INVITED ARTICLE

Memory and Learning of Sequential Patterns by Nonmonotone Neural Networks

Masahiko Morita

University of Tsukuba

Abstract—Conventional neural network models for temporal association generally do not work well in the absence of synchronizing neurons. This is because their dynamical properties are fundamentally not suitable for storing sequential patterns, no matter what storage or learning algorithm is used. The present article describes a nonmonotone neural network (NNN) model in which sequential patterns are stored by being embedded in a trajectory attractor of the dynamical system, and recalled stably and smoothly without synchronization; recall is done in such a way that the network state successively moves along the trajectory. A simple and natural learning algorithm for the NNN is also presented, where one only has to vary the input pattern gradually and modify the synaptic weights according to a kind of covariance rule; then the network state follows slightly behind the input pattern, and its trajectory grows to be an attractor with a small number of repetitions.

Keywords—Sequential pattern memory, Temporal association, Nonmonotone neural networks, Nonmonotone dynamics, Trajectory attractors, Spatiotemporal pattern learning, Covariance rule.

This article is to be published in the 1996 special issue of *Neural Networks*. Requests for reprints should be sent to Masahiko Morita, Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba, Ibaraki 305, Japan.

1 Introduction

In associative memory of static patterns, the basic behavior of a network hardly depends on whether the network dynamics is discrete or continuous, synchronous or asynchronous in updating the states of neurons. In contrast, when sequential patterns are stored, conventional networks of a fully recurrent type do not work well unless discrete synchronous dynamics (or an equivalent synchronizing mechanism) is used. This implies that the network cannot recall the pattern sequence in such a way that the state of the network changes gradually from one stored pattern to another, which is unnatural as a model of memory; it also restricts application of neural networks to temporal information processing.

These problems are thought to originate from a fundamental dynamical property that virtually only point-type attractors can be embedded in conventional networks. That is, since strong attractors have to be isolated from each other, the network state must jump a distance to move from one attractor to another, or neurons must change their state synchronously. This dynamical property is hardly changed by improving the storage or learning algorithm. It is therefore necessary to improve the network dynamics, and one of the most effective ways to do so is to use a nonmonotone neural network (NNN).

The NNNs are recurrent networks consisting of neurons whose input-output characteristics are nonmonotonic. They are broadly divided into two models, discrete and analog (continuous). The analog NNN model, first suggested by Morita, Yoshizawa, and Nakano (1990), differs from the so-called analog Hopfield model (Hopfield, 1984) only in that the output function (also called activation, gain, or transfer function) of neurons is nonmonotonic as shown in Figure 1. We will deal with this model in the present article.

As previously reported (Morita, 1993), the NNN exhibits good performance in terms of associative memory, that is, memory capacity (the number of patterns that can be stored) and retrieval ability (the basin of attraction) are greatly enhanced. Moreover, there is little, if any, spurious memory; when the network fails in correct retrieval, it is driven into a chaotic state of behavior.

These features have recently been studied theoretically by many researchers. For example, Yoshizawa, Morita, and Amari (1993) analyzed an analog NNN model based on a geometrical argument; they showed that the memory capacity can be as large as about

$0.4n$ (n is the number of neurons), which is much larger than that of the Hopfield model. Shiino and Fukai (1993) developed the self-consistent signal-to-noise analysis and applied it to NNN models with various output functions to examine their memory capacity (see also Fukai, Kim, & Shiino, 1995). The possible capacity for optimal synaptic weights was studied by Boffetta, Monasson, and Zecchina (1993) using the replica symmetry breaking approximations (see also Kobayashi, 1991). Nishimori and Opris (1993) investigated the retrieval process based on the method of statistical neurodynamics; a more rigid theory was developed by Okada (1995). For the discrete model, analysis was made by Yanai and Amari (1996).

However, the NNN has another important property which is not treated in these studies. That is, it can make good use of the memory structure (i.e., organized distribution of stored patterns) owing to a fundamental change in its dynamical properties. This will appear most clearly in the case of temporal association, where sequential patterns are stored with a string-shaped structure in the state space of the network.

Indeed, the NNN can have attractors of the string type, namely, trajectory attractors; thus it can memorize sequential patterns in a natural way and with a large capacity, and recall a sequence stably and smoothly (Morita, 1994). We will examine these properties and how they are realized.

2 Conventional Models

Before describing the NNN, let us review basic models of temporal association and why they require synchronization.

2.1 Cross-Correlation Matrix Memory

As a starting point of our discussion, we will consider the simplest model (Amari, 1972) where the output of neurons is ± 1 and all the neurons act synchronously. Specifically, each neuron calculates its output at discrete times $t = 0, 1, 2, \ldots$ according to

$$x_i(t+1) = \mathrm{sgn}\Big(\sum_{j=1}^{n} w_{ij} x_j(t)\Big), \tag{1}$$

where x_i is the output of the i-th neuron, w_{ij} is the synaptic weight from the j-th neuron to the i-th one, and $\mathrm{sgn}(u) = 1$ for $u > 0$ and -1 for $u \leq 0$. The current state of the

network is represented by a column vector $X = (x_1(t), \ldots, x_n(t))^T$ (superscript T denotes the transpose). Using the weight matrix $W = [w_{ij}]$, we may write eqn (1) as

$$X(t+1) = \mathrm{sgn}(WX(t)), \tag{2}$$

where the function sgn operates componentwise on vectors.

Assume m pattern vectors $S^0, S^1, \ldots, S^{m-1}$ ($S^\mu = (s_1^\mu, \ldots, s_n^\mu)^T$, $s_i^\mu = \pm 1$) to be orthogonal to each other, or ${S^\mu}^T S^\nu = 0$ for all $\mu \neq \nu$. Then a sequence $S^0 \to S^1 \to \cdots \to S^{m-1}$ is stored in the network by setting

$$W = \frac{1}{n} \sum_{\mu=0}^{m-2} S^{\mu+1} {S^\mu}^T. \tag{3}$$

Since w_{ij} is proportional to the correlation between $s_i^{\mu+1}$ and s_j^μ, this model is a kind of cross-correlation matrix memory.

When S^0 or a similar pattern is given as an initial state $X(0)$,

$$X(1) = \mathrm{sgn}\Big(\frac{1}{n} \sum_{\mu=0}^{m-2} ({S^\mu}^T X(0)) S^{\mu+1}\Big) = S^1, \tag{4}$$

that is, S^1 is recalled at $t = 1$. In the same way, $X(2) = S^2$, $X(3) = S^3, \cdots$, or the stored sequence is recalled.

Let us regard the network as a dynamical system and consider its state space. The space consists of 2^n points representing the possible states of the network, where the current state X moves from point to point. Patterns S^μ are represented by m mutually distant points.

Figure 2 shows the distribution of flow vectors in the state space schematically. Flow at S^μ ($\mu = 0, \ldots, m-1$), which is nearly parallel with the vector $V(S^\mu) \equiv WS^\mu$, points to $S^{\mu+1}$ since $WS^\mu = S^{\mu+1}$. At a state which lies in between S^μ and $S^{\mu+1}$, however, the flow points not to $S^{\mu+1}$ but between $S^{\mu+1}$ and $S^{\mu+2}$.

If the network dynamics is given by eqn (2), the network state X jumps a long distance along the flow at S^μ to reach the next pattern. On the other hand, if the dynamics is not synchronous, X moves step by step because only a few neurons can change their state at a time. As a result, although X starts from S^0 toward S^1, it gradually varies its direction of movement and departs from the sequence, and thus recall fails.

It might be possible that X reaches S^1 without synchronization if S^0 and S^1 are close enough and few states lie in between; however, such a situation does not satisfy the as-

sumption that S^μ are mutually orthogonal and thus only a very short sequence can be stored.

Sompolinsky and Kanter (1986) and Kleinfeld (1986) proposed a model that acts with continuous time. In their model, however, "fast synapses" are introduced to the above network, which work as a synchronizing mechanism. That is, X is attracted by S^μ by the function of fast synapses, then forced to move near $S^{\mu+1}$ within a short time by the function of "slow synapses" with a large delay, attracted by $S^{\mu+1}$ again, and so on. Accordingly, recall is not very smooth; besides, it does not work well when such an initial state that lies midway between S^μ and $S^{\mu+1}$ is given.

Baram (1994) proposed another type of model using internal neurons. Although his model does not explicitly employ synchronization, synapses with different delays are also required; besides, it uses an extremely large number of neurons increasing exponentially with the dimensions of the input pattern.

2.2 Temporal learning

As described above, synchronization is essential to the cross-correlation type of temporal association model. One might think that this is due to the improper weight matrix and that synchronization will not be necessary if the pattern sequence is stored using a proper learning algorithm such as the recurrent back-propagation (BP) algorithm (e.g., Pineda, 1987; Pearlmutter, 1989; Doya & Yoshizawa, 1989). Actually however, as long as the network dynamics is time-continuous, learning is not achieved unless the size of the network is small enough and very limited patterns are learned.

