k}^{LTM}\}$ 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 $\boldsymbol{y}^{(1)}=\boldsymbol{x} \in \Re^n$ and the network outputs $\hat{\boldsymbol{y}} \in \Re^m$ are the outputs of the output layer $\boldsymbol{y}^{(3)}=[y_1^{(3)} \cdots y_q^{(3)} \cdots y_m^{(3)}]^T \in \Re^m$ defined as

$$y_q^{(3)}=f_q^{(3)}\left(\sum_{h=1}^{n_H} w_{qh}^{(3)} f_h^{(2)}\left(\sum_{p=1}^{n} w_{hp}^{(2)} x_p+\theta_h^{(2)}\right)+\theta_q^{(3)}\right) \qquad (5)$$

where f and θ 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 $\Delta w_{ba}(kT)$ denote the changes in weights $w_{ba}(kT) \in \{w_{qh}^{(3)}, w_{hp}^{(2)}\}$ given by the discrete-time BP algorithm at the k-th iteration, and $w_{ba}^{BP}(k)$ denote the resulting weights at the k-th iteration such that

$$w_{ba}^{BP}(k) \triangleq w_{ba}(kT)+\Delta w_{ba}(kT) \qquad (6)$$

[1] This neural model does provide a relation between STM and LTM, however, it does not have positive and reliable biological evidences. The relation seems to be not very clear yet in biological processes.

Using this notation, for $kT \leq t_k < (k+1)T$, the references in (2) are defined as

$$r_{ba}(t_k) \triangleq w_{ba}^{BP}(k) u_0(t_k - kT) \qquad (7)$$

where u_0 is the unipolar unit step function. Substituting the references for (2), and solving the differential equation (2) piece-wisely for $kT \leq t_k < (k+1)T$, we have

$$w_{ba}(t_k) = w_{ba}^{BP}(k) - \Delta w_{ba}(kT) \exp\left(-\frac{t_k - kT}{\tau_{ba}}\right)$$

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 τ_{ba} is defined as

$$\tau_{ba}((k+1)T) = \tau_{ba}(kT) + \Delta\tau_{ba}(kT) \qquad (8)$$

Here, to transfer STM into LTM for $\boldsymbol{x}_i$ such that $E_i \leq \gamma$

1. All the initial values are zero: $\tau_{ba}(0) = 0$;
2. The following *monotonous* change is used

$$\Delta\tau_{ba}(kT) = \begin{cases} 0, & (E_i(k) > \gamma) \\ \infty, & (E_i(k) \leq \gamma) \end{cases} \qquad (9)$$

The dynamic changing rule for biases $\theta_h^{(2)}$ and $\theta_q^{(2)}$ 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[6], a simple structural adaptation strategy is used in this paper. For an input $\boldsymbol{x}_i$, 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 γ, 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* LTMs, the network output corresponding to one of the past LTMs 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 LTMs 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, $N_{th}(<0)$, whose absolute value is large but bounded, $|N_{th}| < \infty$. Then inhibitory weights from the l-th neuron, $l = 1, 2, \cdots, n_H(k)$, to the h-th neuron in the hidden layer are initialized by

$$w_{hl}^{(2)}(0) = \begin{cases} N_{th}, & (l > h) \\ 0, & (l \leq h) \end{cases} \qquad (10)$$

and they are fixed by $\tau_{hl} \to \infty$, i.e, $w_{hl}^{(2)}(t) = w_{hl}^{(2)}(0)$. Therefore, when a new hidden neuron is added, all the possible inhibitory connections from the *existing* neurons to the new *learnable* neuron $w_{hl}^{(2)}$, $l < h$, 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, $s_l = \boldsymbol{w}_l \cdot \boldsymbol{x}$, is used. The inner product corresponding to LTM, s_l^L, is stored by the same dynamic rule as that of the weights

$$\tau_{s_l} \frac{ds_l^L(t_k)}{dt} = -s_l^L(t_k) + s_l(kT) u_0(t_k - kT) \qquad (11)$$

where $l=1, 2, \cdots, n_H(k)$. The updating rule is similar to that of τ_{ba} : $\Delta\tau_{sl} \to \infty$ when a LTM is formed ($E_i(k) \leq \gamma$), and otherwise $\Delta\tau_{sl}=0$.

