Associative Memory for Online Learning in Noisy Environments Using Self-Organizing Incremental Neural Network

Article *in* IEEE Transactions on Neural Networks · May 2009

DOI: 10.1109/TNN.2009.2014374 · Source: PubMed

CITATIONS
44

READS
590

3 authors, including:

Akihito Sudo
Tokyo Institute of Technology
7 PUBLICATIONS **67** CITATIONS

SEE PROFILE

Osamu Hasegawa
SOINN Inc.
180 PUBLICATIONS **2,142** CITATIONS

SEE PROFILE

Associative Memory for Online Learning in Noisy Environments Using Self-Organizing Incremental Neural Network

Akihito Sudo, Akihiro Sato, and Osamu Hasegawa, *Associate Member, IEEE*

***Abstract*—Associative memory operating in a real environment must perform well in online incremental learning and be robust to noisy data because noisy associative patterns are presented sequentially in a real environment. We propose a novel associative memory that satisfies these requirements. Using the proposed method, new associative pairs that are presented sequentially can be learned accurately without forgetting previously learned patterns. The memory size of the proposed method increases adaptively with learning patterns. Therefore, it suffers neither redundancy nor insufficiency of memory size, even in an environment in which the maximum number of associative pairs to be presented is unknown before learning. Noisy inputs in real environments are classifiable into two types: noise-added original patterns and faultily presented random patterns. The proposed method deals with two types of noise. To our knowledge, no conventional associative memory addresses noise of both types. The proposed associative memory performs as a bidirectional one-to-many or many-to-one associative memory and deals not only with bipolar data, but also with real-valued data. Results demonstrate that the proposed method's features are important for application to an intelligent robot operating in a real environment. The originality of our work consists of two points: employing a growing self-organizing network for an associative memory, and discussing what features are necessary for an associative memory for an intelligent robot and proposing an associative memory that satisfies those requirements.**

***Index Terms*—Associative memory, neural network, online learning, robustness to noise.**

I. Introduction

ASSOCIATIVE memory is an important process of human intelligence. We consider it important also for intelligent agents and intelligent robots that can complete complex tasks which only humans have heretofore accomplished with sufficient accuracy. Some authors have proposed using associative memory for intelligent agents [1], [2] or intelligent robots [3], [4]. Desired characteristics of neural associative memories for intelligent robots are expected to have sufficiently large storage capacity and perform well on online incremental learning. In nonstationary environments, agents must learn patterns that are presented sequentially; the number of patterns to be learned cannot be revealed before learning because the surrounding environment changes continuously. However, conventional neural associative memories such as the Hopfield network [5] and bidirectional associative memory (BAM) [6] forget previously learned data incrementally when learning new data. Even when the associative memories learn patterns in a batch manner, all previously stored memory patterns become unstable if the provided patterns are sufficiently numerous. Many studies have been conducted to find methods to avoid this. The strategies investigated in those studies can be classified into two types: improving storage capacity, as described in [7] and [8], and employing a learning rule including a forgetting process, as was examined in [10], and which was analyzed theoretically in [11]. Nevertheless, the problem described above remains unsolved. French has pointed out that distributed representations of patterns cause this instability phenomenon [12]. Neural associative memories such as the Hopfield network and their variants store patterns as distributed data. Therefore, it is an inherent and serious shortcoming. A useful survey of such a forgetting phenomenon of these models has been done [13].

On the other hand, several authors have used competitive learning models for associative memories. Ichiki *et al.* proposed the associative memory model using a Kohonen feature map (KFM) [14]. A model that is an extension of an associative memory using competitive learning was applied to control of dynamic systems [15]. In stark contrast to associative memory models that store knowledge with distributed representation, these which employ competitive learning are much more suited to environments in which patterns are presented sequentially because these models adopt a local representation of knowledge. In these models, every network node can store an associative pair instead of storing it with the whole network; consequently, they suffer less from forgetting previously learned patterns. However, such forgetting can occur even with associative memory using a competitive learning model because the weight of the node that stores a previously learned pattern can be changed when a new training pattern is presented. One study fixed a node that had been trained well and whose weight closely approximated that of a given training pattern and thereby avoided forgetting patterns [16]. Another study proposed lowering the learning rates for nodes near the fixed nodes introduced into that previous study, thereby rendering the associa-

Manuscript received December 03, 2007; revised May 06, 2008 and November 04, 2008; accepted December 02, 2008. First published April 24, 2009; current version published June 03, 2009. This work was supported by the Industrial Technology Research Grant Program in 2004 from the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

A. Sudo and A. Sato are with the Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Yokohama 226-8503, Japan (e-mail: sudo@isl.titech.ac.jp; snowmoon@isl.titech.ac.jp).

O. Hasegawa is with the Imaging Science and Engineering Lab., Tokyo Institute of Technology, Yokohama 226-8503, Japan (e-mail: hasegawa@isl.titech.ac.jp).

Digital Object Identifier 10.1109/TNN.2009.2014374

tive memory as structurally robust [17]. Still another study introduced a reliability parameter to prevent forgetting repeatedly presented patterns [18].

However, those proposed methods are less suited to environments in which the maximum number of patterns to be learned cannot be revealed in advance. They cannot store all the training patterns if a user has not pre-allocated sufficient memory (node size) to the systems. The systems might be able to store all presented patterns, but redundant allocation of nodes might exacerbate memory waste and impart unnecessary computational loads when a user has pre-allocated numerous nodes; holding many nodes is linked directly to the high computational load of competitive learning models because existing node weights are referenced for every input. This kind of memory system is insufficient for intelligent systems that learn various or varying environments autonomously.