The reason for this is that the BP learning only attempts to minimize the difference (mean-square errors) between the trajectory of the network state and the given orbit. For stable recall of the target sequence, it is necessary that the learned trajectory becomes an attractor of the system, but this is not guaranteed by the BP algorithm; in fact, it is hardly possible for the following reason.

Let us first consider the case that a static pattern, say S^0, has been stored. Then in the state space, S^0 is a point attractor around which flow is toward it. Such a flow distribution is naturally achieved by directing the vector $V(S^0)$ to S^0, since generally $V(S) \simeq V(S^0)$ if $S \simeq S^0$. If the energy function of the network is defined, S^0 is located

at a local minimum of the energy. It should be noted that the energy function is regarded as sharply pointed at its minimum (Figure 3) since the network state X is attracted more strongly as it approaches S^0 (in this respect, the NNN is different; see Section 4.2).

In contrast, if a spatiotemporal pattern gradually changing from S^0 to another pattern S^1 has been successfully stored, the corresponding trajectory in the state space should be a string-type attractor with gentle flow toward S^1. Intuitively, the energy function takes the form of a gutter whose bottom is the trajectory (see Figure 12). Formation of such a flow distribution, however, is very difficult, since flow differs considerably between two adjacent points on and off the trajectory, that is, vector $V = WX$ has to vary greatly depending on only a few specific components of X. In order to realize such flow everywhere along the trajectory, many hidden neurons are required, the number of which is thought to grow increasingly (possibly exponentially) with the dimensions of the input pattern. This also causes an explosive increase in learning time, computational cost and local minima of learning.

These problems do not depend on the learning algorithm but are generic in the neural networks with conventional dynamics, and thus are quite difficult to solve without improving the network dynamics.

3 Nonmonotone Neural Networks

3.1 Network Dynamics

The dynamics of the analog NNN is expressed by

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} w_{ij} y_j, \tag{5}$$

$$y_i = f(u_i), \tag{6}$$

where u_i denotes the instantaneous potential of the i-th neuron, y_i the output and τ a time constant; $f(u)$ is the output function shown in Figure 1. In mathematical terms,

$$f(u) = \frac{1 - e^{-c_1 u}}{1 + e^{-c_1 u}} \cdot \frac{1 + \kappa e^{c_2(|u|-h)}}{1 + e^{c_2(|u|-h)}}, \tag{7}$$

where c_1, c_2 and h are positive constants and κ is a parameter which is usually negative. Note that the formulas (5)–(7) for this dynamics (nonmonotone dynamics) are the same as those for conventional monotone dynamics used in the analog Hopfield model provided

that $\kappa = 1$. In the computer simulations described later, $c_1 = 50$, $c_2 = 10$, $h = 0.5$ and $\kappa = -1$ are used, but these parameter values are not very critical; the most essential point is the nonmonotonic property of the output function.

In this model, $x_i = \mathrm{sgn}(u_i)$ is assumed to be observable, and the result of retrieval is given by $X = (x_1, \ldots, x_n)^T$. We will deal mainly with the discrete state space and call X the network state as before, though it does not uniquely specify the state of the system.

3.2 Storage

Let $Q^\nu = (q_1^\nu, \ldots, q_n^\nu)^T$ $(\nu = 0, \ldots, m-1)$ be n-dimensional patterns whose elements are ± 1. For convenience, we deal with a cyclic sequence

$$\{Q^\nu\}_{\nu=0}^{m-1} \equiv Q^0 \to Q^1 \to \cdots \to Q^{m-1} \to Q^0 \to Q^1 \to \cdots,$$

where m is large enough and the cyclic property is not critical. We may choose Q^ν arbitrarily unless Q^{ν_1} and Q^{ν_2} for $|\nu_1 - \nu_2| > 1$ are very similar. It is desirable that adjacent patterns Q^ν and $Q^{\nu+1}$ have a proper overlap; otherwise, we need to interpolate some patterns between them so that nonmonotone dynamics can make full use of the memory structure.

Here we assume that Q^ν are randomly selected out of 2^n possible patterns and thus distributed uniformly in the state space. In order to achieve smooth recall using a simple synaptic weight matrix of the cross-correlation type, we construct a sequence $\{S^\mu\}_{\mu=0}^{lm-1}$ by interpolating $l-1$ patterns between Q^ν and $Q^{\nu+1}$ for each $\nu = 0, \ldots, m-1$ (we put $Q^m = Q^0$), as schematically shown in Figure 4. Concretely, we let $S^{l\nu} = Q^\nu$ and obtain $S^{l\nu+\xi}$ $(\xi = 1, \ldots, l-1)$ by flipping the last ξ/l of the elements q_i^ν such that $q_i^\nu \neq q_i^{\nu+1}$.

Then we determine the synaptic weights by

$$W = \frac{1}{n} \sum_{\mu=0}^{lm-1} \frac{1}{l} S^{\mu+1} S^{\mu T}, \tag{8}$$

where $S^l = S^0$, and the term $1/l$ is put after $\sum$ for the case that the number $l-1$ of interpolation patterns is varied with ν. We can store more than one sequence in the same way.

3.3 Computer Simulation

To examine the behavior of the network during recall, computer simulations were performed. A sequence of $m = 100$ patterns was stored in a network of $n = 1000$ neurons according to the above procedure. Three patterns ($l = 4$) were interpolated between the original patterns. We define the overlap p_μ between X and S^μ by

$$p_\mu = \frac{1}{n} \sum_{i=1}^{n} x_i s_i^\mu, \tag{9}$$

where $p_\mu = 1$ implies that $X = S^\mu$ and S^μ is recalled.

First of all, conventional dynamics ($\kappa = 1$) was applied to this network. Figure 5 shows the result. In this figure, the overlaps p_μ ($\mu = 0, \ldots, 16$) are plotted against time.

Since $X = S^0$ is given as an initial state, the initial values of p_0, p_1, p_2 and p_3 are 1, 0.75, 0.5 and 0.25, respectively, and the others are about 0. We see that the peak value of p_μ decreases with an increase of μ and that p_1 to p_7 converge at about 0.3. This means that the network state X starts to move along the stored sequence but gradually departs from it, reaches an intermediate state among S^1 to S^7, and finally almost stops there.

On the other hand, the NNN could successfully recall the sequence as shown in Figure 6, where the initial state was so given that $p_0 = 0.3$ and $p_4 \simeq 0$. Although p_0 does not increase to more than 0.67, p_1 and p_2 increase up to 0.85 and 0.95, respectively, and p_μ for $\mu \geq 3$ successively take their peak value of about 1. In this process, $p_{l\nu}$ (thick lines) increase gradually from a small value to the peak while $p_{l(\nu-1)}$ decrease from about 1 to 0, which indicates that X moves continuously from $Q^{\nu-1}$ to Q^ν. The network continued to recall the sequence and recalled S^0 at $t \simeq 300\tau$, that is, the sequence was recalled with a period of about 300τ.

Figure 7 shows the retrieval process in a different way, where trajectories of X are projected onto a p_4-p_0 plane. Patterns S^0 ($= Q^0$) and S^4 ($= Q^1$) are located at $(\varepsilon, 1)$ and $(1, \varepsilon)$, respectively, where $\varepsilon = {Q^0}^T Q^1/n \simeq 0$, and S^1, S^2 and S^3 lie on the straight line between the two points at regular intervals.

We can see from this figure that X approaches the pattern sequence and then moves toward Q^1 along the line $p_0 + p_4 = 1 + \varepsilon$. Together with Figure 6, this indicates that near the orbit connecting Q^0 and Q^1, flow is gentle and nearly parallel to it, whereas at a distance from the orbit, flow is rapid and nearly perpendicular to it. We may therefore say

that the orbit along the sequence is a string-type attractor with a large basin of attraction and that recall is highly tolerant to noise (note that random noise moves X almost always in the perpendicular direction to the orbit in the state space).

Retrieval fails when the initial overlap is too small and the initial state is outside the basin of attraction, as is the case with associative memory of static patterns. This model does not work also when too long a sequence or too many sequences are stored.

Memory capacity of this model depends on many parameters and is not clear. It seems, however, that the capacity is mainly determined by the total amount of information contained in the sequence, and that the most characteristic parameter except κ (the capacity is null if $\kappa \simeq 1$) is the number m of independent patterns in the sequence. It should be noted that even if l or the number of interpolation patterns increases, the amount of information does not increase very much. For reference, the maximum value of m for which the sequence can be retrieved was about $0.25n$ in the above experimental condition.