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

$$s_{hl}(t_k)=\begin{cases} N_{th}, & (l<h \text{ and } |s_l(t_k)-s_l^L(t_k)|<\varepsilon) \\ 0, & (\text{otherwise}) \end{cases} \tag{12}$$

where $\varepsilon>0$ is a *sensitivity* parameter. The *recall* decision is more sensitive to the input noise for smaller ε, while for larger ε 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 $\{\boldsymbol{x}\}_k \triangleq \{\boldsymbol{x}(t_0), \boldsymbol{x}(t_1), \cdots, \boldsymbol{x}(t_k)\}$ denote an input series, and $\{\boldsymbol{x}\}_{k_i}^i$ be a homogeneous input series consisting of only one vector $\boldsymbol{x}_i$, i.e, $\boldsymbol{x}(t_0)=\cdots \boldsymbol{x}(t_{k_i})\equiv\boldsymbol{x}_i$.

We consider a BP-NN trained with the *homogeneous* input series $\{\boldsymbol{x}\}_{ki}^i$ and denote such BP-NN as $NN^{BP}(\{\boldsymbol{x}\}_{k_i}^i)$. On the other hand, let $NN^{MS} \triangleq NN^{MS}(\{\boldsymbol{x}\}_k)$ 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 $\boldsymbol{x}_j^{LMT}$, $j \in \{1, 2, \cdots, N_k\}$ denote an input vector corresponding to a LTM. The evaluation error by a BP-NN trained with the homogeneous input corresponding to a LTM, $NN^{BP}(\{x^{LTM}\}_{kj}^j)$, $j \in \{1, 2, \cdots, N_k\}$, is defined as

$$J_j^{BP} \triangleq E_j^{LTM}(k_j+1, NN^{BP}(\{\boldsymbol{x}^{LTM}\}_{k_j}^j))$$

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

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

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 $\gamma_j \equiv J_j^{BP}$, the capability to achieve this *strict* incremental learning is summarized as follows.

Theorem 1 (Memory superimposition theorem) $\forall\{NN^{BP}(\{\boldsymbol{x}^{LTM}\}_{k_j}^j), j=1, 2, \cdots, N_k\}$, $\exists NN^{MS}$ such that

$$J(k, NN^{MS}) \to J^{BP}, \left(\min_j M_j^{LTM} \to \infty\right) \tag{13}$$

where M_j^{LTM} is the number of learning iterations for the input $\boldsymbol{x}_j^{LTM}$, and the tolerance is given as $\gamma_j=J_j^{BP}$. ■

The error tolerance J_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 $\{\boldsymbol{x}_j^{LTM}, j=1, 2, \cdots, N_k\}$. 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 $\boldsymbol{x}_i \in \{0, 1\}^{64}$, $i \in \{1, 2, \cdots, 26\}$, consists of 64($=8\times8$) binary pixels of a capital letter pattern as shown in Fig. 3. The desired output vectors $\boldsymbol{y}_i=[y_4^i\ y_3^i\ y_2^i\ y_1^i\ y_0^i]^T \in \{0, 1\}^5$ are the corresponding binary numbers from 1 to 26, i.e, $(y_4^i\ y_3^i\ y_2^i\ y_1^i\ y_0^i)=(2^4\ 2^3\ 2^2\ 2^1\ 2^0)$: for example, for pattern "A" ($=\boldsymbol{x}^1$) the desired vector is $\boldsymbol{y}_1=[0\ 0\ 0\ 0\ 1]^T$ and $\boldsymbol{y}_{26}=[1\ 1\ 0\ 1\ 0]^T$ for pattern "Z" ($=\boldsymbol{x}_{26}$). Then, the MS-NN structure was initially composed of 64-1-5 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, $f_b(x)=1/(1+\exp(-x))$.

For the k-th iteration, one of the 26 patterns is randomly chosen as the input $\boldsymbol{x}(t_k)$. 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.