As described in this paper, we propose a novel neural associative memory called neural associative memory with self-organizing incremental neural network (SOINN) [19], or SOIAM, which overcomes the limitations of the neural models described in the preceding paragraph. Actually, SOIAM generates a new node representing the presented pattern in its network if and only if no existing node is sufficiently close to the presented pattern when an associative pair is presented. Nodes are added incrementally, depending on the number of presented patterns. Therefore, SOIAM not only suffers less from forgetting previously learned patterns, but also it is well suited to environments in which the maximum number of patterns to be learned is unknown in advance. Because of this property, SOIAM realizes one-to-many and many-to-one associations. Some conventional associative memories are related to many-to-many associations. However, their users must predetermine the number of directions of association by presetting the number of layers. For example, multidirectional associative memory (MAM) [20] holding three layers deals with three-to-three association but cannot process four-to-four association. The use of SOIAM requires no determination of the number of directions in advance.

The proposed method is also robust to noise. Intelligent robots operating in real environments often obtain noisy patterns. The SOIAM robustness to noisy data supports its operation in real environments. Noisy inputs in real environments are classifiable into two types: a noise-added original pattern and a faultily presented random pattern. Examples of the two noisy data types are presented in Fig. 1. Associative memory is expected to interpret the noisy pattern shown in Fig. 1(a) as capital A. The Hamming associative models summarized in [21] can accommodate this type of noise well. The situation is slightly more complex for the noise presented in Fig. 1(b). At the training phase, associative memory will probably not regard the pattern shown in Fig. 1(b) as noise if it is presented many times. Therefore, associative memory must have a function by which only patterns presented for few times are eliminated during the training phase. On the other hand, at the recall phase, associative memory can eliminate such noise if it has the function by which a sufficiently different pattern from those of stored patterns is judged to be an unknown pattern. The SOIAM can process noisy data of both types appropriately. Moreover, SOIAM is more robust to the former type of noise than conventional methods, as described in Section III. To our knowledge, no associative memory deals with noise of the latter type.

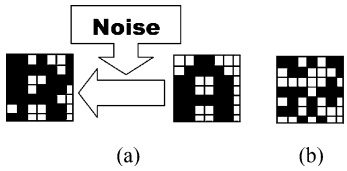


Fig. 1. Noise of two types: noise shown in (a) is added the original pattern representing "A." The noise depicted in (b) is a random pattern that is presented to associative memory.

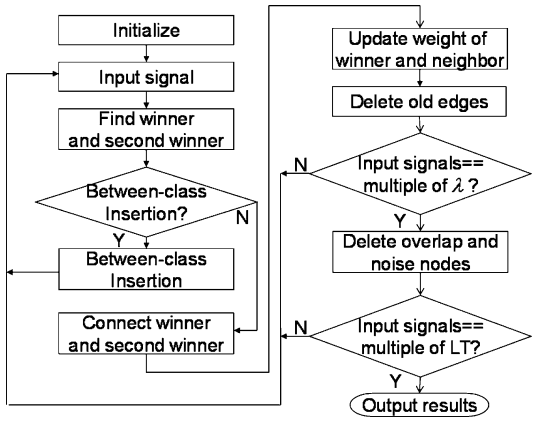


Fig. 2. Flowchart of SOINN.

Many conventional associative memories store and recall binary or bipolar patterns. Several researchers have devised associative memory models for real-valued or gray-scale patterns [8], [9]. The proposed associative memory can store and recall real-valued data precisely. We used gray-scale facial images in the following experiment to illustrate this fact.

II. Proposed Method

A. Overview of SOINN

We describe SOINN [19], which is used as the basis of the proposed method. Initially proposed for data clustering (unsupervised classification) and topology learning, SOINN is a two-layered neural network. The first layer is used to generate a topological structure of presented patterns. The second layer outputs the number of clusters and gives prototype vectors of the distribution of presented patterns. The learning algorithms of both layers are almost identical, but the inputs into them differ. The input to the first layer is the data presented to SOINN. After the first layer finishes learning them, the second layer obtains exactly the same vectors as the weights of the nodes which had been generated in the first layer. The distances are derived both between the input and the nearest node and between the input and the second nearest node when each layer obtains input. A new node with equal weight to that of the input is generated if the distances are sufficiently great. Otherwise, a new edge is generated between the nearest and the second nearest node; the weights of the nearest node and its neighbors are updated. A flowchart of the SOINN algorithm is depicted in Fig. 2.

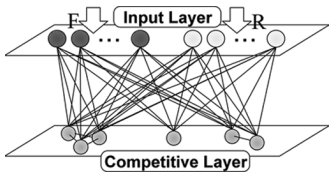


Fig. 3. Structure of the proposed associative memory.

Network growth is an important feature to adapt to nonstationary environments. Many conventional clustering algorithms such as k-means demand that a user predetermine how many clusters are to be generated. For a topology learning problem, a user of many conventional methods such as the KFM must decide the number of nodes in advance. However, SOINN chooses quantities of both clusters and nodes adaptively during learning. Actually, SOINN not only generates new nodes, but also it eliminates unnecessary nodes. This property renders SOINN immune to noise. Some authors have also proposed growing self-organizing networks that remove unnecessary nodes [22]–[27].

