Reducing Computations in Incremental Learning for Feedforward Neural Network with Long-Term Memory

Masataka Kobayashi[†] Anuar Zamani[‡] Seiichi Ozawa[†] Shigeo Abe[†]

† Graduate School of Science and Technology, Kobe University, Kobe 657-8501, JAPAN
‡ Faculty of Engineering, Kobe University, Kobe 657-8501, JAPAN
kobayasi@chevrolet.eedept.kobe-u.ac.jp {ozawa, abe}@eedept.kobe-u.ac.jp

Abstract

When neural networks are trained incrementally, input-output relationships that are trained formerly tend to be collapsed by the learning of new training data. This phenomenon is called "interference". To suppress the interference, we have proposed an incremental learning system (called RAN-LTM), in which Long-Term Memory (LTM) is introduced into Resource Allocating Network (RAN). Since RAN-LTM needs to train not only new data but also some LTM data to suppress the interference, if many LTM data are retrieved, large computations are required. Therefore, it is important to design appropriate procedures for producing and retrieving LTM data in RAN-LTM. In this paper, these procedures in the previous version of RAN-LTM are improved. In simulations, the improved RAN-LTM is applied to the approximation of a one-dimensional function, and the approximation error and the training speed are evaluated as compared with RAN and the previous RAN-LTM.

1 Introduction

In real world problems, training data are often given incrementally to learning systems. It is well known that the learning of neural networks becomes difficult especially when the distribution of given training data is temporally varied. In such a situation, the input-output relations acquired in the past are easy to be collapsed by the learning of new data. This disruption in neural networks is called "interference" that is caused by excessive adaptation of connection weights for new data.

There have been proposed several approaches to suppression of the interference. They can be categorized into two types. In the first approach, the connection weights trained formerly are not modified by new training data as much as possible; that is, connection weights adapted for new data are separated from those for old data. Although this approach is easily implemented by scaling up networks (i.e. adding extra hidden units or module networks) [1, 2], the problem is that the scale of networks tends to be large with the increase of training data. In the second approach, some data as well as newly given data are simultaneously trained in neural networks to suppress the interference. Several methods to generate these additional data have been proposed so far [3, 4]. Yamakawa has proposed Active Data Selection (ADS) in which the contribution of additional data to the approximation accuracy of neural networks are evaluated each time of the learning, and significant data are left in a storage buffer (short-term memory) [3]. Yamauchi has proposed a different type of incremental learning system in which storage data in a buffer are dynamically produced based on the estimation of interference caused by given training data [4]. In these approaches, all storage data in a buffer are trained with new data. The second approach has an advantage that the scale of network does not become so large even with the increase of training data. However, the computation costs of learning tend to be higher as compared with the first approach because the number of training data at one time is larger in almost all cases.

We have proposed an incremental learning system based on the second approach [5], in which Long-Term Memory (LTM) was introduced into Resource Allocating Network (RAN) [6]. For the notational convenience, this system is noted as "RAN-LTM". In RAN-LTM, storage data in LTM (noted as "LTM data") are produced from inputs and outputs of networks whose relationships are accurately approximated. LTM data to be recalled are selected based on the distance to center vectors of active hidden units; that is, LTM data that have smaller distance to these center vectors are frequently retrieved.

Although a suitable number of LTM data tends to be recalled in RAN-LTM, the computation costs of learning are still high to hold good approximation accuracy. Therefore, the reduction of learning time is one of the significant issues in our system.

In this paper, we propose a new version of RAN-LTM whose approximation ability and computation costs are enhanced. Section 2 briefly explains the learning algorithm of RAN. In Section 3, some problems of the previous version of RAN-LTM are described, then we propose a new RAN-LTM to solve these problems. To evaluate the performance, the proposed system is applied to a function approximation problem in Section 4. In Section 5, we present conclusions of this work.

