Computational Study on the Neural Mechanism of Sequential Pattern Memory

Masahiko Morita

Institute of Information Sciences and Electronics, University of Tsukuba

Keywords: Sequential pattern memory, Neural network models, Network dynamics, Nonmonotonic characteristics, Local inhibition cells, Sparse coding, Learning algorithm, Covariance rule.

Summary: The brain stores various kinds of temporal sequences as long-term memories, such as motor sequences, episodes, and melodies. The present study aims at clarifying the general principle underlying such memories. For this purpose, the memory mechanism of sequential patterns is examined from the viewpoint of computational theory and neural network modeling, and a neural network model of sequential pattern memory based on a simple and reasonable principle is presented. Specifically, spatiotemporal patterns varying gradually with time are stably stored in a network consisting of pairs of excitatory and inhibitory cells with recurrent connections; such a pair can achieve nonmonotonic input-output characteristics which are essential for smooth sequential recall. Storage is performed using a simple learning algorithm which is based on the covariance rule and requires only that the sequence be input several times, and retrieval is highly tolerant to noise. It is thought that a similar principle is used in cerebral memory systems, and the relevance of this model to the brain is discussed. Also, possible roles of hippocampus and basal ganglia in memorizing sequences are suggested.

This article is to be published in *Cognitive Brain Research*.

Masahiko Morita
Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Ibaraki 305, Japan
Phone: (+81-298) 53-5321
Fax: (+81-298) 53-5206
E-mail: mor@is.tsukuba.ac.jp

1 Introduction

In the brain, it is thought that motor sequences are represented by sequential patterns of neuronal activities; some of these patterns are stored as long-term memories in the cerebral cortex (possibly in the premotor cortex), and retrieved when necessary. Also, memory of episodes (meaning chains of events) and melodies is regarded as sequential pattern memory. Although these various kinds of memories are stored in different cortical areas by different mechanisms, there must be a common principle underlying them because the basic structure and characteristics of the cerebral neural networks do not differ much among the areas.

This fundamental principle is not understood; in fact, we do not have even a likely candidate for it. A number of artificial neural network models of sequential pattern memory have been proposed, but they are based on principles which are not reasonable for application to the brain.

The purpose of the present paper is illuminate the structure, dynamics, coding, and learning algorithm of the neural networks of sequential pattern memory from a computational viewpoint. For this purpose, I will show a neural network model based on a simple and reasonable principle, and discuss the relevance of the model to the brain.

Before describing such a model, I will briefly explain why conventional neural network models are unreasonable and the origin of the problem.

2 Conventional Models

Let us consider coding. For simplicity, we assume that each neural element has two states, active and inactive.

The simplest type of coding is grandmother-cell coding, where only a single element is active at a time. Since this coding is very susceptible to noise, a variation as that shown in Fig. 1a is often used. In this case, the active period of an element is overlapped with that of others, but the coding is still of a grandmother-cell type rather than a population type because each element codes only a single part of a particular sequence.

If sequential patterns are encoded in this way, they can easily be stored by connecting the elements in order; retrieval is also easy. This coding, however, is quite inefficient because it requires as many sets of elements as there are stored sequences; in addition, it still has a low tolerance to noise.

Another type of coding is population coding; that is, a sequence is represented by a spatiotemporal pattern, where various sets of elements are active at a time and an element codes various parts of various sequences, as shown in Fig. 1b. In contrast with grandmother-cell coding, population coding is efficient and tolerant to noise. Actually

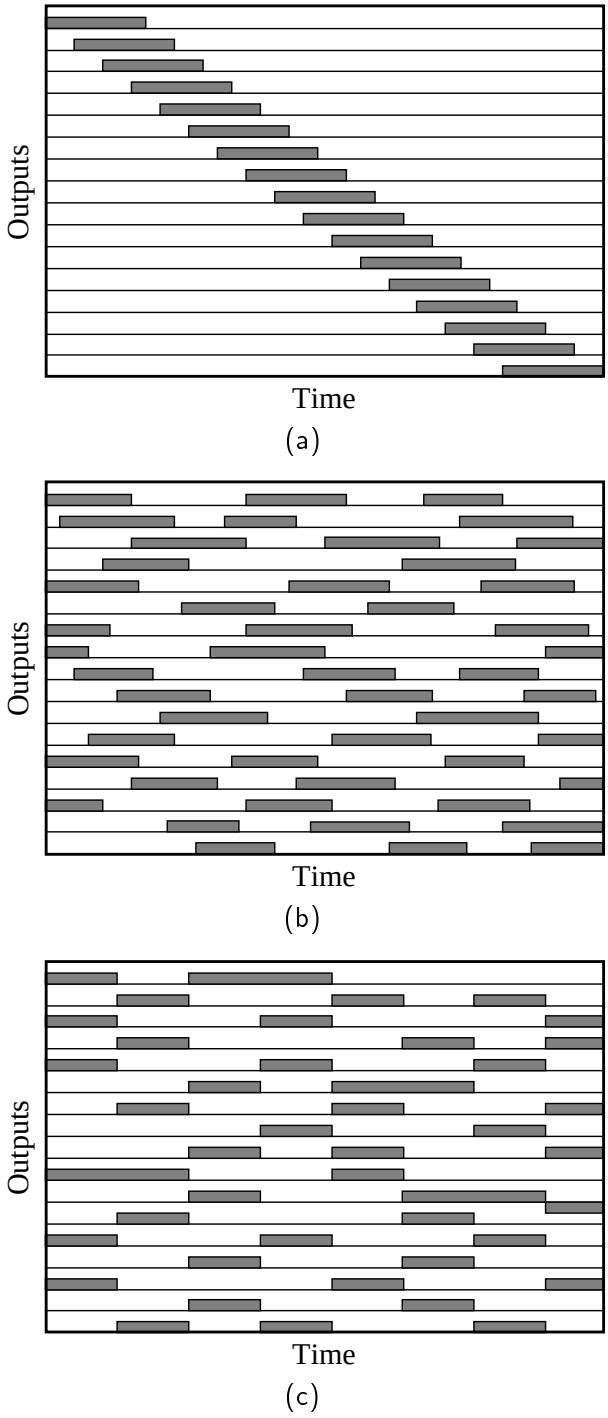

Fig. 1. Types of coding used in neural network models of sequential pattern memory: (a) grandmother-cell coding; (b) population coding; (c) population coding with synchronization.

however, coding as in Fig. 1b has not been used; a special type of population coding where all the elements update their state synchronously (Fig. 1c) has been used in conventional models[1,3,11].