B. Extension of SOINN Into Associative Memory

The proposed neural associative memory has an input layer in addition to a competitive layer. Fig. 3 shows the SOIAM structure. At a training phase, the input layer obtains the associative key F and a corresponding output R. Both F and R are combined into one vector X, which is perturbed with additive white Gaussian noise (AWGN) as

$$X = \begin{bmatrix} F \\ R \end{bmatrix} \tag{1}$$

$$I_c = X + n_{\sigma_i} \tag{2}$$

where $F \in \mathbb{R}^M,\ R \in \mathbb{R}^N,\ n_{\sigma_i} \sim N(0, \sigma_i^2)$. The second layer (competitive layer) absorbs I_c and either generates a new node with weight I_c or updates weights of nodes in the second layer. Even when binary data are provided to the input layer, the competitive layer obtains real-valued data because of added perturbation. The presented pattern to SOINN is from the distribution of data; consequently, SOINN can not only generate new nodes, but also it can distinguish them from noise. However, the presented pattern to SOIAM is an associative pair, each of which is just a single pattern, not from a distribution of data. Therefore, the input layer of SOIAM must perturb the presented data.

A path exists between the two nodes if two nodes can be linked with a series of edges. Nodes are considered as belonging to the same cluster when a path exists between them. Then, the competitive layer generates new nodes which have the mean-valued weight of the nodes in the same cluster. This node, called the representative node, plays an important role when SOIAM recalls patterns: it maintains a precise association even in noisy environments. Results also show that several revisions of SOINN improve the SOIAM performance. Although SOINN removes nodes with fewer than three neighbors during every preset period during data learning, SOIAM removes nodes that have one or no neighbor. Furthermore, SOINN removes no node when continuing to add new nodes, but SOIAM removes nodes even then. The two-layer structure lets SOINN produce clusters more appropriately. Consequently, SOINN adopts a two-layered structure. The objective of SOIAM is not data clustering; the single-layer structure is sufficient for the objective. For that reason, we adopt a single-competitive-layer structure for SOIAM.

SOIAM recalls the other part of the pattern if either F or R is presented. The competitive layer obtains I_c generated from F or R when F or R is presented

$$I_c = \begin{cases} \begin{bmatrix} F \\ 0 \end{bmatrix}, & \text{if } F \text{ is presented} \\ \begin{bmatrix} 0 \\ R \end{bmatrix}, & \text{if } R \text{ is presented.} \end{cases} \tag{3}$$

In the competitive layer, the nodes sufficiently close to I_c are sought. If the ith node is sufficiently close to I_c, the recalled pattern O is generated with weight W_ρ of the representative node of the cluster to which the ith node belongs, as

$$O = \begin{cases} \begin{bmatrix} 0 \\ W_\rho^R \end{bmatrix}, & \text{if } F \text{ is presented} \\ \begin{bmatrix} W_\rho^F \\ 0 \end{bmatrix}, & \text{if } R \text{ is presented.} \end{cases} \tag{4}$$

In that equation

$$W_\rho = \begin{bmatrix} W_\rho^F \\ W_\rho^R \end{bmatrix}, \qquad W_i^F \in \mathbb{R}^M, \qquad W_i^R \in \mathbb{R}^N.$$

Multiple patterns are recalled if plural nodes that belong to other clusters are sufficiently close to I_c. Then, SOIAM regards the input as an unknown pattern and returns *unknown* if no node is sufficiently close to I_c.

C. The Complete Algorithm

We present the complete SOIAM algorithm here. The algorithm comprises two parts: one for learning patterns and one for recalling patterns. Fig. 4 shows a flowchart of the learning algorithm. The following notation is used to describe the algorithms.

A	set of generated nodes;
W_i	weight of the ith node
N_i	set of neighbors of the ith node;
n_p	number of presented patterns after last removing nodes;
d_i	similarity threshold of the ith node;
Λ_{edge}	the lifetime of edges;
χi	the number becoming winners of the ith node;
δ_r	similarity threshold at the recall phase;
λ	frequency of node removal;
$\| a - b \|$	Euclidean distance separating a from b.

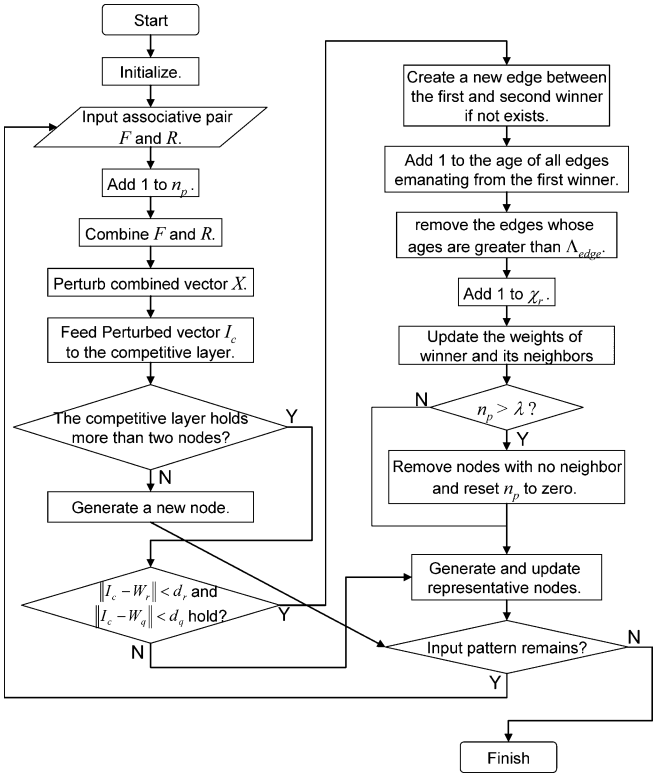


Fig. 4. Flowchart of the learning algorithm.

A node is called a neighbor of the other node when edges connect the two nodes.

Algorithm 1: Learning Associative Pairs

Step 1: Initialize the parameters and initialize A as an empty set; n_p is set to zero.