2 Resource Allocating Network

Resource Allocating Network (RAN) proposed by Platt [6] is an extended version of Radial Basis Function networks [7]. When the training gets started in RAN, the number of hidden units is set to one initially; hence, RAN possesses simple approximation ability at first. As the training proceeds, the approximation ability of RAN is developed with the increase of training data by allocating additional hidden units. Therefore, RAN can adapt itself to dynamical expansion of training domain.

As shown in Eqs. (1)-(2), outputs of hidden units, $\boldsymbol{y} = \{y_1, \cdots, y_J\}'$, are calculated based on the distance, d_j, between the pth network input, $\boldsymbol{x}_p = \{x_{p1}, \cdots, x_{pI}\}'$, and center vectors of hidden units, $\boldsymbol{c}_j = \{c_{j1}, \cdots, c_{jI}\}'$ $(j = 1, \cdots, J)$.

$$d_j = \| \boldsymbol{x}_p - \boldsymbol{c}_j \|^2 = \sum_{i=1}^{I} (x_{pi} - c_{ji})^2 \quad (j = 1, \cdots, J) \tag{1}$$

$$y_j = \exp(-\frac{d_j}{{\sigma_j}^2}) \quad (j = 1, \cdots, J) \tag{2}$$

Here, I and J are respectively the number of input units and hidden units. σ_j^2 corresponds to variance of the jth radial basis function. Network outputs, $\boldsymbol{z} = \{z_1, \cdots, z_K\}'$, are calculated as follows:

$$z_k = \sum_{j=1}^{J} w_{kj} y_j + \gamma_k \quad (k = 1, \cdots, K), \tag{3}$$

where K is the number of output units. w_{kj} and γ_k are a connection weight from the jth hidden unit to the kth output unit and a bias of the kth output unit, respectively.

When a training datum is given to RAN, network outputs are calculated based on Eqs. (1)-(3), and the error, E, between the outputs, $\boldsymbol{z}$, and the targets for the pth inputs, $\boldsymbol{T}_p$, is evaluated. Based on the value of E, the following procedures are selected:

(1) $E > \varepsilon$ and $\|\boldsymbol{x}_p - \boldsymbol{c}^*\| > \delta(t)$
If E is larger than a positive constant, ε, and the distance between the pth input, $\boldsymbol{x}_p$, and its nearest center vector, $\boldsymbol{c}^*$, is larger than a positive value, $\delta(t)$, a hidden unit is added to RAN (i.e. $J \leftarrow J + 1$). Then, the network parameters for the Jth hidden unit (center vector, $\boldsymbol{c}_J$, connection weights, w_{kJ}, and variance, σ_J) are respectively set to the following values:

$$c_{Ji} = x_{pi} \quad (i = 1, \cdots, I) \tag{4}$$

$$w_{kJ} = T_{pk} - z_k \quad (k = 1, \cdots, K) \tag{5}$$

$$\sigma_J = \kappa \|\boldsymbol{x}_p - \boldsymbol{c}^*\|, \tag{6}$$

where κ is a positive constant. $\delta(t)$ decreases with time t as follows:

$$\delta(t) = \max[\delta_{max} \exp(\frac{t}{\tau}), \delta_{min}] > 0, \tag{7}$$

where τ is a decay constant.

(2) Otherwise
The network parameters are modified as follows:

$$w_{kj}^{NEW} = w_{kj}^{OLD} + \alpha(T_{pk} - z_k) y_j \tag{8}$$

$$c_{ji}^{NEW} = c_{ji}^{OLD} + 2\frac{\alpha}{\sigma_j}(x_{pi} - c_{ji}) y_j \sum_k (T_{pk} - z_k) \sigma_{kj} \tag{9}$$

$$\gamma_k^{NEW} = \gamma_k^{OLD} + \alpha(T_{pk} - z_k), \tag{10}$$

where α is a positive learning ratio.

As described before, the approximation ability of RAN can be developed even when training data are incrementally given. However, the suppression of the interference is not considered in RAN. Therefore, it is considered that the interference cannot be suppressed completely only by the automatic allocation of hidden units. In the next section, we describe an extended model of RAN in which Long-Term Memory (LTM) is introduced.