Figure 8 shows the behavior of the network when the number of interpolation patterns is increased ($l = 8$). We see that an almost correct sequence is retrieved though the peak values of p_μ are slightly smaller than those in Figure 6. We also see that the increase and decrease in p_μ are more linear and less steep (note that the time scale is different), which means that X moves more smoothly and slowly with increasing l. Accordingly, we can regulate the recall speed to some extent by changing the method of interpolation. Regulation can be done more easily if we vary the synaptic weights w_{ii} of self-connections (X moves fast when w_{ii} are small and slowly when they are large).

Incidentally, one might want to discriminate the original patterns Q^μ from the interpolation patterns. To do so, the weight matrix given by eqn (8) should be replaced by some other matrix, for example, by

$$W = \frac{a}{n} \sum_{\mu=0}^{lm-1} \frac{1}{l} S^{\mu+1} S^{\mu T} + \frac{1-a}{n} \sum_{\nu=0}^{m-1} Q^\nu Q^{\nu T}, \tag{10}$$

where a is a parameter ($0.5 < a < 1$). Then in recall, X stays at Q^ν (u_i may vary but their signs do not change) for a while and thus Q^ν can be discriminated, although recall becomes less smooth as a decreases and the period of stay increases. The simulation result for $l = 4$ and $a = 0.52$ is shown in Figure 9.

3.4 Dynamical Structure

It is often the case that the network state X does not exactly pass through S^μ (i.e., $p_\mu \neq 1$) but passes by it, especially when m and l are large. Even in such a case, X moves exactly on the line connecting Q^0 and Q^1 in the p_l-p_0 graph. Together with the above results, this fact implies that the model has such a dynamical structure as shown in Figure 10 (we take the case of $l = 4$).

In this figure, the curved surface Ω represents a subspace consisting of the states on the line segment between S^0 and S^4 in Figure 7. If and only if X is on this surface (i.e., $X \in \Omega$), $x_i = q_i^0 = q_i^1$ holds for all i such that $q_i^0 = q_i^1$ is satisfied; thus Ω has about $2^{0.5n}$ states. Off the surface Ω, flow is nearly perpendicular to (X is strongly attracted by) Ω. However, flow becomes gentle as X approaches Ω, and on Ω, it is nearly parallel with (X is not attracted very much by) the orbit via S^μ.

We can assume a similar surface (Ω-subspace) between S^μ and $S^{\mu+l}$ which we denote by $\Omega(S^\mu, S^{\mu+l})$. Neighboring Ω-subspaces overlap with each other; for example, the states on the orbit between S^3 and S^4 are included in $\Omega(S^0, S^4)$, $\Omega(S^1, S^5)$, $\Omega(S^2, S^6)$ and $\Omega(S^3, S^7)$. The network recalls the sequence stably because X is always attracted by and included in a set of several Ω-subspaces (Figure 11).

Why does the NNN have such a dynamical structure? To simplify the discussion, we suppose without loss of generality that

$$Q^0 = (\underbrace{1, 1, \ldots, 1}_{n}) \text{ and } Q^1 = (\underbrace{1, \ldots, 1}_{k}, \underbrace{-1, \ldots, -1}_{n-k}),$$

where $k \simeq n/2$. Then s_i^ξ $(0 \leq \xi \leq 4)$ are 1 if $i \leq k + (n-k)\xi/4$ and -1 otherwise. Also, $x_i = 1$ applies to all $i \leq k$ if and only if $X \in \Omega(S^0, S^4)$.

Assume that X is at a state given by adding random noise to S^2 and thus X is outside $\Omega(S^0, S^4)$ and other Ω-subspaces. Then the vector $V = WX$ (used in conventional dynamics) is given by

$$\begin{aligned} V &= \frac{1}{4n} \sum_{\mu=0}^{l-1} (X^T S^\mu)\, S^{\mu+1} \\ &= \frac{1}{4} \sum_{\mu=0}^{l-1} p_\mu S^{\mu+1} \\ &\simeq \frac{p_2}{16}(S^0 + 2S^1 + 3S^2 + 4S^3 + 3S^4 + 2S^5 + S^6). \end{aligned} \tag{11}$$

Vector $V' \equiv WY$ (used in nonmonotone dynamics) does not differ much from V provided that $|u_i| < h$ for almost all i; this condition is satisfied when p_2 is small.

Since $s_j^0 = \cdots = s_j^4 = 1$ for $j \leq k$, v_j and v'_j for $j \leq k$ take a large value, that is, x_j tends to be 1. In this way, X is strongly attracted toward $\Omega(S^0, S^4)$ by, as it were, cooperation of patterns S^ξ $(0 \leq \xi \leq 4)$ when X is outside the Ω-subspace. This applies, of course, to conventional as well as nonmonotone dynamics.

The situation becomes different, however, as X approaches $\Omega(S^0, S^4)$ and p_2 increases. In conventional dynamics, the direction of V does not change much and thus X moves not along the sequence but toward an intermediate state among S^ξ as shown in Figure 5 (see also Figure 11). On the other hand, in nonmonotone dynamics, u_j for all $j \leq k$ tend to be large so that $y_j = f(u_j)$ become nearly equal to 0, which reduces considerably the ratio of $Y^T S^\xi$ $(\xi = 0, 1, 3, 4)$ to $Y^T S^2$ together with the size (norm) of the vector Y. As a result, V' is decreased in size and directed to S^3, that is, a gentle flow from S^2 to S^3 is generated.

This explanation may be rather intuitive, but it is confirmed by examining the behavior of each neuron in the simulations.

4 Learning Spatiotemporal Patterns

Temporal association by the NNN has a great advantage in that it does not require synchronization of neurons or special synaptic delay in recall. In storage, however, it is necessary to preserve previous patterns somewhere if we use the algorithm described in Section 3.2; besides, the algorithm is not applicable (a more complicated weight matrix should be used) in cases such that spatiotemporal patterns varying successively (i.e., one bit changes at a time) are given.

In this section, we will introduce a learning algorithm to the NNN to solve these problems. Unlike conventional algorithms of the back-propagation type, this algorithm is quite simple, being based on the covariance learning rule.

4.1 Algorithm

Network dynamics for learning is basically the same as before but an external signal z_i is input to each neuron. Specifically, we replace eqn (5) by

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} w_{ij} y_j + z_i. \tag{12}$$

In parallel with this dynamics, covariance learning is performed using z_i as the learning signal. That is, synaptic weights are modified according to

$$\tau' \frac{dw_{ij}}{dt} = -w_{ij} + \alpha z_i y_j, \tag{13}$$

where α denotes a learning coefficient ($\alpha > 0$) and τ' is a time constant of learning ($\tau' \gg \tau$).

The external input vector $Z = (z_1, \ldots, z_n)^T$ is generally a function of the spatiotemporal pattern $S(t)$ to be stored, and determining this function is the problem of encoding. Here we deal with the simplest case

$$Z(t) = \beta S(t), \tag{14}$$

where $S = (s_1, \ldots, s_n)^T$, $s_i = \pm 1$ and β is a positive coefficient representing the input intensity of the learning signal. In this case, the learning rule can be written as

$$\tau' \frac{dw_{ij}}{dt} = -w_{ij} + \alpha \beta s_i y_j, \tag{15}$$

which is regarded as the covariance rule between the input pattern S and the output vector Y.

4.2 Learning Process

The process of learning is schematically shown in Figure 12. In this figure, the "energy landscape" of the network is depicted with a dot and an arrow representing the current network state X and input pattern S, respectively. Although actually the energy function cannot be defined in the NNN, it is useful for our intuitive understanding to suppose a landscape such that X goes down the hill.

First, assume that a static pattern S^0 has been input, or $S = S^0$ is maintained, for a while. Soon $X = S$ owing to the external input $Z = \beta S$; in the meantime, the energy

around S^0 decreases through learning and thus S^0 becomes a point attractor of the system (Figure 12a). It should be noted that unlike Figure 3, the energy landscape is rounded at the bottom. This is because as X approaches S^0, $|u_i|$ in general increases and $|y_i|$ decreases, and thus the vector $V' = WY$ decreases in size.

Next, assume that S has varied slightly so that $S = S^1$. Then X begins to approach S^1, but the movement is slow because of the energy barrier (Figure 12b). In this process, the above learning not only reduces the energy between S^0 and S^1, but also generates the flow from S^0 to S^1, since V' gains a component in the direction of the vector $S - X$.

Similarly, as S continuously moves, X follows slightly behind S and a gutter is, as it were, engraved in the energy landscape along the track of X (Figure 12c,d). However, if S moves too fast and the input intensity β is small, X cannot follow S and learning fails. Hence, S should be varied slowly or input intensively in the early stage of learning.