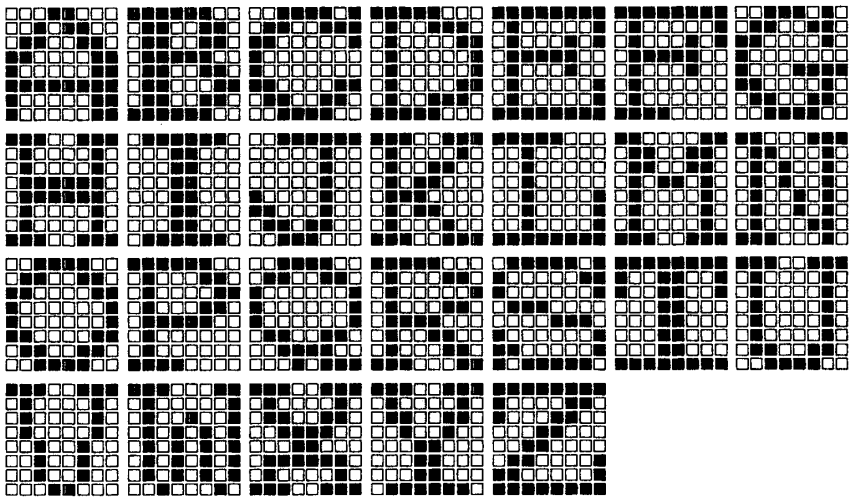

Figure 3. 26 alphabet patterns of capital letters.

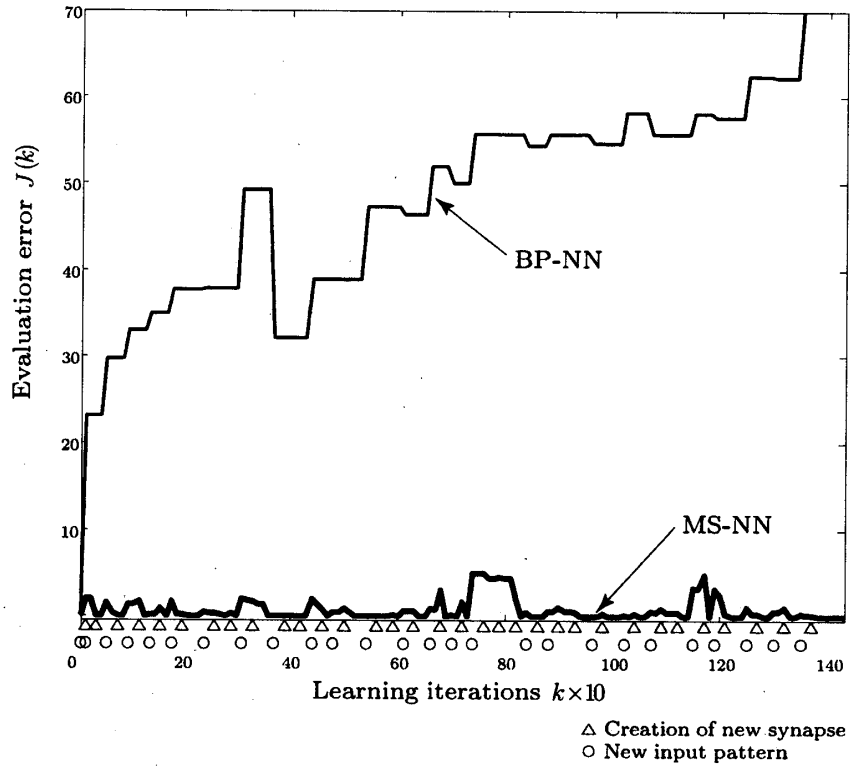

Figure 4. The evaluation errors as functions of learning iterations. The bold and light solid lines show the evaluation errors of the proposed MS-NN and the conventional BP-NN, respectively.

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

1) A. Cichochi and R. Unbehauen: *Neural Networks for Optimization and Signal Processing*, Wiley, Chichester (1993).
2) L. Fu, H.H. Hsu and J.C. Principe: Incremental Backpropagation Learning Networks, *IEEE Trans. Neural Networks*, **7**, 757/761 (1996).
3) L. Fu: Incremental Knowledge Acquisition in Supervised Learning Networks, *IEEE Trans. Syst., Man, Cybern. A*, **26**, 801/809 (1996).
4) K. Yamauchi, N. Yamaguchi and N. Ishii: Incremental Learning Methods with Retrieving of Interfered Patterns, *IEEE Trans. Neural Networks*, **10**, 1351/1365 (1999).
5) Specht, D.F.: A General Regression Neural Network, *IEEE Trans. Neural Networks*, **2**, 568/576 (1991).
6) N. Homma and M.M. Gupta: Incremental Neu-

ral Learning by Dynamic and Spatial Changing Weights, *Proc. 15th IFAC World Congress*, July, (2002 accepted).