The reason is explained intuitively by Fig. 2, which shows typical network dynamics of conventional models. In this figure, the abscissa represents the state or the activity

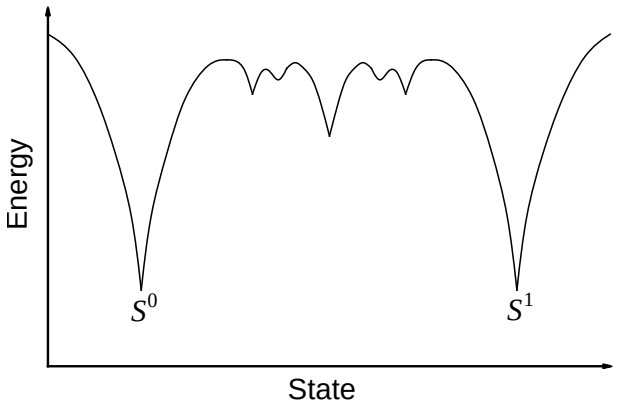

Fig. 2. Schematic energy landscape of conventional neural networks. Two patterns S^0 and S^1 are stored as attractors. The network state cannot move from S^0 to S^1 without jumping a distance because the energy landscape has deep valleys at these states.

pattern of the neural network, and the ordinate is the energy representing stability of the network state. By plotting the energy at each state, the "landscape of energy" of the system can be depicted.

An energy minimum corresponds a stable state called an attractor because the network changes its state in such a way that the energy decreases; memory patterns are embedded in strong attractors located at the bottom of deep valleys. Generally the energy landscape is acute at the stored patterns and they are always separated from each other because the attracting force becomes stronger as the state of the network approaches them.

Accordingly, to retrieve stored patterns sequentially, the network state has to jump a distance, that is, many elements must change their state simultaneously, since it cannot move gradually from one attractor to another because of the energy barrier. That is why synchronization among elements is necessary for conventional models.

However, such synchronous coding is unnatural because neighboring parts of a sequence are encoded in quite different patterns; that is, an intermediate pattern between them does not code an intermediate part of the sequence, nor can the sequence be retrieved from such an intermediate pattern. In addition, a special mechanism for synchronization is required.

It is thought, therefore, that coding such as that shown in Fig. 1b is the most reasonable. Then, how can we store so encoded patterns and realize recall without synchronization?

One approach to this problem is to improve the learning algorithm. Several algorithms for sequential pattern learning have been proposed[9,12] which are extensions of the back-propagation algorithm. These algorithms, however, are extremely complicated, and in reality, they do not work very well without synchronization.

This implies that the fundamental cause of the problem lies not in the learning algo-

rithm but in the dynamics and that it is necessary to improve network dynamics. Indeed, the above problem is solved by modifying the network dynamics as described in the next section.

3 Theoretical Model

It has recently been found that most critical problems in conventional memory models originate in a basic property of their dynamics that the output of each element increases with the total input to the element, and a neural network model whose elements have nonmonotonic input-output characteristics was proposed[7]. This model, called a nonmonotone neural network, exhibits very high performance for memory of static patterns; for example, its memory capacity is more than twice as large as that of conventional models[13].

The above problem of sequential pattern memory is also attributed to the conventional monotonic dynamics, and the nonmonotone model is valid for memory of sequences[8]. Though this model is more theoretical than realistic, I will describe it to clarify the principle.

3.1 Structure and Dynamics

The general structure of the model is shown in Fig. 3. Sequential pattern $S(t)$ is input to a neural network N_1 and stored there; this network has recurrent connections, and its current state (activity pattern) is denoted by X. A learning signal R from another network N_2 is also input to the network N_1 in the learning mode, which specifies the code,

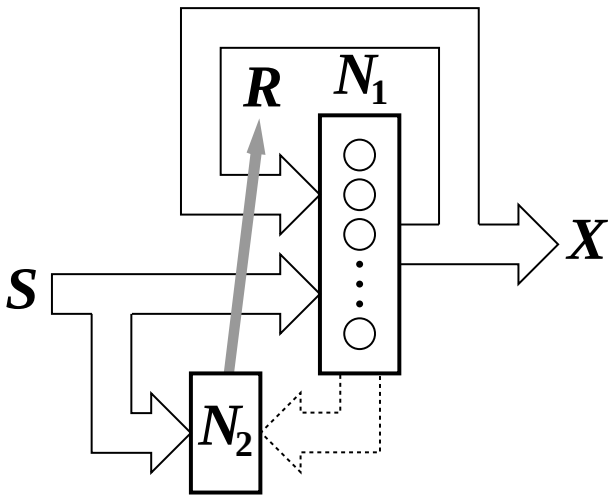

Fig. 3. Structure of the model. S is the input to the model, R is a learning signal, and X is the state (activity pattern) of network N_1.