By learning the same pattern repeatedly, the gutter becomes deep and clear, that is, the trajectory of X becomes a strong attractor (Figure 12e). In the second and subsequent cycles of learning, X can follow S more easily because it moves in the gutter already engraved to some extent. Accordingly, we had better decrease β gradually and make the movement of X less dependent on the external input as learning proceeds.

After finishing several cycles of learning, the network recalls the learned spatiotemporal pattern without external input when a proper initial state is given.

4.3 Computer Simulation and Discussion

Computer simulations on the above learning were performed using a network of $n = 1000$ neurons. The input pattern S starts from Q^0 at $t = 0$, moves at a constant speed via the same points $Q^0, \ldots, Q^{99}$ as in Section 3.3, and returns to Q^0 at $t = 500\tau$; between Q^ν, s_i for i such that $q_i^\nu \neq q_i^{\nu+1}$ are flipped one by one. When S has returned to Q^0, learning proceeds to the next cycle.

The input intensity β was 0.3 in the first cycle, decreased to 0.2 and 0.1 in the second and third cycles, respectively, and was 0.05 thereafter. Parameters $\alpha = 2$ and $\tau' = 5000\tau$ were used (the others were the same as in Section 3.3).

The behavior of the model during the learning process is shown in Figure 13, where

the overlaps p_ν are redefined by

$$p_\nu = \frac{1}{n}\sum_{i=1}^{n} x_i q_i^\nu \tag{16}$$

and are plotted with solid lines. Broken lines represent overlaps between S and Q^ν. This figure shows that the network state X follows S at approximately constant intervals. We see, however, that p_ν does not reach 1, and that X passes not exactly through but slightly off the orbit of S. This is thought to be because X goes straight toward the current input S although the orbit of S is curved.

At this stage, movement of X still partly depends on the external input $Z = 0.1S$, so that X ceases to follow S if the external input is cut off; after about 4 cycles of learning, however, X is able to move roughly along the orbit without the external input.

The performance of the network after 6 cycles (3000τ) of learning was examined. Figure 14 shows the retrieval process when an initial state with $p_0 = 0.3$ is given and no external signal is input (i.e., $Z = 0$). Similarly to those in Figure 6 and Figure 8, the peak values of p_ν increase to more than 0.9. Since Q^ν are merely intermediate points of S and have no special meaning, we may say that the target orbit of S has been successfully stored even though there are some errors.

Recall errors, namely the difference between $X(t)$ and $S(t)$, exist in a temporal dimension as well as in a spatial dimension. In the case of Figure 14, X moves more slowly than S (it took about 600τ to finish a round of the trajectory). The recall errors do not decrease very much even if learning time is extended. One way of greatly increasing the recall speed is to increase the moving speed of S as the learning proceeds; we can also achieve fast or slow recall of part of the sequence by varying the speed of S in the corresponding part of the orbit.

Next, let us examine the dynamical structure of the network after learning. Figure 15 shows trajectories of the network state X on the p_1-p_0 plane for various initial states. We see that similarly to those in Figure 7, X approaches the line connecting Q^0 and Q^1 nearly perpendicularly, and moves exactly on the line toward Q^1.

This indicates that the network has about the same dynamical structure as that shown in Figure 10. That is, the trajectory of X is included in attractive Ω-subspaces which include the orbit of S; there exists some gap between X and S, but its extent is restricted by this structure.

Incidentally, to reduce the gap, or the recall errors, we should shorten the interval between X and S in learning. Probably the easiest way to do so is to make the learning coefficient α not a constant but a decreasing function of $|u_i|$; then the synaptic weights w_{ij} of neurons such that $x_i \neq s_i$ are modified emphatically, since they generally have small $|u_i|$. This is approximately realized by putting

$$\alpha = \alpha' |y_i|, \tag{17}$$

α' being a positive constant. This would be a natural change because the amount of modification is proportional to the output intensity of the i-th neuron, similarly to the Hebb rule.

The result of applying this learning rule is shown in Figure 16. Compared with Figure 14, the peak values of p_ν are slightly larger, although only 4 cycles of learning were performed. Moreover, the moving speed of X is nearly equal to that of S in learning. We see, therefore, that the improved learning rule reduces both spatial and temporal recall errors and learning time.

5 Concluding Remarks

A NNN model that acts continuously and recalls the stored sequence smoothly has been presented, and its dynamical structure has been discussed. I have also presented a simple algorithm for learning spatiotemporal patterns.

Distinctive features of the NNN model are enumerated in the following.

1. The composition of the network is simple: it is a fully recurrent network without particular delay, synchronization, or control mechanisms.

2. A pattern sequence is stored as a trajectory attractor in the state space: this enables the network to recall the sequence stably even in the presence of substantial noise.

3. The memory capacity is rather large: sequences containing at least $0.2n$ independent patterns can be stored in the network of n neurons.

4. Storage can be performed by a simple learning algorithm: the algorithm is based on a covariance rule and only requires a few repetitions of input.

5. The speed of recalling the sequence can be regulated by varying the synaptic weights or the transition speed of the learning signal.

This is a basic model showing principles of sequential pattern memory; there remain many subjects for future study. For example, mathematical analysis on the dynamical structure and memory capacity is required. Also, the model should be extended for memory of more complex sequences such that the same patterns appear repeatedly in different parts of the sequences (one possible solution to this problem is adding some extra components to the input patterns so that the trajectories in the state space do not intersect).

Finally, I will briefly mention the biological relevance of the model. The NNN itself is not biologically realistic, of course, since the real neuron in the brain does not have such peculiar input-output characteristics as in Figure 1. However, it is significant as a model of the cerebral memory mechanism for the following reasons.

First, the above features, which conventional models do not have, are desirable also for the brain; in particular, the learning algorithm is much more natural than conventional ones and simple enough for the brain to realize. Secondly, the nonmonotonic input-output characteristics can be virtually realized by a local neural circuit consisting of a small number of neurons (in the simplest case, a combination of one excitatory and one inhibitory neuron). Lastly, the distribution of outputs of the NNN is generally broad (i.e., the values of y_i are distributed broadly from -1 to 1), which agrees with a physiological observation by Miyashita (1988) on the sustained activities of neurons related to short-term memory (Morita, 1992); note that the analog Hopfield-type neural networks generally exhibit a bipolar distribution of outputs.

For a detailed discussion on this subject and a more realistic version of this model based on sparse coding, see Morita (1996).

References

Amari, S. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, **C–21**, 1197–1206.

Baram, Y. (1994). Memorizing binary vector sequences by a sparsely encoded network. *IEEE Transactions on Neural Networks*, **5**, 974–981.

Boffetta, G., Monasson, R., & Zecchina, R. (1993). Symmetry breaking in non-monotonic neural networks. *Journal of Physics A*, **26**, L507–L513.

Doya, K., & Yoshizawa, S. (1989). Adaptive neural oscillator using continuous-time back-propagation learning. *Neural Networks*, **2**, 375–385.

Fukai, T., Kim, J., & Shiino, M. (1995). Retrieval properties of analog neural networks and the nonmonotonicity of transfer functions. *Neural Networks*, **8**, 391–404.

Hopfield, J.J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the U.S.A.*, **81**, 3088–3092.

Kleinfeld, D. (1986). Sequential state generation by model neural networks. *Proceedings of the National Academy of Sciences of the U.S.A.*, **83**, 9469–9473.

Kobayashi, K. (1991). On the capacity of a neuron with a non-monotone output function. *Network*, **2**, 237–243.

Miyashita, Y. (1988). Neuronal correlate of visual associative long-term memory in the primate temporal cortex. *Nature*, **335**, 817–820.

Morita, M. (1992). A neural network model of the dynamics of a short-term memory system in the temporal cortex. *Systems and Computers in Japan*, **23**-4, 14–24.

Morita, M. (1993). Associative memory with nonmonotone dynamics. *Neural Networks*, **6**, 115–126.

Morita, M. (1994). Smooth recollection of a pattern sequence by nonmonotone analog neural networks. *Proceedings of the IEEE International Conference on Neural Networks*, Orlando, **2**, 1032–1037.

Morita, M. (1996). Computational study on the neural mechanism of sequential pattern memory. *Cognitive Brain Research*, in press.

Morita, M., Yoshizawa, S., & Nakano, K. (1990). Memory of correlated patterns by associative neural networks with improved dynamics. *Proceedings of the International Neural Network Conference*, Paris, **2**, 868–871.

Nishimori, H., & Opris, I. (1993). Retrieval process of an associative memory with a general input-output function. *Neural Networks*, **6**, 1061–1067.

Okada, M. (1995). A hierarchy of macrodynamical equations for associative memory. *Neural Networks*, **8**, 833–838.

Pearlmutter, B.A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, **1**, 263–269.

Pineda, F.J. (1987). Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, **59**, 2229–2232.