Step 2: Input a training associative pair F and R, then add 1 to n_p.

Step 3: Combine F and R as (1).

Step 4: Perturb X as (2) and give I_c to the competitive layer.

Step 5: Go to Step 6 if the competitive layer holds more than two nodes. Otherwise, generate a new node whose weight is identical to that of I_c; then go to Step 14.

Step 6: Find the first winner node r whose weight W_r is the nearest to I_c and the second winner node q whose weight W_q is the second nearest to I_c.

Step 7: Verify that

$$||I_c - W_r|| < d_r,\ ||I_c - W_q|| < d_q \tag{5}$$

where

$$d_i = \begin{cases} \max\limits_{k\text{th node} \in N_i} ||W_i - W_k||, & \text{if } N_i \neq \emptyset \\ \min\limits_{k\text{th node} \in A} ||W_i - W_k||, & \text{if } N_i = \emptyset. \end{cases}$$

Go to Step 8, if both inequalities in (5) hold. Otherwise, generate the new node whose weight is the same as that of I_c; then go to Step 13.

Step 8: Create a new edge between r and q if the edge between r and q does not exist.

Step 9: Set the age of the edge between r and q to zero.

Step 10: Add 1 to the age of all edges emanating from r and remove the edges whose ages are greater than Λ_{edge}.

Step 11: Add 1 to χ_r and add ΔW_r and ΔW_i to the weights of r and its neighbors, where

$$\Delta W_r = \frac{1}{\chi_r}(I_c - W_r)$$
$$\Delta W_i = \frac{1}{100\chi_r}(I_c - W_i) \qquad (^{\forall}i\text{th node } \in N_r).$$

Step 12: Remove nodes with no neighbors if n_p is more than λ; reset n_p to zero.

Step 13: Generate or update the representative nodes whose weights are mean values of the respective clusters. They are regarded as being in the same cluster when a path through edges exists between two nodes.

Step 14: Go to Step 2, if a remaining pattern exists to input.

Algorithm 2: Recalling the Associative Pattern

Step 1. Initializing the parameters, and input a pattern K as an associative key.

Step 2. Derive the mean distance $d_{k\leftrightarrow i}$ between K and each node as

$$d_{k\leftrightarrow i} = \begin{cases} \dfrac{\| K - W_i^F \|}{\sqrt{M}}, & \text{if } K \text{ corresponds to } F \\ \dfrac{\| K - W_i^R \|}{\sqrt{N}}, & \text{if } K \text{ corresponds to } R \end{cases}$$

where

$$W_i = \begin{bmatrix} W_i^F \\ W_i^R \end{bmatrix}, \qquad W_i^F \in \mathbb{R}^M, \qquad W_i^R \in \mathbb{R}^N.$$

Step 3. Generate O from the representative node ρ of the cluster to which the ith node belongs as (4) if $d_{k\leftrightarrow i} < \delta_r$.

Step 4. Output all patterns generated in Step 3. Reply unknown pattern if no node satisfies $d_{k\leftrightarrow i} < \delta_r$.

D. Toy Example

Here, we present toy examples of Algorithms 1 and 2, respectively, to demonstrate how SOIAM actually works.

1) Toy Example of Algorithm 1: What SOIAM does from Step 1 through Step 6 is very simple: initializing parameters, receiving input, making a new vector by combining two input vectors, perturbing the new vector, check if more than two nodes exist on the competitive layer, and find the nearest node and the second nearest node of the perturbed vector. Two flows exist after Step 7 depending on whether (5) holds. Therefore, we describe examples of these two cases.

Fig. 5 portrays an example of distribution of nodes on the competitive layer when (5) holds. Then, the edge between the first nearest node and the second nearest node is created to include them in the same cluster (Step 8). Subsequently, setting

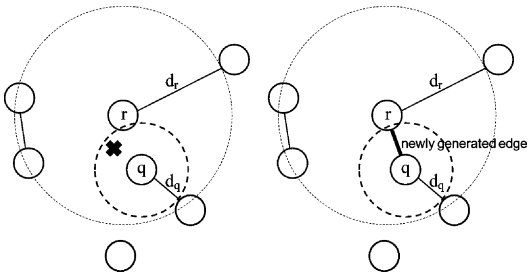


Fig. 5. Example of the competitive layer when (5) holds. Circles represent nodes in the competitive layer; lines between nodes are edges. (a) When the competitive layer obtains the perturbed vector denoted by the cross marks, (b) the new edge is generated between the nearest node and the second nearest node.

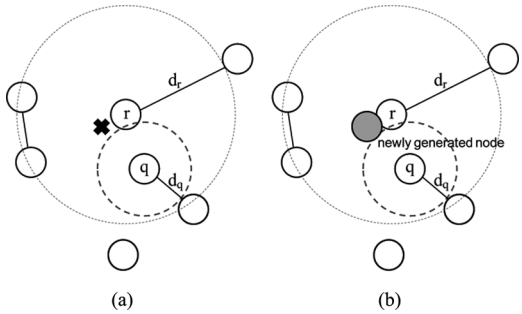


Fig. 6. Example of the competitive layer when (5) does not hold. Circles represent nodes in the competitive layer; lines between nodes are edges. (a) When the competitive layer obtains the perturbed vector denoted by the cross marks, (b) the new node, the weight of which is the same as the perturbed vector, is generated.

the age of edges (Step 9 and 10), removing unnecessary edges (Step 10), moving the first nearest node and nodes connecting with the first nearest nodes (Step 11), removing unnecessary nodes (Step 12), and generating the representative nodes (Step 13) are all executed.