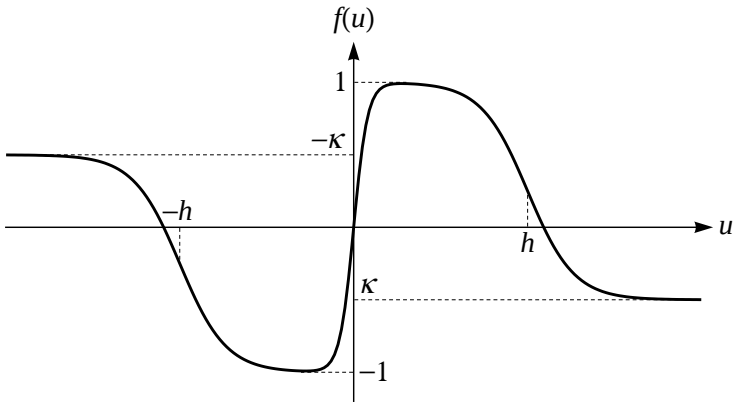

Fig. 4. Input-output characteristics of the nonmonotonic element. The detailed form of the function $f(u)$ is not very critical as long as it is nonmonotonic and $\kappa \leq 0$. In conventional models, a monotonically increasing function ($\kappa = 1$) is used.

or the network state in which the current input pattern should be stored.

We will deal with the simplest case where $R = S$, that is, where the input pattern is stored as is without transformation and S itself is used as the learning signal. In this case, N_2 is simply a relay point (the function of network N_2 will be discussed in Sec. 5.2).

The most distinctive feature of this model is that the elements of the memory network N_1 have nonmonotonic input-output characteristics as shown in Fig. 4. Specifically, each element acts according to the equations

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} w_{ij} y_j + z_i, \quad (1)$$

$$y_i = f(u_i), \quad (2)$$

where u_i denotes the instantaneous potential of the i-th element, y_i the output, z_i the external input, τ a time constant, and n the number of elements; $f(u)$ is a nonmonotonic function shown in Fig. 4.

These formulas are the same as those often used in conventional models except that the output (or activation) function $f(u)$ is not a monotonically increasing function of u but a nonmonotonic function. This modification causes a significant change in dynamical properties of the network, and enables the network to memorize sequential patterns easily.

3.2 Coding

We assume that the input pattern $S = (s_1, ..., s_n)$ at an instant is a binary vector of which about half of the components s_i are 1 and the rest -1, and that S gradually varies with time. That is, this model deals with non-sparse population coding without synchronization.

We also assume that the result of retrieval is obtained by the vector $(\mathrm{sgn}(u_1), ..., \mathrm{sgn}(u_n))$,

where $\mathrm{sgn}(u) = 1$ for $u > 0$ and -1 otherwise; we will treat this vector as the network state X for convenience.

3.3 Learning Algorithm

The learning algorithm is very simple. In parallel with the above network dynamics, we have only to input patterns successively and modify the synaptic weights according to the covariance rule between the recurrent input vector $Y = (y_1, \ldots, y_n)$ and the learning signal vector $R = (r_1, \ldots, r_n)$. Specifically, synaptic weights are modified according to

$$\tau' \frac{dw_{ij}}{dt} = -w_{ij} + \alpha\, r_i y_j, \tag{3}$$

where α denotes a positive learning coefficient and τ' is a time constant of learning ($\tau' \gg \tau$). The coefficient α may be a constant, but the learning performance is improved if α is a decreasing function of $|u_i|$, which is approximately realized by putting $\alpha = \alpha'|y_i|$ (α' is a positive constant).

The external input vector $Z = (z_1, \ldots, z_n)$ is generally a function of the input pattern S and the learning signal R. Since we are dealing with the case of $R = S$, we may simply put

$$Z = \lambda S, \tag{4}$$

where λ is a positive coefficient representing the input intensity of the learning signal. If $R \neq S$, then Z should be given, for example, by

$$z_i = \sum_k a_{ik} s_k + \lambda r_i, \tag{5}$$

where a_{ik} are synaptic weights from the k-th input element; a_{ik} should also be modified according to

$$\tau' \frac{da_{ik}}{dt} = -a_{ik} + \alpha\, r_i s_k \tag{6}$$

so that Z becomes proportional to R.

3.4 Learning Process

The process of learning is schematically shown in Fig. 5. In this figure, change in the "energy landscape" of the memory network is depicted, where the n-dimensional state space of the network is represented by the x-y surface and the energy is represented by the z-axis; a solid circle and an arrow represent the current network state X and the current learning signal R ($= S$), respectively. Although actually the energy cannot be defined in nonmonotone neural networks, it would be useful for our intuitive understanding to assume a landscape such that X goes downhill.