Shiino, M., & Fukai, T. (1993). Self-consistent signal-to-noise analysis of the statistical behavior of analog neural networks and enhancement of the storage capacity. *Physical Review E*, **48**, 867–897.

Sompolinsky, H., & Kanter, I. (1986). Temporal association in asymmetric neural networks. *Physical Review Letters*, **57**, 2861–2864.

Yanai, H., & Amari, S. (1996). Auto-associative memory with two-stage dynamics of non-monotonic neurons. *IEEE Transactions on Neural Networks*, in press.

Yoshizawa, S., Morita, M., & Amari, S. (1993). Capacity of associative memory using a non-monotonic neuron model. *Neural Networks*, **6**, 167–176.

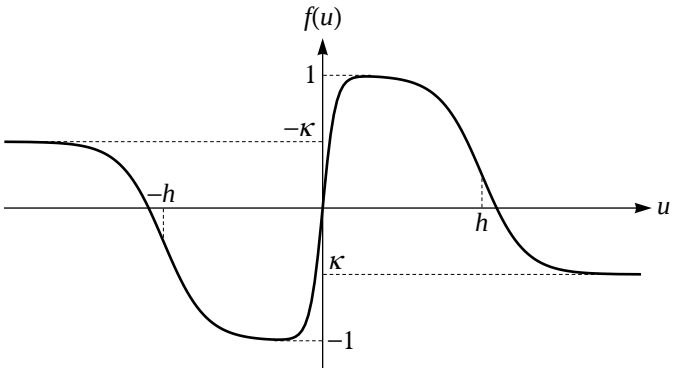

FIGURE 1. Nonmonotonic output function. The detailed shape of the function is not very critical as long as it is nonmonotonic and $uf(u) \leq 0$ when $u \to \pm\infty$. The conventional function (sigmoid function) corresponds to the case that $\kappa = 1$.

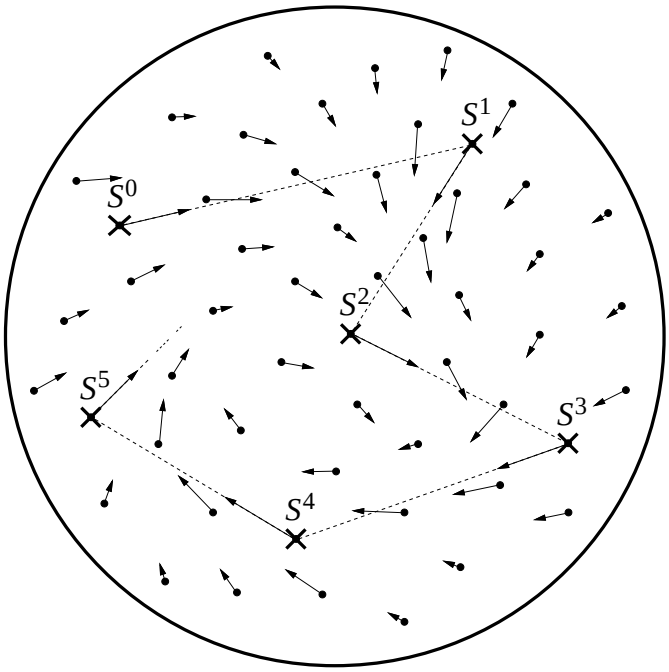

FIGURE 2. Dynamics of a conventional neural network of temporal association. Flow vector field in the state space is depicted. Sequential patterns $S^0, S^1, \ldots$ that are mutually orthogonal are stored using a cross-correlation matrix.

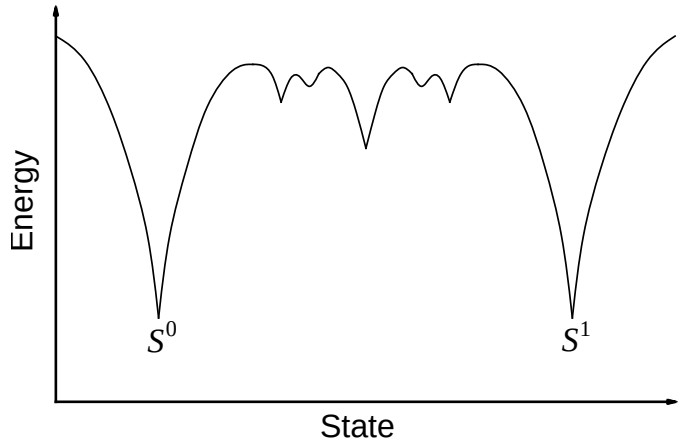

FIGURE 3. Schematic energy landscape of conventional neural networks. Two patterns S^0 and S^1 are stored as attractors. Generally the landscape is bumpy and has many local minima corresponding to spurious memories.

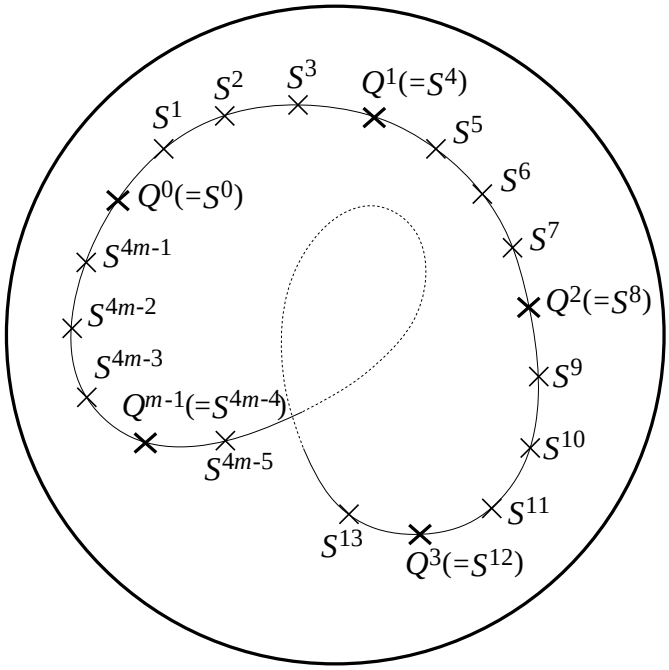

FIGURE 4. Interpolation of patterns. The case of $l = 4$ is shown. A sequence $\{S^\mu\}$ is obtained by interpolating three patterns in each interval of the original sequence $\{Q^\nu\}$.

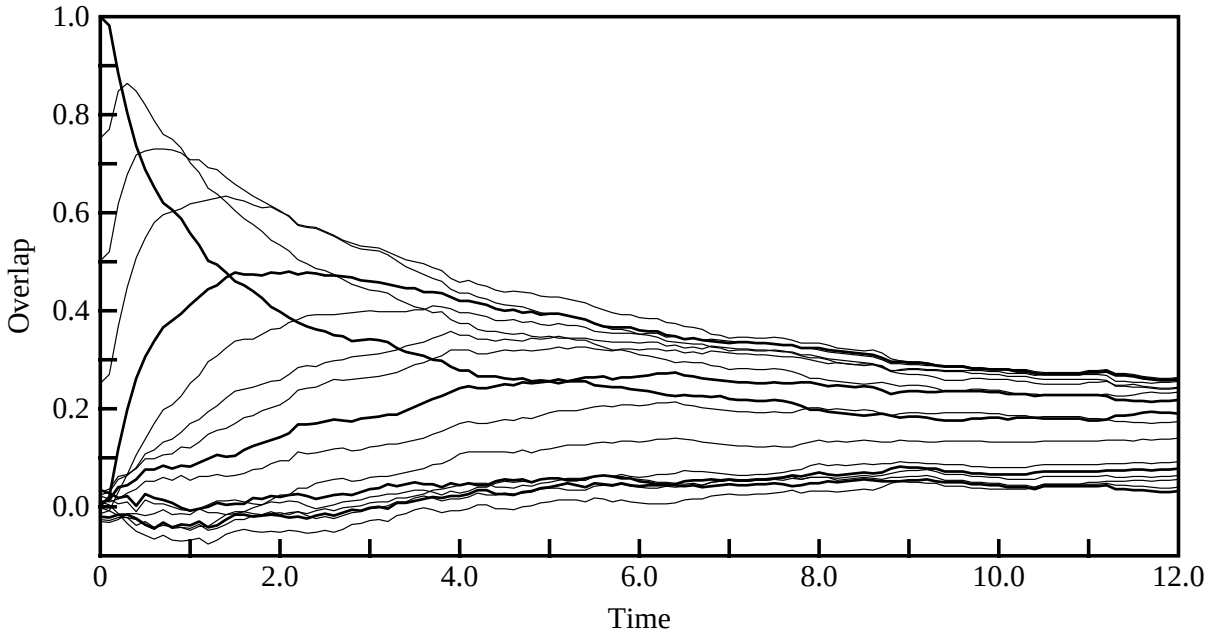

FIGURE 5. Behavior of the network with conventional dynamics. Time course of the overlaps $p_0, \ldots, p_{16}$ is plotted. Time (the abscissa) is scaled by the time constant τ. The initial state is that $X = S^0$. Thick lines represent $p_{l\nu}$, namely, the overlaps with the original patterns Q^ν.