3 RAN with Long-Term Memory

3.1 Previous Model

In RAN-LTM, data stored in LTM (called "LTM data" for short) are utilized for suppressing the interference. Therefore, it is important to design appropriate procedures for producing and retrieving LTM data. Due to the space limitation, only these two procedures in RAN-LTM are described in the followings.

Production and retrieval of LTM data are simultaneously carried out during the learning of networks. Figures 1 and 2 show the procedures for producing and retrieving LTM data.

1) If all outputs of hidden units, y_j $(j = 1, \cdots, J)$, for the pth input are less than a threshold value, θ_c, then go to Step 5. Otherwise, go to Step 2.
2) Obtain all indices, j, of hidden units whose outputs, y_j, are larger than θ_c, and we define a set, $\mathcal{I}_1$, of these indices. The following approximation criterion, r_j, is updated for all $j \in \mathcal{I}_1$:
$$r_j \leftarrow r_j + f(\rho E(\boldsymbol{x}_p, \boldsymbol{T}_p)),$$
$$\text{where} \quad f(u) = 2\exp(-u) - 1.$$
Here, $E(\boldsymbol{x}_p, \boldsymbol{T}_p)$ is the error between the target, $\boldsymbol{T}_p$, and the output for the pth input, $\boldsymbol{x}_p$. ρ is a positive constant.
3) If $r_j > \beta$ for $j \in \mathcal{I}_1$, then r_j is initialized and go to Step 4. Otherwise, go to Step 5. Here, β is a positive constant.
4) If no LTM datum for the jth hidden unit had been produced yet, then the jth center vector, $\boldsymbol{c}_j$, is given to the network as its input, $\tilde{\boldsymbol{x}}_M$, and the output, $\tilde{\boldsymbol{z}}_M$ is calculated. $(\tilde{\boldsymbol{x}}_M, \tilde{\boldsymbol{z}}_M)$ is stored into LTM as the Mth LTM data. The number of LTM data, M, increases by one (i.e. $M \leftarrow M + 1$).
5) $p \leftarrow p + 1$ and go to Step 1.

Figure 1: Procedure for producing LTM data.

1) Obtain all indices, j, of hidden units whose outputs, y_j, are larger than θ_r, and define a set of these indices as $\mathcal{I}_2$.
2) If $\mathcal{I}_2 \neq \phi$, then go to Step 3. Otherwise, no LTM datum is retrieved and goes to Step 5.
3) For all hidden units belonging to $\mathcal{I}_2$, obtain LTM data that have the nearest distance to center vectors, $\boldsymbol{c}_j$ $(j \in \mathcal{I}_2)$.
4) Recall these LTM data and modify connection weights with them as well as new training data.
5) $p \leftarrow p + 1$ and go to Step 1.

Figure 2: Procedure for retrieving LTM data.

As seen in Fig. 2, the number of LTM data to be retrieved is equivalent to the number of active hidden units. Therefore, in general, so much computation costs of learning are not needed to suppress the interference in RAN-LTM. However, the above RAN-LTM has the following problems in the production of LTM data.

i) As mentioned above, the maximum number of LTM data is restricted to the number of hidden units; that is, only one LTM datum is produced for each hidden unit. If the variance parameters of hidden units are set to large value in order to enhance the generalization ability of networks, the number of hidden units becomes small; hence the number of LTM data also becomes small. Consequently, due to the shortage of LTM data, it becomes difficult that RAN-LTM suppresses the interference completely.

ii) As the learning proceeds, center vectors of hidden units are varied to refine the approximation ability. If this variation is large, the center vectors might be greatly different from the LTM data that were previously produced from the corresponding hidden units. In this situation, retrieved LTM data might be also greatly different from the center vectors of active hidden units; hence, it might be difficult to suppress the interference efficiently.