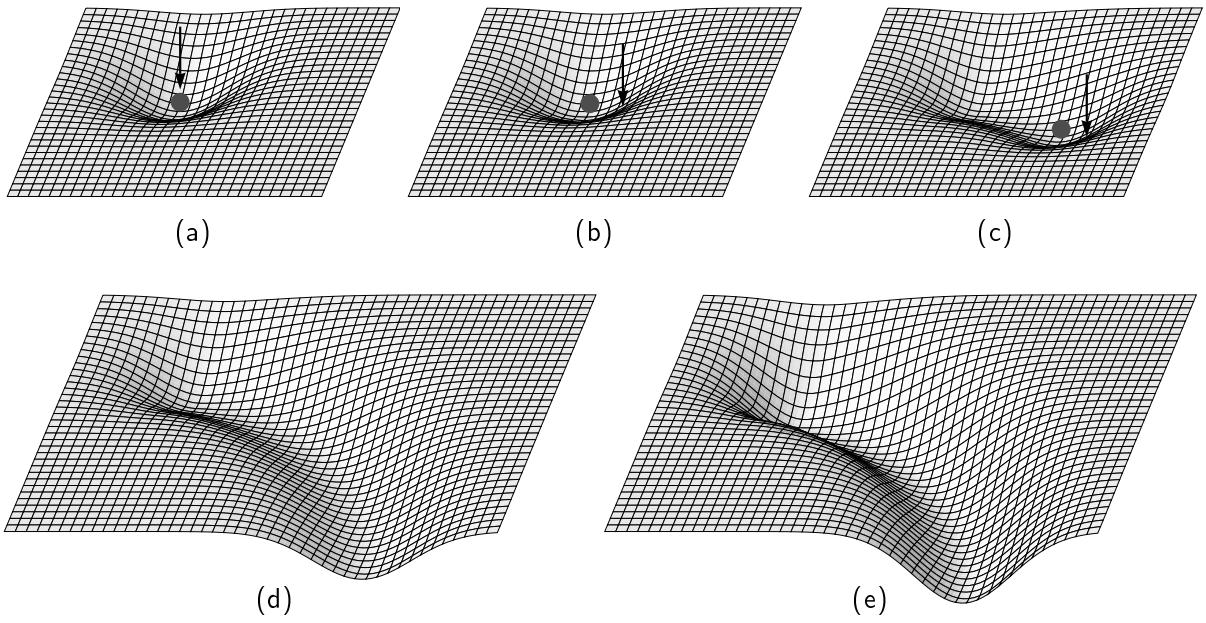

Fig. 5. Illustration of the learning process. Change in the imaginary energy landscape is depicted in (a)–(e). Each point on the surface corresponds to a state or an activity pattern of the network, and the depth represents energy or stability at the state. The current network state X and input pattern S are represented by the solid circle and arrow, respectively.

First, assume that a static pattern has been input and R is kept constant for some time. Soon $X = R$ due to the external input $Z = \lambda R$. In the meantime, the energy around this point decreases through learning and thus it becomes a point attractor of the system (Fig. 5a). Accordingly, the static pattern is stored in the same way as in conventional models.

It should be noted, however, that unlike Fig. 2, the energy landscape is rounded at the bottom. This is the effect of the nonmonotonic characteristics; that is, as X approaches the attractor, $|u_i|$ in general increases and $|y_i|$ decreases, and thus the attractive force decreases.

Next, assume that the input pattern has varied so that R has moved slightly. Then X begins to approach R, but the movement is slow because of the energy barrier (Fig. 5b). In this process, the above learning not only reduces the energy between X and R, but also induces the flow from X toward R.

Similarly, as the input pattern changes successively and R moves continuously, X follows slightly behind R and a gutter is engraved in the energy landscape along the track of X (Fig. 5c,d). However, if R moves too fast and the input intensity λ is small, X cannot follow R and learning fails. Hence, R should be varied slowly or input intensively in the early stage of learning (however, when a different sequence starts, R should jump so that the sequence may not be joined to the previous one).

By learning the same sequence repeatedly, the gutter becomes deep and clear, that is,

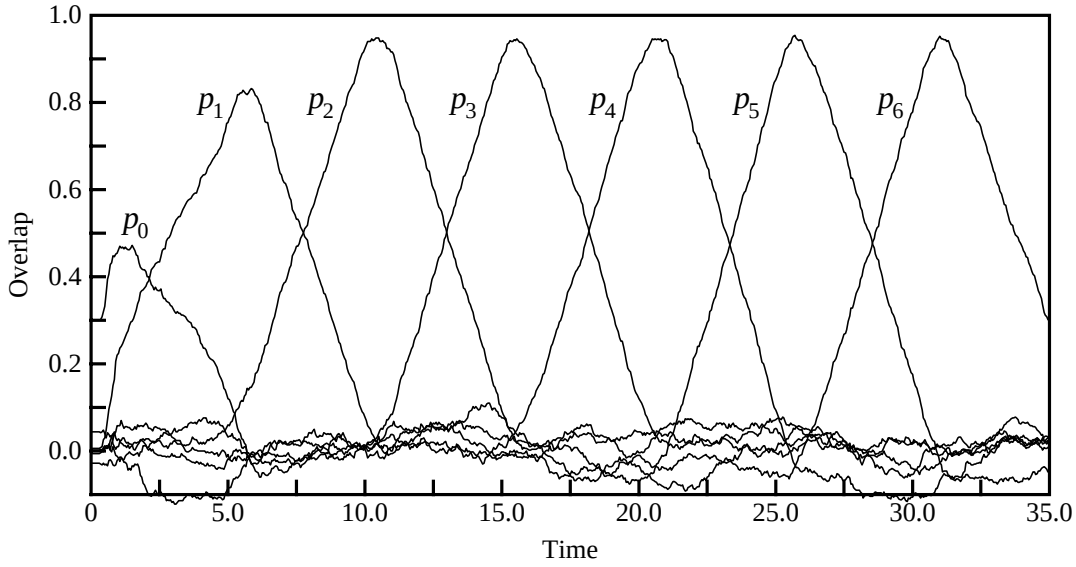

Fig. 6. Recall of the sequence. The time course of the overlaps p_μ between the network state X and the intermediate patterns S^μ after learning is plotted. Time is scaled by the time constant τ.

the trajectory of X becomes a strong string-type attractor (Fig. 5e). In the second and subsequent cycles of learning, X can follow R more easily because it moves in the gutter already engraved to some extent. Accordingly, it is best to decrease λ 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 sequence without external input when a proper initial state or a key pattern, usually the head of the sequence, is given.

3.5 Computer Simulation

Computer simulations on the above model were performed using a network with $n = 1000$ elements. For convenience, a cyclic sequence $S^0 \to S^1 \to \cdots \to S^{99} \to S^0$ was input to the network. Intermediate patterns S^μ were selected at random, and the learning signal R was varied gradually from S^μ to $S^{\mu+1}$. After finishing 4 cycles of learning, the external input was cut off (i.e., $Z = 0$), and various key patterns were given to the network.

Figure 6 shows the time course of the overlaps p_μ between the network state X and the intermediate patterns S^μ defined by

$$p_\mu = \frac{1}{n} \sum_{i=1}^{n} x_i s_i^\mu. \tag{7}$$

The initial network state was given such that $p_0 = 0.3$ and $p_\mu \simeq 0$ for $\mu \geq 1$.