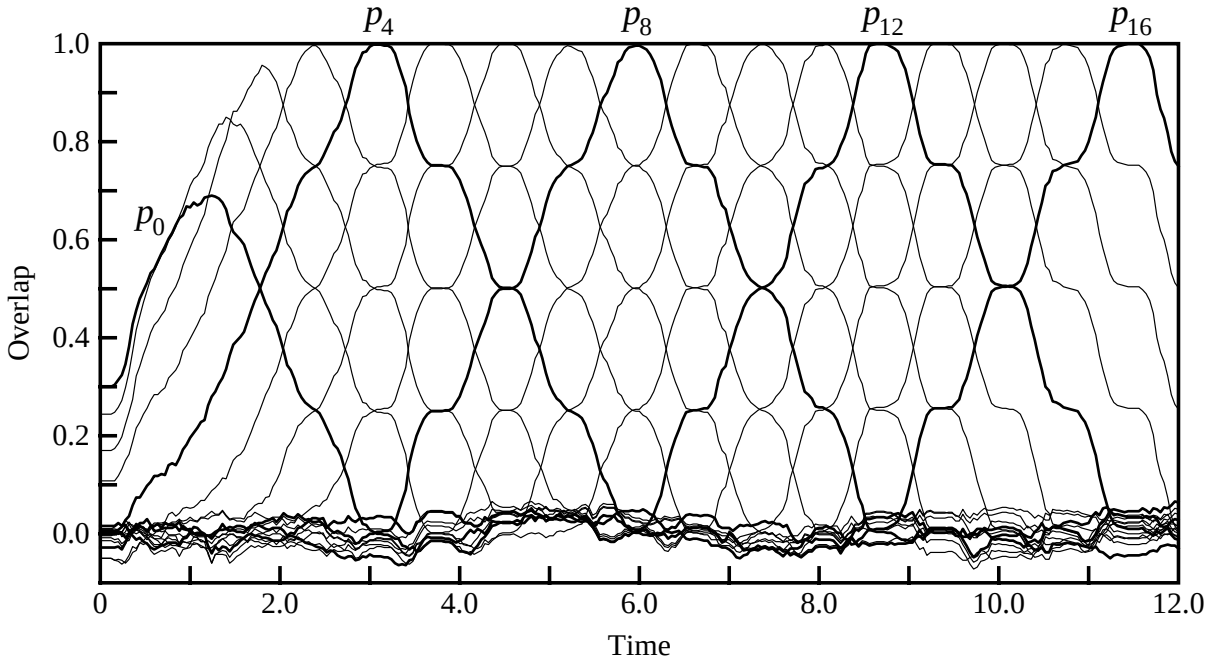

FIGURE 6. Behavior of the NNN. The weight matrix W is the same as that in Figure 5. The initial state was randomly selected out of the states such that $p_0 = 0.3$.

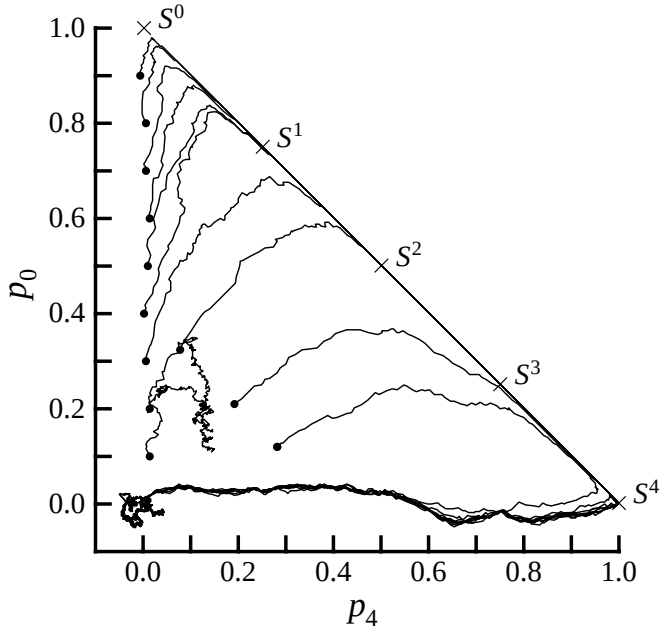

FIGURE 7. Retrieval process of the NNN. Trajectories of $X(t)$ $(0 \leq t \leq 10\tau)$ are plotted for various initial states. The ordinate and the abscissa are the overlaps with $Q^0(= S^0)$ and $Q^1(= S^4)$, respectively. Dots denote the initial states generated by adding noise to S^0 so that $p_0 = 0.1$ to 0.9 (nine along the line $p_4 = 0$) and to S^μ $(\mu = 1, 2, 3)$ so that $p_\mu = 0.4$ (the other three).

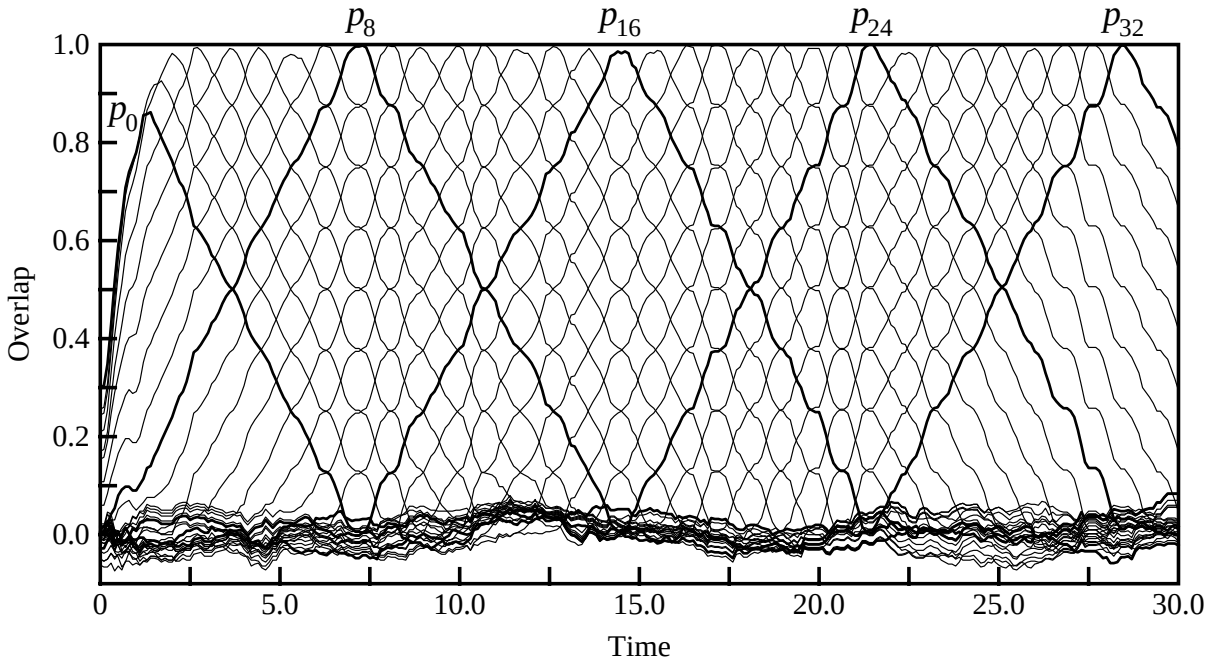

FIGURE 8. Time course of the overlaps for large l. Seven patterns ($l = 8$) were interpolated between Q^ν and $Q^{\nu+1}$. The same initial state as that in Figure 6 is given. The overlaps with Q^ν ($\nu = 0, \ldots, 4$) are represented by thick lines.