Fig. 6 depicts an example of the distribution of nodes in the competitive layer when (5) does not hold. If (5) does not hold, a new node whose weight is equal to the perturbed vector is generated on the competitive layer; then the representative nodes are generated.

2) Toy Example of Algorithm 2: Figs. 7 and 8 show how SOIAM works when an associative key is respectively presented to the F part and the R part of the input layer. When an associative key is presented to F part, nodes whose weight of F part is sufficiently close to input are searched; the representative nodes' weights of the R part are output.

III. Experiments

This section presents results of several experiments to examine the performance of the proposed method. We compared it with bidirectional associative memory (BAM) with the pseudorelaxation learning algorithm for BAM (PRLAB) [28], associative memory using the Kohonen feature map (KFMAM) [14], and KFMAM with weights fixed and semifixed neurons (KFMAM-FW) [17]. Table I shows that we followed the parameters used in [17] and [28]. The initial weights of all methods were chosen randomly between -1 and $+1$. The SOIAM method learned each training pattern 50 times: 50

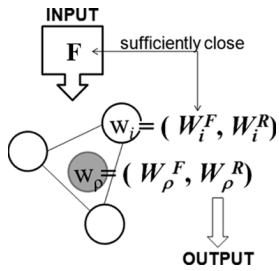


Fig. 7. Toy example of the recalling algorithm when the associative key is provided to the F part of the input layer. White circles represent normal nodes; a gray circle denotes a representative node.

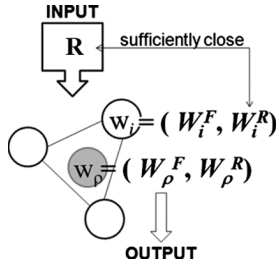


Fig. 8. Toy example of recalling algorithm when an associative key is provided to the R part of the input layer. White circles represent normal nodes; a gray circle denotes a representative node.

TABLE I
Parameter Setting

Method	Values of Parameters
SOIAM	$\Lambda_{edge} = 50, \lambda = 100,$ $\delta_r = 0.15, \sigma_i^2 = 0.02$
BAM with PRLAB	$\lambda = 1.9, \xi = 0.1$
KFMAM	$\alpha_0 = 0.1, \sigma_i = 3.0, \sigma_f = 0.5,$ $T = 2500$
KFMAM-FW	$\alpha_0 = 0.1, \sigma_i = 3.0, \sigma_f = 0.5,$ $T = 2500, d_f = 10^{-3}$

different perturbations were added to vector x; then 50 different inputs were provided to SOIAM in the learning phase. The BAM with PRLAB, and KFMAM were trained until their weights converged. For the experiments described in Sections III-A–III-C, the training patterns shown in Fig. 9 were used. Feature vectors were generated by converting white pixels into $+1$ and black ones into -1. The input layer's number of nodes is 98 because the dimensions of a feature vector are 49.

A. Online Learning Ability

All pairs of capital and corresponding small letters (A, a)–(Z, z) were presented *sequentially* to the associative memories. SOIAM generated 99 nodes in the competitive layer to store (A, a)–(Z, z). In fact, SOIAM required 4 s to learn them on a Xeon Processor (Intel Corporation, Santa Clara, CA) 3.60 GHz with 2-GB RAM. Table II shows that SOIAM was able to recall all presented patterns. However, BAM with PRLAB and KFMAM were not. Fig. 10 shows the recalled patterns of BAM with PRLAB and KFMAM. Both BAM with PRLAB and KFMAM recalled the latter part of the patterns such as "x" and "y" with less error than the former patterns such as "a" and "b," which implies that learning new patterns causes forgetting of previously learned information. Although KFM-FW, which had been allocated a sufficient size of nodes, was also able to

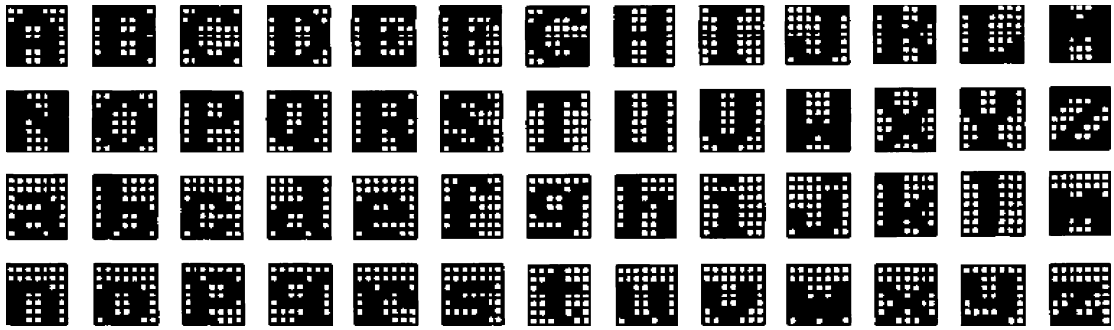
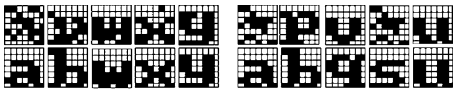


Fig. 9. Training patterns used in the experiments described in Sections III–A–III-C. They are drawn from the IBM personal computer compatible computer graphics adapter (PC CGA) character font.



(a) (b)

Fig. 10. Recalled patterns (upper row) and the desired outputs (bottom row): (a) BAM with PRLAB; (b) KFMAM with 100 nodes.