On the other hand, considering that the parameter modification for hidden units depends on their output values, the following problems can arise through the retrieval of LTM data in RAN-LTM.

iii) If a threshold, θ_r, is set to a large value, only the nearest LTM data to the center vectors of highly active hidden units are recalled. In this case, another interference can arise by the learning of these LTM data.

iv) If θ_r is set to a small value in order to avoid the above problem, the number of retrieved LTM data increases; hence, the computation costs in RAN-LTM also increases.

3.2 Improved Model

To solve the above problems i)-iv), the procedures for producing and retrieving LTM data are improved. Concretely, the following points in the previous RAN-LTM are modified:

a) For each hidden unit, only a LTM datum can be produced from the center vector.

b) The retrieval of LTM data is determined based on only the activation of hidden units.

3.2.1 Procedure for Producing LTM Data. As stated in 3.1, an insufficient number of LTM data causes the interference and the degradation of approximation ability. To solve this problem, the number of LTM data to be produced should be large as much as possible. As we might expect, however, the excessive production of LTM data results in the increase of computation costs of learning. Hence, LTM data should be produced adequately such that each LTM datum is distant from the others to some extent. Considering this point, we propose an improved procedure for producing LTM data shown in Fig. 3.

1) If all outputs of hidden units, y_j ($j = 1, \cdots, J$), for the pth input are less than a threshold value, θ_c, then go to Step 6. Otherwise, go to Step 2.
2) Obtain all indices, j, of hidden units whose outputs, y_j, are larger than θ_c, and we define a set, $\mathcal{I}_1$, of these indices. The following approximation criterion, r_j, is updated for all $j \in \mathcal{I}_1$:
$$r_j \leftarrow r_j + f(\rho E(\boldsymbol{x}_p, \boldsymbol{T}_p)),$$
$$\text{where} \quad f(u) = 2\exp(-u) - 1.$$
Here, $E(\boldsymbol{x}_p, \boldsymbol{T}_p)$ is the error between the target, $\boldsymbol{T}_p$, and the output for the pth input, $\boldsymbol{x}_p$. ρ is a positive constant.
3) If $r_j > \beta$ for $j \in \mathcal{I}_i$, then r_j is initialized and go to Step 4. Otherwise, go to Step 6. Here, β is a positive constant.
4) Calculate the distance between the jth center vector, $\boldsymbol{c}_j$, and the nearest LTM data, $(\tilde{\boldsymbol{x}}^*, \tilde{\boldsymbol{z}}^*)$: $\|\boldsymbol{c}_j - \tilde{\boldsymbol{x}}^*\|$. If the distance is larger than a positive constant, η, then go to Step 5. Otherwise, go to Step 6.
5) The jth center vector, $\boldsymbol{c}_j$, is given to the network as its input, $\tilde{\boldsymbol{x}}_M$, and the output, $\tilde{\boldsymbol{z}}_M$ is calculated. $(\tilde{\boldsymbol{x}}_M, \tilde{\boldsymbol{z}}_M)$ is stored into LTM as the Mth LTM data. The number of LTM data, M, increases by one (i.e. $M \leftarrow M + 1$).
6) $p \leftarrow p + 1$ and go to Step 1.

Figure 3: New procedure for producing LTM data.

3.2.2 Procedure for Retrieving LTM Data. Needless to say, the number of LTM data to be recalled should be large as much as possible in order to suppress the interference completely. However, this results in the increase of learning time. Hence, it is important to devise an efficient procedure for retrieving LTM data; that is, the number of retrieved LTM data becomes as small as possible without increasing approximation error. To realize this, we utilize the curvature information of the approximated function as well as the activation information of hidden units.

Table 1: Mean square errors of RAN-LTM, in which only LTM data on the part of the approximated function with high (or low) curvature, are recalled.