7) M.M. Gupta, L. Jin and N. Homma: Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory, Wiley, New York, (2003).

8) M. Hiramoto, Y. Hiromi, E. Giniger and Y. Hotta: The Drosophila Netrin Receptor Frazzled Guides Axons by Controlling Netrin Distribution, *Nature*, **406**, 886/888 (2000).

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: $NN \triangleq NN((k+1)T) \equiv NN(\{\boldsymbol{x}\}_k)$: Neural network trained with an arbitrary input series $\{\boldsymbol{x}\}_k = \{\boldsymbol{x}(t_0), \boldsymbol{x}(t_1), \cdots, \boldsymbol{x}(t_k)\}$; 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* $\Omega(NN)$ where the errors as functions of the network configuration, $E_i(k, NN)$ and $J(k, NN)$, exist. In the case of the first long-term memory $\boldsymbol{x}_1^{LTM}$, for any BP-NN trained with only this input $\boldsymbol{x}_1^{LTM}$, $NN^{BP}(\{\boldsymbol{x}^{LTM}\}_{k_1}^1)$, we can have a proposed MS-NN, $NN^{MS}(k'T)$, $k' \in \{1, 2, \cdots\}$, which has the same space (the same number of *learnable* hidden neurons) as that of the conventional BP networks. That is, $\forall NN^{BP}(\{\boldsymbol{x}^{LTM}\}_{k_1}^1)$, $\exists NN^{MS}(k'T)$ such that

$$\Omega(NN^{BP}(\{\boldsymbol{x}^{LTM}\}_{k_1}^1)) = \Omega(NN^{MS}(k'T)) \quad (14)$$

Local minima of the errors on the space are not changed during the learning, because the space $\Omega(NN^{BP}(\{\boldsymbol{x}^{LTM}\}_{k_1}^1))$ and the error functions are not changed by the conventional scheme. Also the space $\Omega(NN^{MS}(k'T))$ and the error functions are not changed by the MS-NN structure unless the learning is trapped in a local minimum which is larger than the error tolerance.

Since both the networks use the same reference value, the MS-NN can reach any local minimum on the space in (14) as $M_1^{LTM} \to \infty$ if the conventional BP-NNs can. This proves the theorem for the first LTM. □

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 $N_L \geq 1$, $\boldsymbol{w}_{ba}$ and $\boldsymbol{\theta}_b$ storing LTMs in the MS-NN yield fixed dendrite potentials or biases on output neurons $\boldsymbol{\theta}_q(i)$ for each input $\boldsymbol{x}_i$. Thus the MS-NN is equivalent to a network with the fixed biases of the output neurons, $\boldsymbol{\theta}_q(i)$. That is, $\forall \{NN^{BP}(\{\boldsymbol{x}^{LTM}\}_{k_j}^j), j = 1, 2, \cdots, N_L\}$, $\exists \boldsymbol{\theta}_q^{LTM}(j)$ and $\exists NN^{MS}(k_j'T)$ such that

$$\Omega(NN^{BP-\theta_q^{LTM}(j)}(\{\boldsymbol{x}^{LTM}\}_{k_j}^j)) = \Omega(NN^{MS}(k_j'T)) \quad j = 1, 2, \cdots, N_L$$

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

A.3. Complete Proof

The bias vector $\boldsymbol{\theta}_q(i)$ for each input $\boldsymbol{x}_i$ 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 $N_L \geq 1$.

However, a complete solution is also avail-

able if we introduce a self inhibition mechanism for the biases of output neurons.

$$\theta_q(t_k)=\begin{cases}\theta_q^j(t_k), & (|s_q(t_k)-s_q^j(t_k)|<\varepsilon)\\ \theta_q(t_k), & (\text{otherwise})\end{cases} \tag{15}$$

Here s_q^j and θ_q^j are, respectively, the inner product and the bias for the input corresponding to the j-th LTM. The s_q^j and θ_q^j can be memorized by the same dynamics of the inner product s_l^L 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. □