We see that though p_0 does not increase much, p_1 increases up to 0.94 and p_μ for $\mu \geq 2$ successively reach a peak value of more than 0.95. This graph indicates that X changes

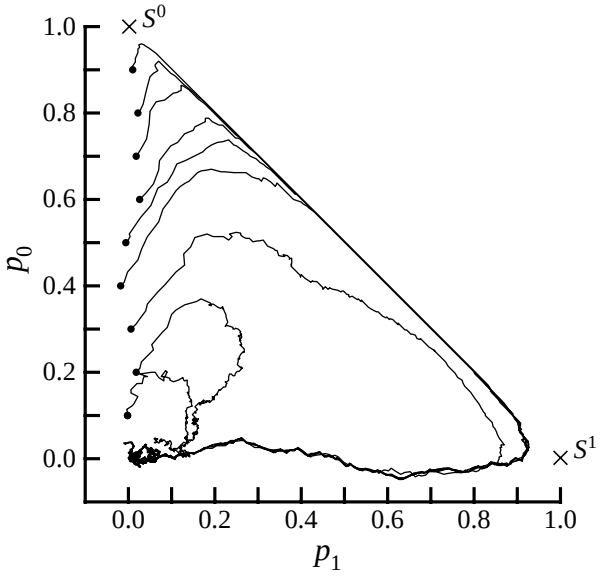

Fig. 7. Retrieval process. Trajectories of $X(t)$ $(0 \leq t \leq 20\tau)$ for various initial states (represented by dots) are projected onto the p_1-p_0 plane. Recall is successful when the initial overlap $p_0 \geq 0.3$.

continuously from S^μ to $S^{\mu+1}$; in other words, the learned sequence is smoothly recalled. It should be noted that the moving rate of X, or the recall speed, is almost the same as that of R in learning.

Figure 7 shows the retrieval process in a different way, where trajectories of the network state X for various initial states are plotted on a plane. We can see from this graph that even if the key pattern is rather different from S^0, X quickly approaches the line connecting S^0 and S^1, which corresponds to the bottom of the energy gutter in Fig. 5, and then moves along it. This indicates that the learned sequence is indeed stored as a string-type attractor and that it can be retrieved even in the presence of substantial noise.

4 Realistic Model

The model in the previous section provides many significant merits, and its structure and learning algorithm are, in principle, simple enough to be realized in the brain. Concerning dynamics and coding, however, the model has some problems when we regard it as a model of the brain.

Firstly, neurons do not have the unusual input-output characteristics described in Fig. 4, but generally have monotonic characteristics. Hence, the nonmonotonic element, which is essential for the above dynamics, is not realistic.

Secondly, active and inactive states were symmetric and thus active elements were about half of all the elements; however, this is not the case with the brain. Though detailed coding in the brain is not clear, it is certain that the number of active neurons

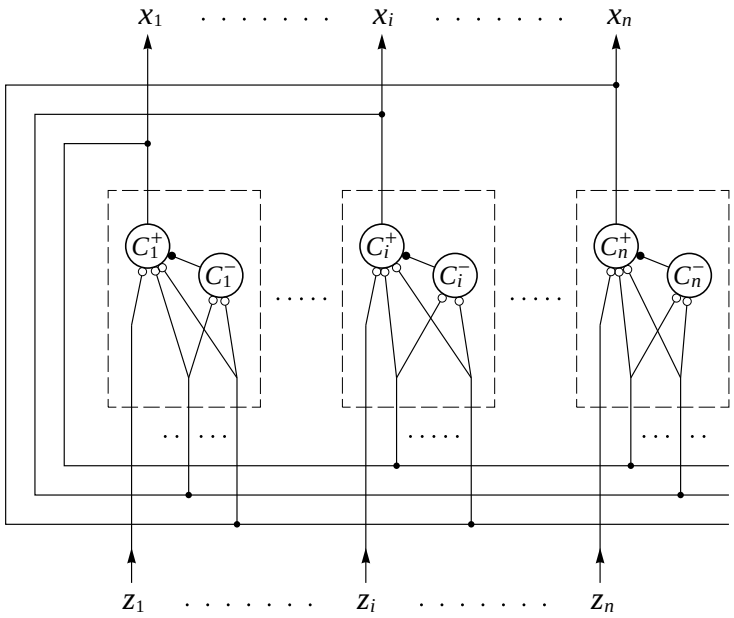

Fig. 8. Structure of the memory network. A pair of excitatory and inhibitory cells composes a unit corresponding to a single element in the previous model.

is much smaller than that of inactive ones.

From physiological observations, it seems that most cerebral memory systems use population coding with a limited number of active neurons, called sparse coding. Theoretical studies have shown that the memory capacity of the network increases markedly if sparsely encoded patterns are stored and the total activities (or the number of active elements) of the network are kept constant[2]; however, it is not easy to make a realistic model with sparse coding because fixing the total activities in a natural manner is difficult.

To solve these problems and make the model more realistic, I will present another neural network model. This model was originally constructed by Morita[6] for explaining the sustained activities of neurons in the inferotemporal cortex[5]. These neurons are thought to relate to memory of static patterns, but the model can also memorize sequential patterns in the following way.

4.1 Structure and Dynamics

The general structure of the model is the same as that in Fig. 3, but the memory network N_1 is so modified that it consists of excitatory and inhibitory cells as shown in Fig. 8.

A part surrounded by broken lines in this figure represents a unit, where the output cell C_i^+ emits the output of the unit and the inhibition cell C_i^- sends a strong inhibitory signal to the output cell. Both cells receive recurrent signals from other units. In mathematical

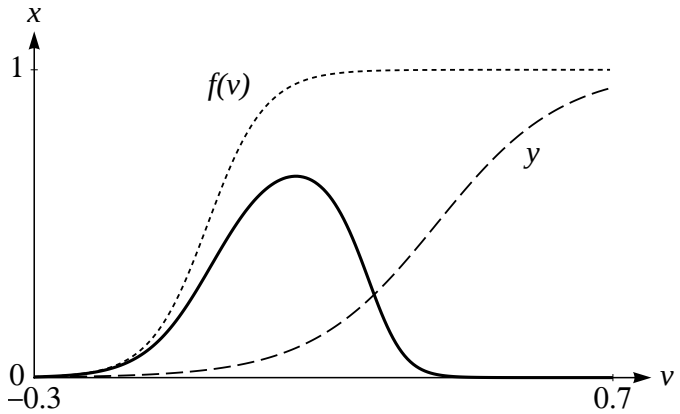

Fig. 9. Input-output characteristics of the unit. Without the inhibition cell, the output x monotonically increases with the total input v (dotted line), but x is a nonmonotonic function of v (solid line) because of the output y of the inhibition cell (broken line).