	High Curvature	Low Curvature
g_1	0.00496	0.04416
g_2	0.00063	0.00469
g_3	0.00063	0.00144
g_4	0.00164	0.02908
g_5	0.00131	0.00421
Ave.	0.00183	0.01672

Table 1 illustrates the results of a preliminary experiment. In this experiment, the approximation errors of RAN-LTM, in which only LTM data on the part of the approximated function with high (or low) curvature are recalled, are estimated for 5 different target functions: $g_1 \sim g_5$. As seen in Table 1, the approximation error is small when only LTM data on high curvature parts are trained in RAN-LTM. This experiment suggests that LTM data on high curvature parts are more useful for suppressing the interference as compared with those on low curvature points. Therefore, it is expected that if the retrieval of LTM data is determined based on the curvature information as well as the activation of hidden units, we can decrease the number of LTM data to be recalled.

Considering these results, we propose a new procedure for retrieving LTM data (see Fig. 4).

4 Simulations

4.1 Experiment Conditions

To examine the performance of the new version of RAN-LTM, we apply it to the approximation of the following one-dimensional function, $g(x)$ (see also Fig. 5):

$$g(x) = 4\exp(-\frac{(x+0.02)^2}{0.005}) + 6\exp(-\frac{(x-0.25)^2}{0.02}) + 3\exp(-\frac{(x-0.8)^2}{0.01}) + 2\exp(-\frac{(x-0.6)^2}{0.005}) \quad (11)$$

Forty-one points are selected at intervals of 0.025 in the input domain, $\{x | 0 \leq x \leq 1\}$, as training data, and different 60 points are selected at random as test data.

For comparative purposes, the performances of RAN and two previous versions of RAN-LTMs are also examined. These two RAN-LTMs are different in the values

1) For every hidden unit, find a LTM datum in which $\tilde{\boldsymbol{x}}_m$ $(m = 1, \cdots, M)$ has the smallest distance to the center vector. Such a LTM datum is noted as $(\tilde{\boldsymbol{x}}_j^*, \tilde{\boldsymbol{z}}_j^*)$ $(j = 1, \cdots, J)$.
2) Calculate the curvature, $C(\tilde{\boldsymbol{x}}_j^*)$ $(j = 1, \cdots, J)$, for the approximated function, $\tilde{f}(\cdot)$, as follows:
$$C(\tilde{\boldsymbol{x}}_j^*) = \tilde{f}(\tilde{\boldsymbol{x}}_j^* + \Delta) - 2\tilde{f}(\tilde{\boldsymbol{x}}_j^*) + \tilde{f}(\tilde{\boldsymbol{x}}_j^* - \Delta),$$
where Δ is a small constant.
3) Calculate outputs of hidden units, y_j, for the pth input, $\boldsymbol{x}_p$.
4) Obtain the following index sets, $S_1 \sim S_4$, for all LTM data, $(\tilde{\boldsymbol{x}}_j^*, \tilde{\boldsymbol{z}}_j^*)$ $(j = 1, \cdots, J)$:
$$\begin{aligned} S_1 &= \{j | y_j > \theta_1, C(\tilde{\boldsymbol{x}}_j^*) > a_1\} \\ S_2 &= \{j | y_j > \theta_2, C(\tilde{\boldsymbol{x}}_j^*) > a_2\} \\ S_3 &= \{j | y_j > \theta_3, C(\tilde{\boldsymbol{x}}_j^*) > a_3\} \\ S_4 &= \{j | y_j > \theta_4, C(\tilde{\boldsymbol{x}}_j^*) > a_4\} \end{aligned}$$
where $\theta_1 > \theta_2 > \theta_3 > \theta_4$ and $a_1 > a_2 > a_3 > a_4$. Here, $\theta_1 \sim \theta_4$ and $a_1 \sim a_4$ are positive constants.
5) Based on the retrieval probability, $P_{S1} \sim P_{S4}$, for $S_1 \sim S_4$, every LTM datum is determined to be recalled. Retrieved LTM data as well as new training data are given to learn connection weights.
6) $p \leftarrow p + 1$ and go to Step 3.