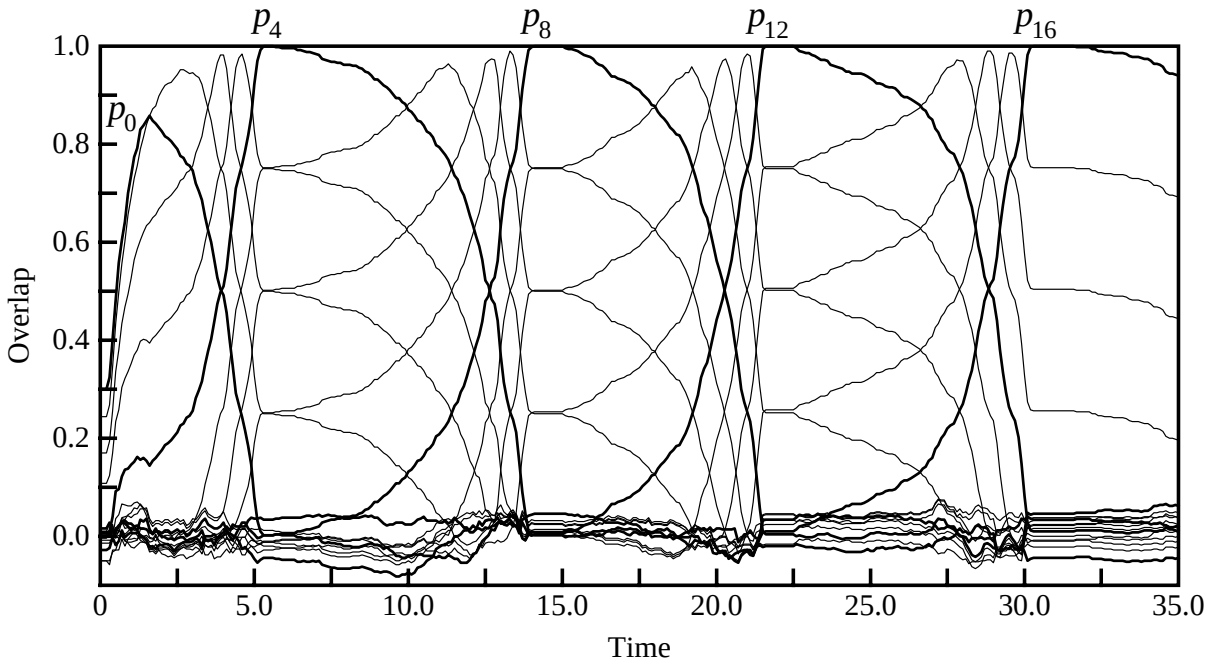

FIGURE 9. Behavior when the weight matrix is given by eqn (10). The parameter $a = 0.52$. The initial state and the interpolation patterns are the same as those in Figure 6.

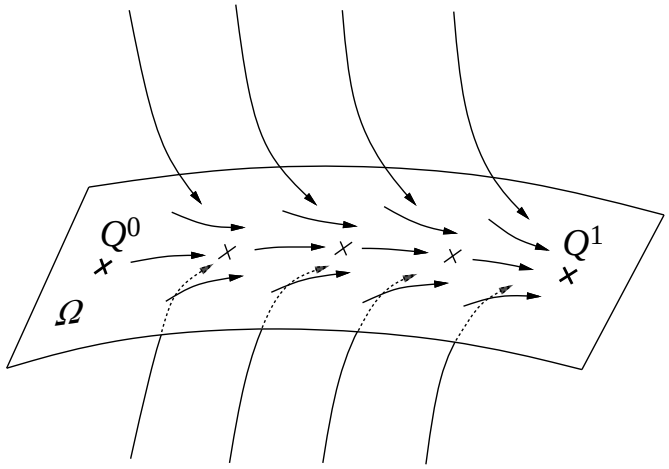

FIGURE 10. Dynamical structure of the model. Flow lines around the orbit between Q^0 and Q^1 are depicted.

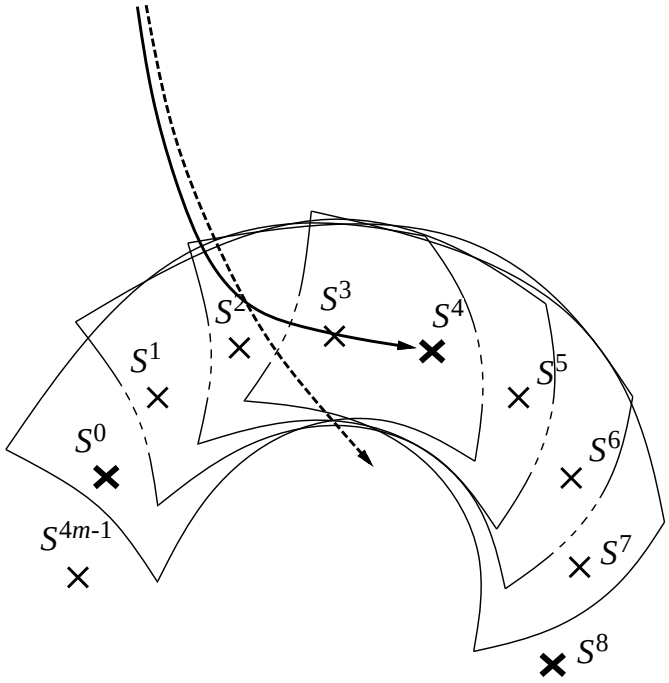

FIGURE 11. Mechanism of stable recall. Neighboring Ω-subspaces mutually overlap near the orbit connecting S^μ. After the network state X has reached the subspace, it moves along the overlapping parts. The broken line represents a trajectory for conventional dynamics.

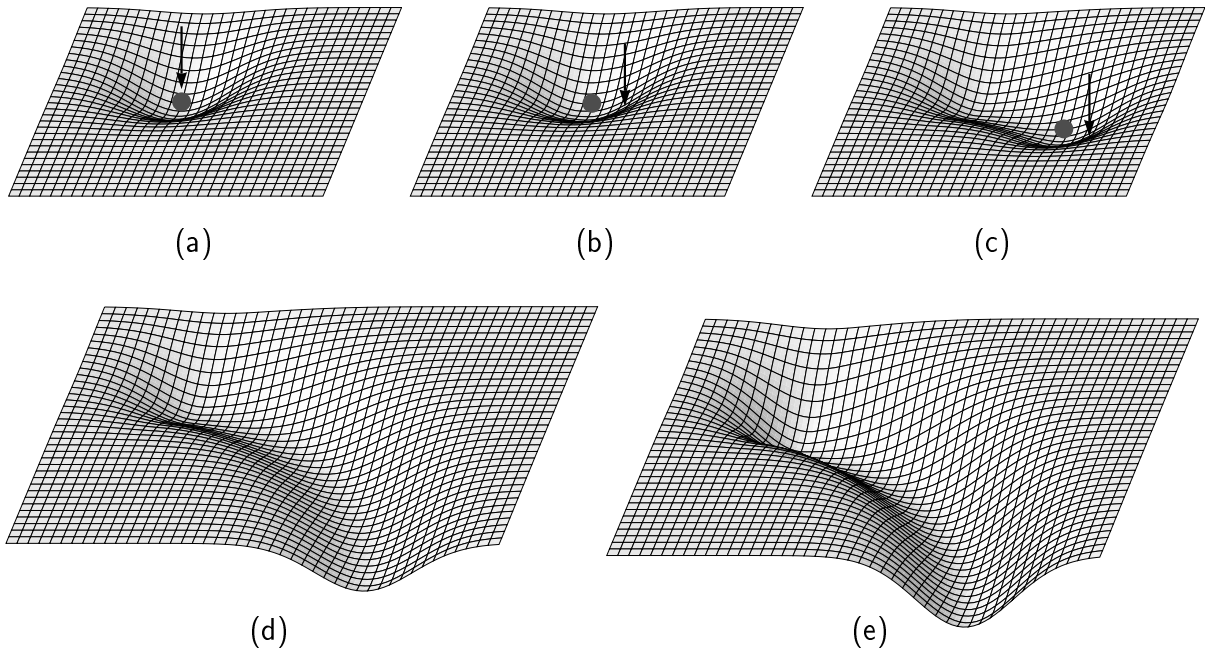

FIGURE 12. Illustration of the learning process. Change in the imaginary energy landscape is depicted in (a)–(e). The current network state X and input pattern S are represented by the dot and the arrow, respectively.