terms,

$$y_i = f\Big(\sum_{j=1}^{n} w_{ij}^{-} x_j - \theta\Big), \tag{8}$$

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{n} w_{ij}^{+} x_j - w_i^{*} y_i + z_i, \tag{9}$$

$$x_i = f(u_i), \tag{10}$$

where x_i and y_i are the outputs of C_i^+ and C_i^-, respectively, u_i is the potential, z_i is the external input, w_{ij}^+ and w_{ij}^- are synaptic weights from the j-th unit to C_i^+ and C_i^-, respectively, w_i^* represents the efficiency of the inhibitory synapse from C_i^- to C_i^+, and θ is a positive constant.

The output function $f(u)$ of each cell is a monotonic sigmoid function increasing from 0 to 1. However, the input-output characteristics of the unit are nonmonotonic as shown in Fig. 9. The output x increases with the total input v when v is small enough and the output y of the inhibition cell is small, but it decreases when v becomes large. The unit corresponds to the nonmonotonic element of the previous model; however, the peak value of x is not fixed but varies with the ratio of w_{ij}^+ to w_{ij}^-.

4.2 Coding

In this model, the learning signal R should be a sparse vector in which most components r_i are 0 and the rest are 1, and r_i vary gradually with time so that sparse coding without synchronization can be realized. The rate of active components should be around 1 to 10 percent.

Here we deal again with the simple case where the input pattern S itself can be used as the learning signal; thus we assume that $R = S$ and $Z = \lambda S$.

4.3 Learning Algorithm

The learning algorithm of this model is similar to that of the previous model except for the learning rule which has to be modified because more than one kind of synapses exist.

As a result of theoretical examination and computer simulations on various rules, the following formulas were obtained:

$$\tau' \frac{dw_{ij}^{+}}{dt} = -w_{ij}^{+} + \alpha\, r_i x_j, \tag{11}$$

$$\tau' \frac{dw_{ij}^{-}}{dt} = -w_{ij}^{-} - \beta_1 r_i x_j + \beta_2 x_i x_j + \gamma, \tag{12}$$

where α, β_1, and β_2 are learning coefficients, and γ is a positive constant representing lateral inhibition among units. The coefficient α may be a positive constant, but the learning performance is better when α is a decreasing function of x_i; β_1 and β_2 are constants which satisfy $0 < \beta_1 < \beta_2$.

If the i-th unit receives a learning signal ($r_i = 1$) and its output x_i is small ($x_i < \beta_1/\beta_2$), then w_{ij}^{+} are reinforced and w_{ij}^{-} are depressed, thus the output x_i increases. When x_i becomes large, however, w_{ij}^{-} is reinforced, thus x_i is restrained from growing too much. If $r_i = 0$, then only w_{ij}^{-} is reinforced, thus x_i decreases.

It should be noted that the term $\beta_2 x_i x_j$, or reinforcing w_{ij}^{-} according to the final output x_i of the unit, is indispensable for maintaining the nonmonotonic characteristics. Also, the term γ or lateral inhibition plays an important role in keeping the total activity of units at a low level (a small increase in the total activity causes a large increase in the total inhibitory signals), allowing sparse coding. In contrast, it is not very important that the efficiency w_i^{*} of the inhibitory synapse be modified; it may be a constant with a large value.

4.4 Computer Simulation

Computer simulations were performed using a network of 1000 units. The input pattern varies gradually from S^0 through $S^1, \ldots, S^{99}$ and returns to S^0, where S^μ are random vectors such that 10% of the components are 1 and the rest are 0; thus, on average, each unit codes 10 different parts of the sequence. A part of the input spatiotemporal pattern (from $S = S^0$ to $S = S^{19}$) is shown in Fig. 10a.

The behavior of the model after 5 cycles of learning is shown in Fig. 10b, where the time course of the outputs x_i of units is plotted. A key was given by inputting S^0 with noise (50 components are flipped from 1 to 0 and 50 from 0 to 1) for a short time ($0 \leq t \leq$