Figure 4: New procedure for retrieving LTM data.

of thresholds, θ_r: $\theta_r = 10^{-2}$ and 10^{-30}. The former RAN-LTM with small θ_r is denoted as RAN-LTM(O$_1$) and the latter is denoted as RAN-LTM(O$_2$). On the other hand, the new version of RAN-LTM is denoted as RAN-LTM(N). The approximation error and learning time are evaluated for these four models. Other parameters are set to the following values:

[Parameters of RAN]
$\varepsilon = 0.0008, \quad \kappa = 0.9, \quad \delta_{max} = \delta_{min} = 0.5$
$\tau = 50, \quad \alpha = 0.0001, \quad \sigma_j^2 = 0.0081,$

[Parameters of LTM]
$\theta_c = 0.97, \quad \rho = 1, \quad \beta = 0.98, \quad \eta = 0.997$
$\theta_1 = 10^{-1}, \quad \theta_2 = 10^{-2}, \quad \theta_3 = 10^{-3}, \quad \theta_4 = 10^{-30}$
$P_{s1} = 0.8, \quad P_{s2} = 0.6, \quad P_{s3} = 0.4, \quad P_{s4} = 0.2.$

$a_1 \sim a_4$ are determined based on the maximum and minimum values of curvatures.

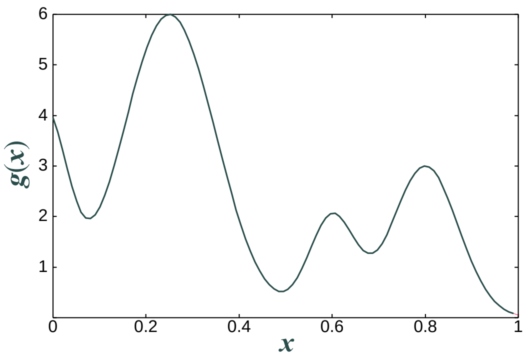

Figure 5: Target function, $g(x)$, to be approximated.

4.2 Experiment 1

The domain of input, x, is divided into two areas, $W_1 : \{x | 0.0 \le x \le 0.5\}$ and $W_2 : \{x | 0.5 < x \le 1.0\}$. After the learning in one area converges, the incremental learning is carried out in the other area. Here, two different experiments are conducted in which the learning is carried out in the following order: $W_1 \to W_2$ and $W_2 \to W_1$. The average approximation error and convergence time are evaluated for the above experiments.

Table 2: Approximation errors (M.S.E.) and convergence time (sec.). In the column of Error, numerals in brackets correspond to errors for training data and the others are errors for test data.

	Error	Time
RAN	0.079 (0.076)	222
RAN-LTM(O$_1$)	0.055 (0.053)	259
RAN-LTM(O$_2$)	0.013 (0.013)	437
RAN-LTM(N)	0.002 (0.002)	261

Table 2 shows the results of Experiment 1. As for RAN, although the convergence is the fastest, the approximation error is the worst. Since the threshold, θ_r, in RAN-LTM(O$_1$) is larger than that in RAN-LTM(O$_2$), LTM data to be retrieved in RAN-LTM(O$_1$) is fewer. Therefore, as shown in Table 2, although the convergence in RAN-LTM(O$_1$) is faster, the approximation error becomes larger. On the other hand, RAN-LTM(N) achieves high approximation ability with comparatively small learning time. Therefore, one can say that incremental learning is conducted efficiency in RAN-LTM(N).

As seen in Table 2, the approximation ability of RAN-LTM(O_2) is inferior to that of RAN-LTM(N) even though most LTM data are always recalled in RAN-LTM(O_2). The reason can be explained from the result shown in Fig. 6. Figure 6 shows the approximation error and the location of LTM data in RAN-LTM(O_2) and RAN-LTM(N). Here, the learning is conducted in the following order: $W_1 \rightarrow W_2$. As you can see in Fig. 6, a LTM datum does not exist in RAN-LTM(O_2) at around $x = 0.3$ where the error due to the interference is large. This result demonstrates that the production of plural LTM data for each hidden unit in RAN-LTM(N) leads to the enhancement of suppressing the interference.