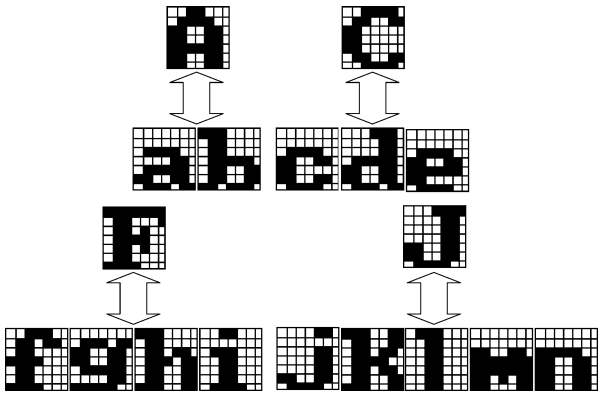


Fig. 11. Training patterns for one-to-many association.

recall all patterns, those with 16 and 25 nodes went into an infinite loop after all nodes had been fixed to store patterns. For that reason, KFMAM-FW might enter an infinite loop in an environment where the maximum number of patterns to be learned is not revealed in advance.

B. One-to-Many Association

We presented SOIAM one-to-many patterns to test whether SOIAM can learn and recall a one-to-many associative pair accurately: (A, a), (A, b), (C, c), (C, d) (C, e), (F, f), (F, g), (F, h), and (F, i) shown in Fig. 11. The directions of association were different among associative keys; no information about the number of directions of association had been given to SOIAM. Nevertheless, SOIAM was able to store and recall all patterns perfectly, which means that the perfect recall rate is 100% in this experiment. It also works as many-to-one associative memory and was able to recall the corresponding capital letter when it was presented with small letters after it learned the patterns. In this experiment, SOIAM generated 81 nodes in the competitive layer to store (A, a)–(F, i). Actually, SOIAM required less than 1 s to learn them on a Xeon Processor (Intel Corporation) of 3.60 GHz with 2-GB RAM.

TABLE II
PERFECT RECALL RATE ON SEQUENTIAL LEARNING

Method	Perfect Recall Rate	Perfectly Recalled Patterns
SOIAM (99 nodes)	100%	a - z
BAM with PRLAB	3.8%	z
KFMAM (64 nodes)	31%	o, p, r, v - z
KFMAM (81 nodes)	38%	h, l, o, p, r, v - z
KFMAM (100 nodes)	42%	d, h, l, o ,p ,r ,v - z
KFMAM-FW (16 nodes)	-	(Going into an infinite loop)
KFMAM-FW (25 nodes)	-	(Going into an infinite loop)
KFMAM-FW (36 nodes)	100%	a–z
KFMAM-FW (64 nodes)	100%	a–z

C. Sensitivity to Noise

We examined sensitivity to the noisy data of two types. The noise shown in Fig. 1(a), binary noise, is noise that randomly flips $+1$ into -1 or -1 into $+1$. It was added to the original pattern and completely random bipolar data were used as the noise-added data shown in Fig. 1(b). In this experiment, KFMAM and KFMAM-FW had each been allocated 100 nodes. Results of KFMAM(-FW) are shown only for the case in which KFMAM(-FW) holds 100 nodes because we confirmed that KFMAM(-FW), with 36, 64, or 81 nodes, is less robust than that with 100 nodes.

1) Noise Adding Into Patterns: After noiseless training data (A, a)–(Z, z) had been learned, patterns generated by adding binary noise to capital letters were presented as associative keys to examine the recall rate from noisy patterns. As noisy associative keys, we generated 2600 noisy patterns, of which 100 patterns corresponded to each capital letter. Fig. 12 shows that when the noise level reaches 20%, the success rate of perfect recall using SOIAM is 91.3%, even though those of the other methods are less than 70%. When the noise level reaches 24%, the perfect recall rate of SOIAM is 85.8%, although those of other methods are less than 49%.

2) Random Patterns as Noise: After noiseless training data (A, a)–(Z, z) had been learned, random patterns were presented

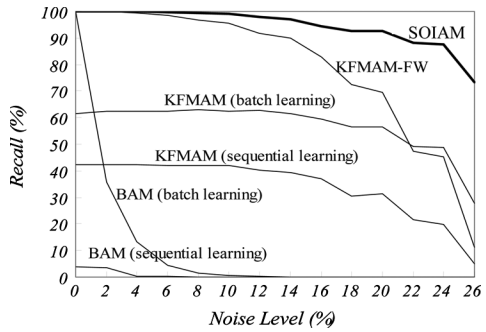


Fig. 12. Perfect recall rate when noisy inputs are provided to SOIAM and conventional methods.

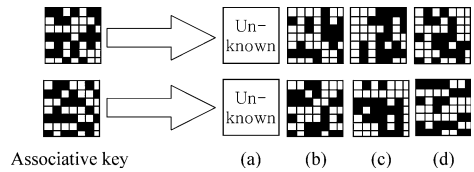


Fig. 13. Recalled patterns from random patterns as noise: (a) SOIAM, (b) BAM with PRLAB on batch learning, (c) KFMAM with 100 nodes, and (d) KFMAM-FW with 100 nodes.