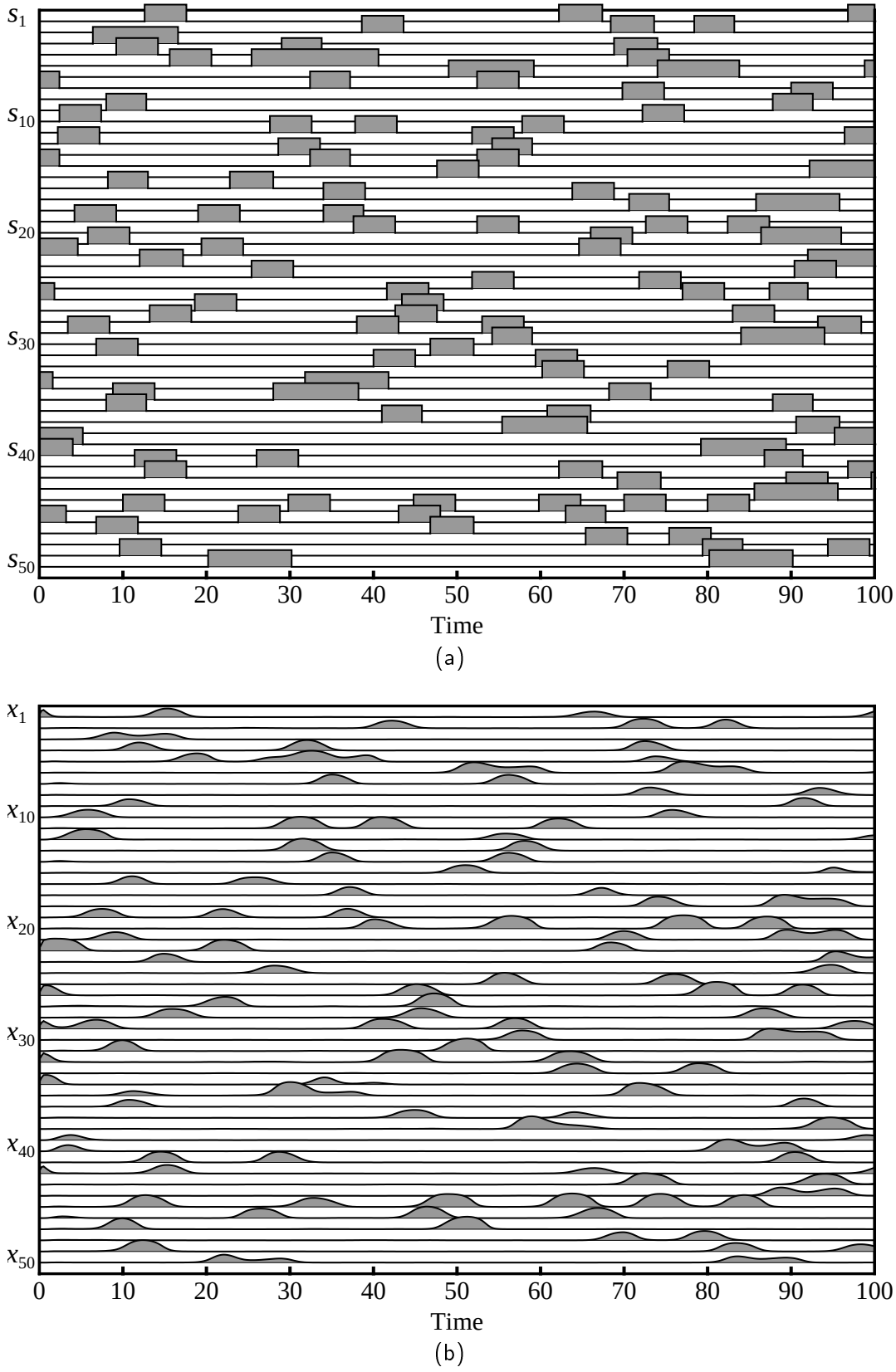

Fig. 10. Result of the computer simulation: (a) stored pattern; (b) retrieved pattern. A small part of each pattern is shown. Actually a cyclic spatiotemporal pattern with a period of 500τ and with 1000 components was input.

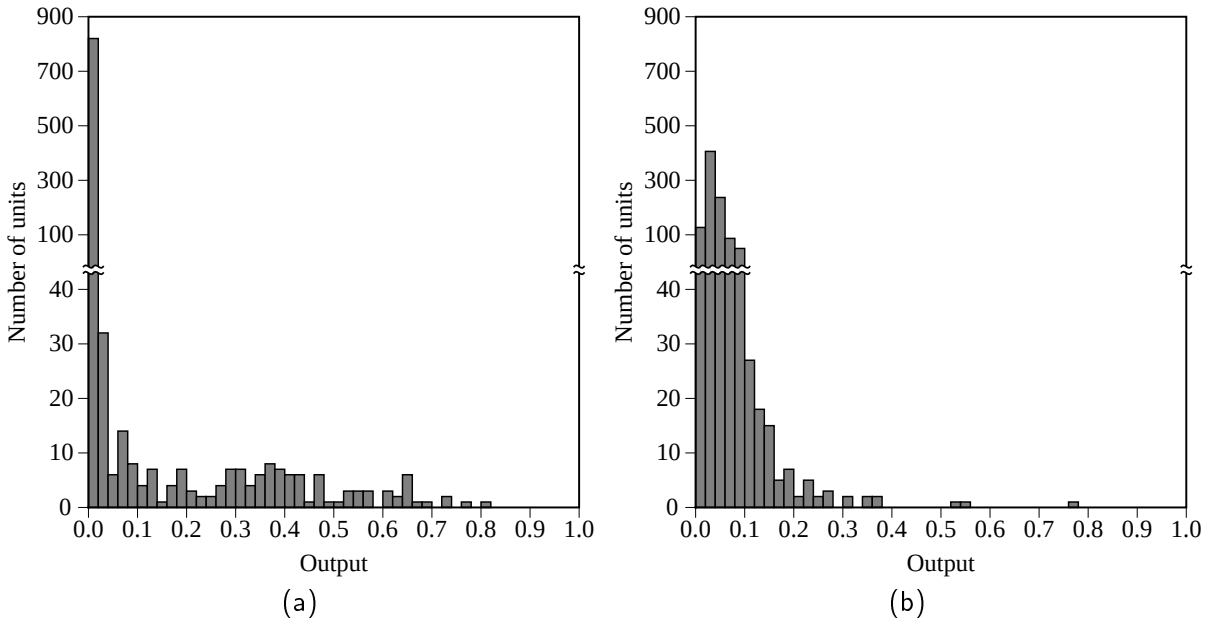

Fig. 11. Distribution of outputs x_i (a) when stored patterns are retrieved and (b) when stored patterns are not retrieved.

0.5τ) and then the external input was cut off. We see that the graph is similar to that in Fig. 10a, which indicates that the input sequence was successfully stored and retrieved.

Figure 11a shows the distribution of outputs at an instant during recall. Each output varies with time, but the form of the histogram is almost constant while the network is recalling the stored sequence correctly. In contrast, the outputs are distributed as in Fig. 11b when a random pattern is given initially and none of the stored patterns are retrieved. The two distributions are obviously different, though their averages are nearly equal.

These histograms are quite similar to those for inferotemporal neurons observed by Miyashita[4], which can hardly be explained by conventional models with monotonic characteristics that generally exhibit a bipolar distribution of outputs. Also, neurons which exhibit increasing or decreasing activities during the delay period of a pair-association task[10] can be explained by this model.