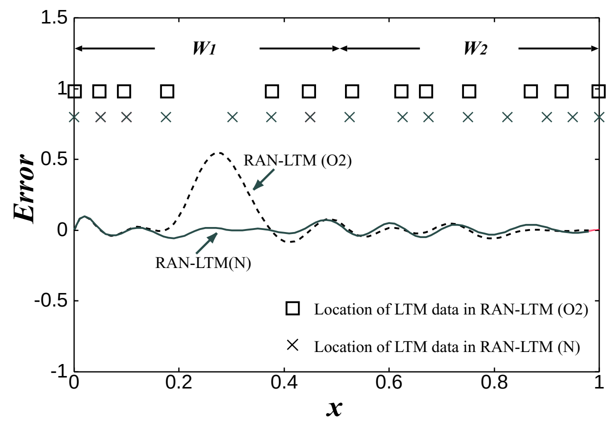

Figure 6: Approximation error (M.S.E.) and location of LTM data after incremental learning.

4.3 Experiment 2

The domain of input, x, is divided into the following four areas, $W_1 : \{x|0.0 \leq x \leq 0.25\}$, $W_2 : \{x|0.25 < x \leq 0.5\}$, $W_3 : \{x|0.5 < x \leq 0.75\}$, $W_4 : \{x|0.75 < x \leq 1.0\}$. After the learning in one area converges, the other three areas are trained sequentially. In Experiment 2, 24 different learning orders can be considered. Therefore, 24 experiments are conducted and the performances of the above four models are evaluated. As you can see in Table 3, RAN-LTM(N) also has excellent approximation ability with comparatively small computations even if incremental learning is conducted repeatedly.

5 Conclusions

In this paper, a new version of RAN-LTM, in which the procedures for producing and retrieving LTM data were improved, was proposed and it was applied to the approximation of a one-dimensional function. We evaluated the approximation ability and the convergence time through the comparison with RAN and the previous version of RAN-LTM. As a result, we certified that the proposed RAN-LTM could suppress the interference more completely with low computation costs.

Table 3: Approximation errors (M.S.E.) and convergence time (sec.). In the column of Error, numerals in brackets correspond to errors for training data and the others are errors for test data.

	Error	Time
RAN	0.070 (0.069)	147
RAN-LTM(O_1)	0.028 (0.027)	1376
RAN-LTM(O_2)	0.004 (0.004)	4245
RAN-LTM(N)	0.005 (0.002)	1547

References

[1] J. Mańdziuk, and L. Shastri: "Incremental class learning - an approach to longlife and scalable learning", *CD-ROM Proc. Int. Joint Conf. on Neural Networks* (1999).

[2] M. Kotani, K. Akazawa, S. Ozawa, and H. Matsumoto: "Detection of leakage sound by using modular neural networks", *Proc. of Sixteenth Congress of the Int. Measurement Confederation*, **IX**, 347/351 (2000).

[3] H. Yamakawa, D. Masumoto, T. Kimoto, and S. Nagata: "Active data selection and subsequent revision for sequential learning" (in Japanese), *Technical Report of IEICE*, **NC92/99** (1993).

[4] K. Yamauchi, N. Yamaguchi, and N. Ishii: "Incremental learning methods with retrieving of interfered patterns", *IEEE Trans. on Neural Networks*, **10**, 6, 1351/1365 (1999).

[5] S. Ozawa, T. Tamaoki, and N. Baba: "Incremental learning for neural networks with long-term memory" (in Japanese), *Proc. of the 27th Intelligent System Symposium*, 173/178 (2000).

[6] J. Platt: "A resource allocating network for function interpolation", *Neural Computation*, **3**, 213/225 (1991).

[7] T. Poggio and F. Girosi: "Networks for approximation and learning", *Proc. IEEE Trans. on Neural Networks*, **78**, 9, 1481/1497 (1990).