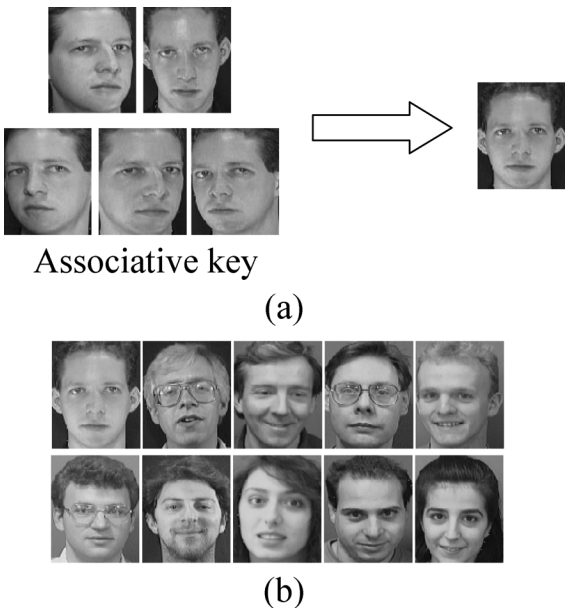


Fig. 14. (a) Example of five-to-one associative pairs, the images of which are gray-scale photos. (b) Example of the images of each person. Actually, 92–112 pixel images are presented to SOIAM.

as associative keys. The associative memories are expected to interpret the patterns as noise and not as correct associative keys. Results are shown in Fig. 13. In fact, SOIAM correctly replied that the associative key is an unknown pattern because the distance between the feature vector of the input pattern and the weight of the nearest node was greater than the threshold. However, BAM with PRLAB and KFMAM-FW output fault patterns because they have no function with which they judge whether a given pattern has already been trained.

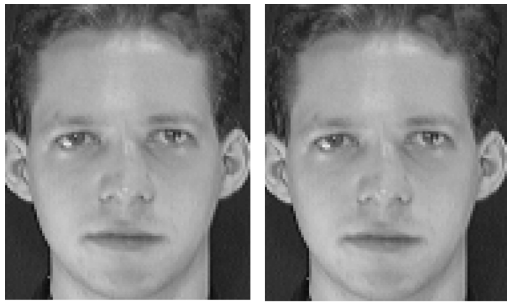


Fig. 15. (a) Original frontal facial image. (b) Recalled image by SOIAM. In this case, the Euclidean distance is 0.126.

Fig. 16. (a) Recalled image from the (b) noisy associative key.

D. Real-Valued Data Storage and Recall

We made SOIAM learn and recall gray-scale facial images to examine whether the proposed method was able to store and recall real-valued data. The AT&T face database was used. The facial images are gray-scale and have 92×112 pixels. The number of nodes in the input layer is 20 608 because the dimensions of a feature vector are 10 304. Before presenting them to SOIAM, the value of each pixel was normalized to the range of $[-1, 1]$. Fig. 14(a) shows that the task here is to associate the facial image from the five images representing the same person, but being taken in different expressions and angles. Here, SOIAM learned 100 associative pairs including 20 different people's facial images. One facial image of each person is shown in Fig. 14(b). These data were presented sequentially as training patterns; then 100 associative keys were presented to SOIAM. Here, SOIAM generated 201 nodes in the competitive layer to store those associative pairs. In fact, SOIAM required 72 min 5 s to learn them on a Xeon Processor (Intel Corporation) of 3.60 GHz with 2-GB RAM. We calculated the Euclidean distance between the original facial image and the recalled image as the recall error to check how precisely SOIAM was able to recall the original image. The recall errors of respective recalled data were almost equal: 0.1–0.2. The pattern dimensionality is 10 304. Therefore, the mean error per pixel was between 1.0×10^{-5} and 2.0×10^{-5}. A typical training example and corresponding recalled images are shown in Fig. 15. This result suggests that SOIAM can learn and recall gray-scale images precisely. A noisy associative key was also presented to SOIAM to check the robustness of the proposed method to a noisy gray-scale image. We used Gaussian noise

to corrupt the original image. The signal-to-noise ratio (SNR) of the image shown on the left-hand side of Fig. 12 is 20. The SNR is defined as

$$\mathrm{SNR} = 10\log_{10}\left(\frac{S}{\sigma}\right)^2 \tag{6}$$

where S is the range of values of each pixel of the image and σ is the deviation of Gaussian noise. Fig. 16 shows that SOIAM was able to execute recall precisely from the noisy associative key.

IV. Conclusion

We proposed a novel neural associative memory SOIAM, which performs well in tests of online incremental learning and realizes both bidirectional and multidirectional association. Results show that SOIAM can learn sequentially presented associative pairs and recall them accurately without forgetting previously learned patterns. Adopting that mode to increment nodes adaptively during learning, SOIAM is well suited to environments in which the maximum number of patterns to be learned cannot be revealed in advance. We conducted an experiment in which SOIAM accommodated two noise types to examine whether SOIAM performs well in noisy environments: noise-added original input and a random pattern were presented faultily as inputs. The proposed associative memory is more robust to noise of both types. Changing noise added to input depending on noise expected in an associative key might improve the noise robustness of SOIAM. That is a subject for our future work. Both bipolar data and real-valued data can be learned accurately by SOIAM. We consider that these features are desirable for applying SOIAM to both an intelligent agent and an intelligent robot. It should store thousands of associative pairs when SOIAM is employed in a large environment. We plan to estimate the size of storage that SOIAM has when we apply it to an intelligent robot or agent.

References

[1] A. Bisler, "An associative memory for autonomous agents," in *Proc. 4th Int. Conf. Intell. Syst. Design Appl.*, 2004, pp. 765–770.

[2] B. J. Rhodes, "Margin notes: Building a contextually aware associative memory," in *Proc. Int. Conf. Intell. User Interface*, 2000, pp. 9–12.

[3] K. Itoh, H. Miwa, Y. Nukariya, M. Zecca, H. Takanobu, P. Dario, and A. Takanishi, "New memory model for humanoid robots—introduction of co-associative memory using mutually coupled chaotic neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2005, pp. 2790–2795.

[4] K. Mizutani and T. Omori, "On-line map formation and path planning for mobile robot by associative memory with controllable attention," in *Proc. Int. Joint Conf. Neural Netw.*, 1999, pp. 2051–2056.