5 Discussion

5.1 Summary of the Model

Before further discussing the biological relevance of this model, I will summarize its distinctive features.

First, the model is based on a principle which can reasonably be applied to the brain as follows:

- The model has the simple structure shown in Fig. 3. No delay or synchronization mechanisms are required.
- Nonmonotone dynamics, which is essential for smooth sequential recall, is realized by introducing local inhibition cells. It should be noted that the composition described in Sec. 4.1 is the simplest solution for achieving nonmonotonic characteristics with monotonic cells.
- Sparse population coding without synchronization is used. It enables the network to store a much larger amount of information and gives higher tolerance to noise than the grandmother-cell coding.
- Storage is performed by a very simple learning algorithm. It is based on a covariance rule and requires only a few (about 3–6) repetitions of input.

Second, the model exhibits high memory performance in many respects (see also [8]). It is worth noting that this model can memorize various kinds of patterns (static patterns, cyclic sequences, and sequences ending with static patterns) in the same way.

Lastly, so far no other principle of sequential pattern memory can be reasonably applied to the brain.

It is therefore probable that basically the same principle as that described above is actually used in cerebral memory systems. Of course, this must be determined by physiological experiment and observation. Nevertheless, assuming that the model does describe the memory principle in the brain, I will discuss the correspondence of the model to the brain in the next section.

5.2 Relevance to the Brain

As was shown in Fig. 3, this model consists of two networks, N_1 and N_2. Since input patterns are finally stored in the network N_1, it is natural to assume that N_1 corresponds to a cortical area for storing long-term memories, probably the premotor or supplementary motor area for motor sequences and some area in the temporal association cortex for episodic memories.

Then what cerebral area does the network N_2 correspond to? Before answering this question, let us review the function of N_2.

The learning signal R is similar to the output X of the network N_1; in particular, when a static pattern is stored, they may be identical. However, for storage of spatiotemporal patterns, R has to be different from X because the difference enables sequential recall. Thus, the learning signal should be generated outside the memory network.

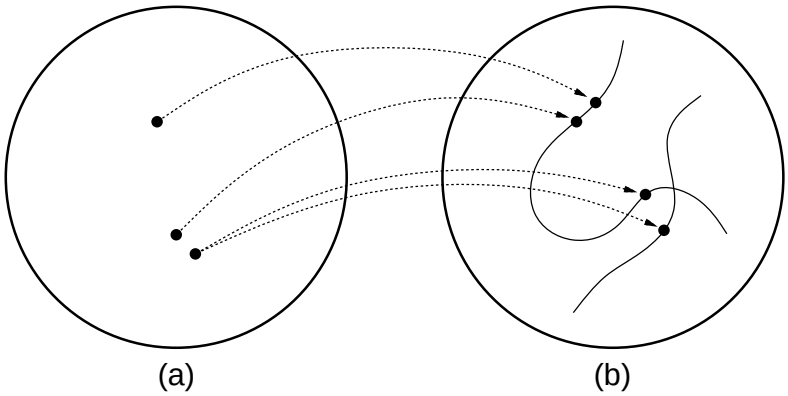

Fig. 12. Transformation of information representation: (a) represents the pattern space of S and (b) represents that of X and R.

We assumed in previous sections that the input pattern can be used as the learning signal. Generally however, this assumption is improper, since storage of sequences usually involves transformation of information representation, as schematically shown in Fig. 12.

In the space of the network state and the learning signal (Fig. 12b), closeness between points, or similarity between patterns, mainly represents closeness in temporal relation. That is, very similar states of N_1 usually code neighboring patterns in a sequence, as was actually observed in the inferotemporal cortex[4].

By contrast, in the space of the input pattern (Fig. 12a), more general features such as shape and motion are represented. That is, two similar patterns code, for example, two figures of similar shape. Accordingly, quite different patterns can appear successively and similar or identical patterns can appear in different parts of sequences.

Such transformation of representation is a hard task, and its specific mechanism remains for future study; however, it is certain that a large network receiving both S and X is necessary for the task.

That is why the model has the part of N_2 in addition to the memory network. In the brain also, such a part that transforms the input pattern into the learning signal must exist. It should be noted that the learning signal and N_2 are not required for retrieval when storage is completed.

Considering these factors, the most probable candidate for N_2 is, in the case of episodic memories, the hippocampus. The hippocampus provides many properties suitable for generating the learning signal (further discussion on this subject will be given at some other time).

In the case of motor sequences, the candidate for N_2 is not clear. Possibly, the function of N_2 is shared among various motor-related areas and representation of motor sequences is transformed step by step in each area into the learning signal; also, the basal ganglia

seem to control the whole process of learning, though this has not been confirmed.

6 Concluding Remarks

The memory mechanism for sequences has been discussed, and a neural network model consisting of pairs of excitatory and inhibitory cells has been presented. This model is insufficient in many respects, especially as to the problem of how the learning signal is generated; however, it is based on a reasonable principle which seems necessary for sequential pattern memory, and thus suggests many things about the neural mechanism in the brain.

I emphasize that no model alone can elucidate the brain; physiological and neuropsychological examination is essential. I hope this study will promote combined theoretical and experimental studies to clarify the mechanism of memory.

References

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

2 Amari, S., Characteristics of sparsely encoded associative memory, *Neural Networks*, 2 (1989) 451–457.

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

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

5 Miyashita, Y. and Chang, H.S., Neuronal correlate of pictorial short-time memory in the primate cortex, *Nature*, 331 (1988) 68–70.

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

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

8 Morita, M., Memory and learning of sequential patterns by nonmonotone neural networks, *Neural Networks*, 9 (1996) (in press).

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

10 Sakai, K. and Miyashita, Y., Neural organization for the long-term memory of paired associates, *Nature*, 354 (1991) 152–155.

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

12 Williams, R.J. and Zipser, D., A learning algorithm for continually running fully recurrent neural networks, *Neural Computation*, 1 (1989) 270–280.

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