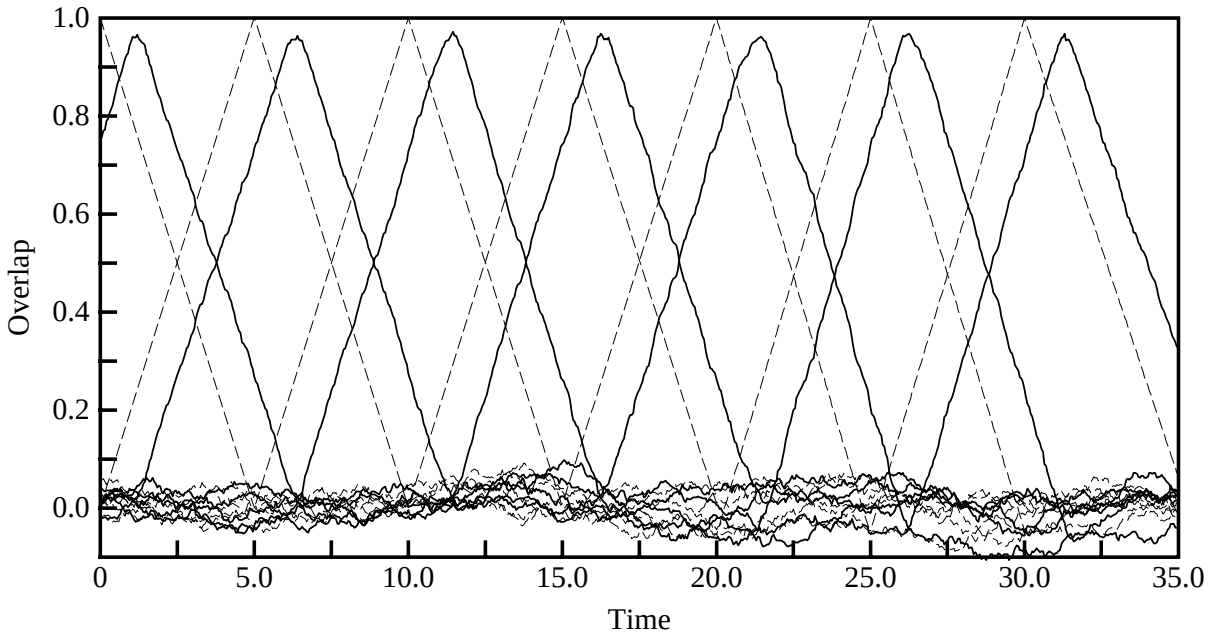

FIGURE 13. Progress of learning. The overlaps between X and Q^ν (solid lines) and between S and Q^ν (broken lines) are shown. The abscissa is time scaled by the time constant τ, and the origin is the point when the third cycle of learning started (1000τ after the start of learning).

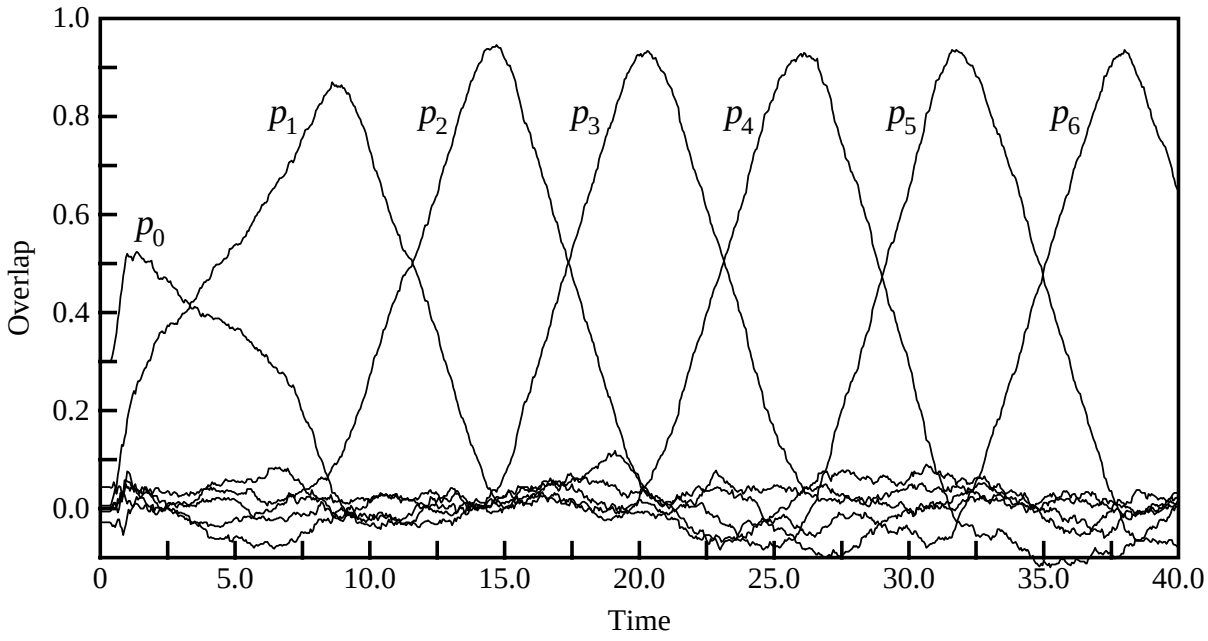

FIGURE 14. Recall process after 6 cycles of learning. Time course of the overlaps p_ν ($\nu = 0, \ldots, 6$) is plotted. The initial state is the same as that in Figure 6.

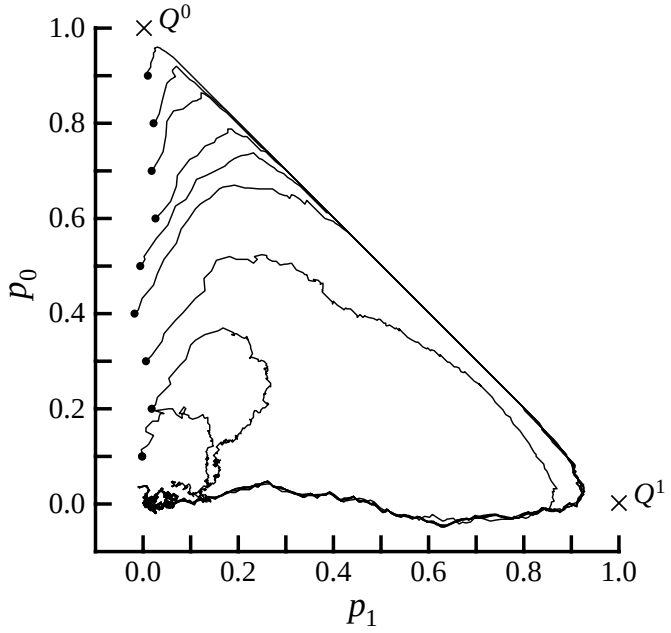

FIGURE 15. Trajectories of $X(t)$ ($0 \leq t \leq 20\tau$) on the p_1-p_0 plane. Initial states denoted by dots have an overlap of 0.1 to 0.9 with Q^0. Retrieval is successful when the initial overlap $p_0 \geq 0.3$.

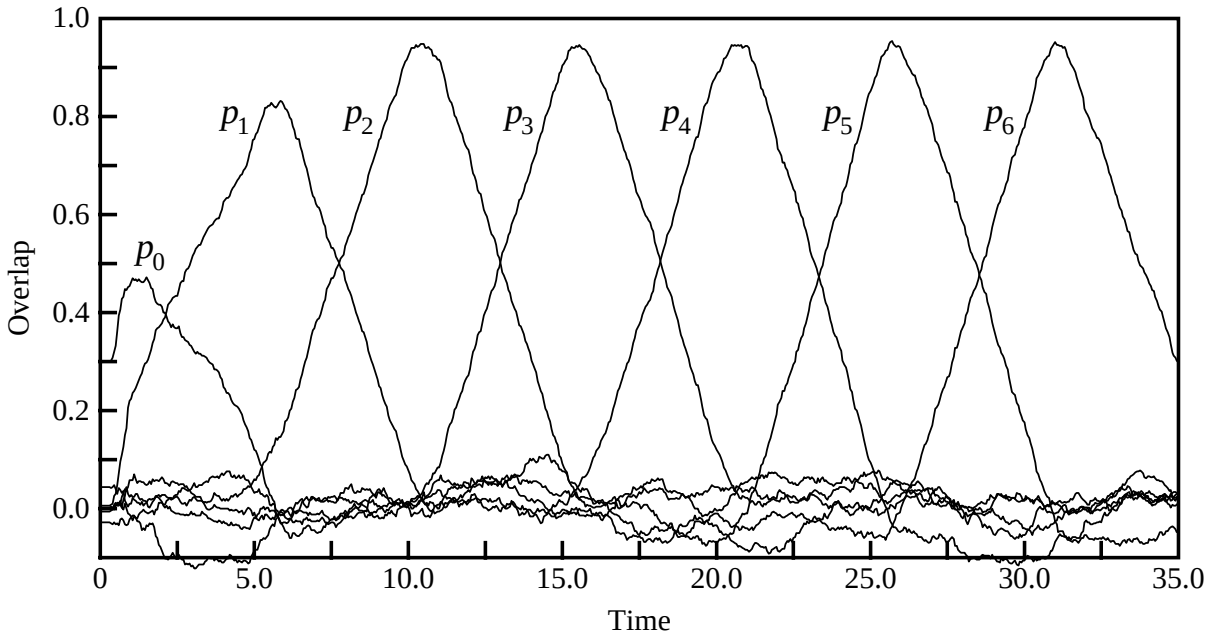

FIGURE 16. Recall process when the improved learning rule was applied. The parameter $\alpha' = 5$ and the learning time was 2000τ (4 cycles); the other conditions were the same as those in the previous simulation.