[5] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.

[6] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-18, no. 1, pp. 49–60, Jan./Feb. 1988.

[7] A. F. R. Araujo and G. M. Haga, "Two simple strategies to improve bidirectional associative memory's performance: Unlearning and delta rule," in *Proc. Int. Conf. Neural Netw.*, 1997, vol. 2, pp. 1178–1182.

[8] P. Sussner and M. E. Valle, "Gray-scale morphological associative memories," *IEEE Trans. Neural Netw.*, vol. 17, no. 3, pp. 559–570, May 2006.

[9] B.-L. Zhang, H. Zhang, and S. S. Ge, "Face recognition by applying wavelet subband representation and kernel associative memory," *IEEE Trans. Neural Netw.*, vol. 15, no. 1, pp. 166–177, Jan. 2004.

[10] M. Mézard, J. P. Nadal, and G. Toulouse, "Solvable models of working memories," *J. Physique*, vol. 47, pp. 1457–1462.

[11] T. Kimoto and M. Okada, "Sparsely encoded associative memory model with forgetting process," *IEICE Trans. Inf. Syst. (Inst. Electron. Inf. Commun. Eng.)*, vol. E85-D, no. 12, pp. 1938–1945, 2002.

[12] R. M. French, "Using semi-distributed representation to overcome catastrophic forgetting in connectionist networks," in *Proc. 13th Annu. Cogn. Sci. Soc. Conf.*, 1991, pp. 173–178.

[13] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends Cogn. Sci.*, vol. 3, no. 4, pp. 128–135, 1999.

[14] H. Ichiki, M. Hagiwara, and M. Nakagawa, "Kohonen feature maps as a supervised learning machine," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 1944–1948.

[15] G. Barreto and A. Araujo, "Identification and control of dynamical systems using the self-organizing map," *IEEE Trans. Neural Netw.*, vol. 15, no. 5, pp. 1244–1259, Sep. 2004.

[16] S. Kondo, R. Futami, and N. Hoshimiya, "Pattern recognition with feature map and the improvement on sequential learning ability," *IEICE Tech. Rep.*, vol. 95, no. 599, pp. 55–60, 1996.

[17] T. Yamada, M. Hattori, M. Morisawa, and H. Ito, "Sequential learning for associative memory using Kohonen feature map," in *Proc. Int. Joint Conf. Neural Netw.*, 1999, pp. 1920–1923.

[18] S. Yoshizawa, S. Doki, and S. Okuma, "A new associative memory system for supplemental learning under restriction on memory capacity," *IEICE Trans. Inf. Syst.*, vol. J82-D, no. II(6), pp. 1072–1081, 1999.

[19] F. Shen and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Netw.*, vol. 19, no. 1, pp. 90–106, 2006.

[20] M. Hagiwara, "Multidirectional associative memory," in *Proc. Int. Joint Conf. Neural Netw.*, 1990, pp. 3–6.

[21] P. Watta and M. H. Hassoun, "Generalizations of the Hamming associative memory," *Neural Process. Lett.*, vol. 13, pp. 183–194.

[22] I. Valovaa, D. Szera, N. Gueorguievab, and A. Buer, "A parallel growing architecture for self-organizing maps with unsupervised learning," *Neurocomputing*, vol. 68, pp. 177–195, 2005.

[23] S. Marsland, J. Shapiro, and U. Nehmzow, "A self-organizing network that grows when required," *Neural Netw.*, vol. 15, no. 8–9, pp. 1041–1058, 2002.

[24] B. Fritzke, "Growing cell structures—a self-organizing network for unsupervised and supervised learning," *Neural Netw.*, vol. 7, no. 9, pp. 1441–1460, 1994.

[25] B. Fritzke, "A self-organizing network that can follow non-stationary distributions," in *Proc. Int. Conf. Artif. Neural Netw.*, 1997, pp. 613–618.

[26] J. Jockusch and H. Ritter, "An instantaneous topological mapping model for correlated stimuli," in *Proc. Int. Joint Conf. Neural Netw.*, 1999, pp. 529–534.

[27] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems (NIPS)*. Cambridge, MA: MIT Press, 1995, pp. 625–632.

[28] H. Oh and S. C. Kothari, "Adaptation of the relaxation method for learning in bidirectional associative memory," *IEEE Trans. Neural Netw.*, vol. 5, no. 4, pp. 576–583, Jul. 1994.

Akihito Sudo received the B.S. degree in physics and the M.E. degree in pure and applied physics from Waseda University, Tokyo, Japan, in 2002 and 2005, respectively, and the Dr. Eng. degree in computational intelligence and systems science from Tokyo Institute of Technology, Tokyo, Japan, in 2008.

Akihiro Sato received the B.S. degree in engineering from the University of Electro-Communications, Chofu, Japan, in 2006 and the M.E. degree in computational intelligence and systems science from Tokyo Institute of Technology, Tokyo, Japan, in 2008.

Osamu Hasegawa received the Dr. Eng. degree in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1993.

He was a Research Scientist with the Electrotechnical Laboratory from 1993 to 1999 and with the National Institute of Advanced Industrial Science and Technology, Tokyo, from 2000 to 2002. From 1999 to 2000, he was a Visiting Scientist with the Robotics Institute, Carnegie Mellon University, Pittsburgh PA. In 2002, he became a Faculty Member with the Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, Yokohama, Japan. In 2002, he was jointly appointed as a Researcher at PRESTO, Japan Science and Technology Agency.

Dr. Hasegawa is a member of the IEEE Computer Society, Institute of Electronics, Information and Communication Engineers, and Information Processing Society of Japan.