ELSEVIER

Cognitive Brain Research 5 (1996) 137–146

COGNITIVE BRAIN RESEARCH

# Computational study on the neural mechanism of sequential pattern memory

Masahiko Morita *

*Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba, Ibaraki 305, Japan*

**Abstract**

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, spatio-temporal 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 non-monotonic 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.

*Keywords:* Sequential pattern memory; Neural network model; Network dynamics; Non-monotonic characteristic; Local inhibition cell; Sparse coding; Learning algorithm; Covariance rule

## 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 memory 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 even do not have 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

* Corresponding author. Fax.: (+81 298) 53-5206; e-mail: mor@is.tsukuba.ac.jp


*PII* S0926-6410(96)00050-X

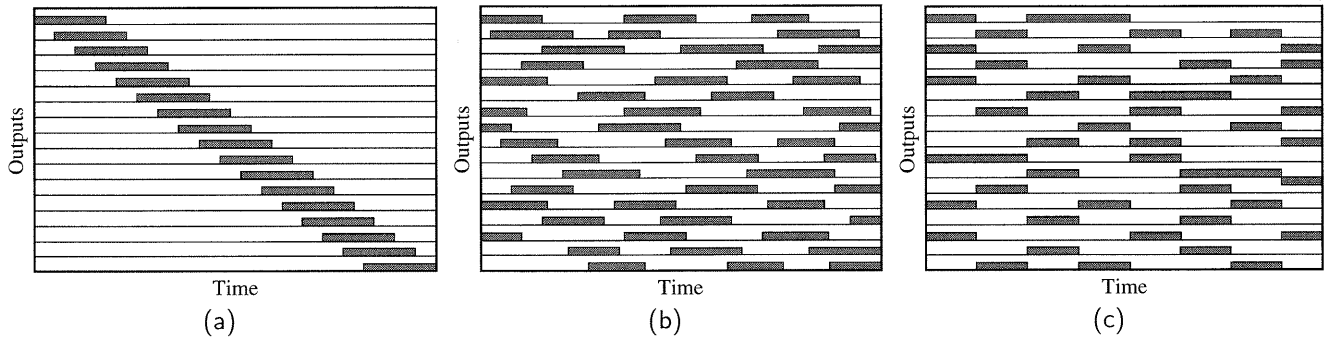


Fig. 1. Types of coding used in neural network models of sequential pattern memory: (a) grandmother-cell coding; (b) population coding; (c) population coding with synchronization.

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 spatio-temporal 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, 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 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 to 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.

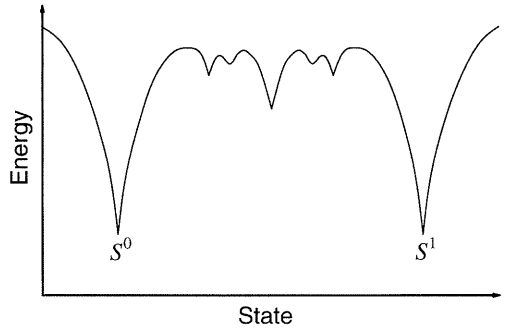


Fig. 2. Schematic energy landscape of conventional neural networks. Two patterns $S^0$ and $S^1$ are stored as attractors. The network state cannot move from $S^0$ to $S^1$ without jumping a distance because the energy landscape has deep valleys at these states.

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 algorithm 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 non-monotonic input-output characteristics was proposed [7]. This model, called a non-monotone neural network, exhibits very high perfor-

mance 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 non-monotone 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, 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 Section 5.2).

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

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

$$y_i = f(u_i), \tag{2}$$

where $u_i$ denotes the instantaneous potential of the $i$th element, $y_i$ the output, $z_i$ the external input, $\tau$ a time constant, and $n$ the number of elements; $f(u)$ is a non-monotonic 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 non-monotonic function. This modification causes a significant change in dynamical properties of the network and enables the network to memorize sequential patterns easily.

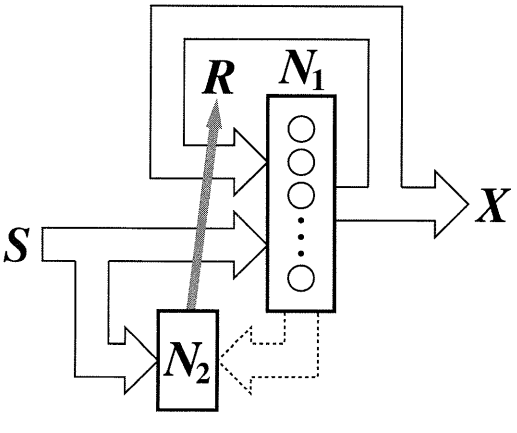


Fig. 3. Structure of the model. $S$ is the input to the model, $R$ is a learning signal, and $X$ is the state (activity pattern) of network $N_1$.

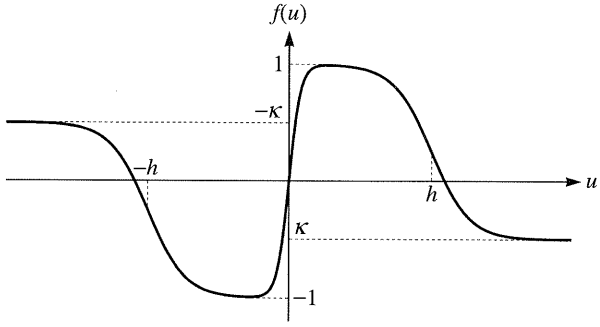


Fig. 4. Input-output characteristics of the non-monotonic element. The detailed form of the function $f(u)$ is not very critical as long as it is non-monotonic and $\kappa \leq 0$. In conventional models, a monotonically increasing function ($\kappa = 1$) is used.

### 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 $Y = (y_1, ..., y_n)$ and the learning signal vector $R = (r_1, ..., r_n)$. Specifically, synaptic weights are modified according to

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

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

The external input vector $Z = (z_1, ..., z_n)$ 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 $\lambda$ 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 non-monotone neural networks, it would be useful for our intuitive understanding to assume a landscape such that $X$ goes downhill.

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 non-monotonic 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 slope of energy (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 and 5d). However, if $R$ moves too fast and the input intensity $\lambda$ 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, 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 en-

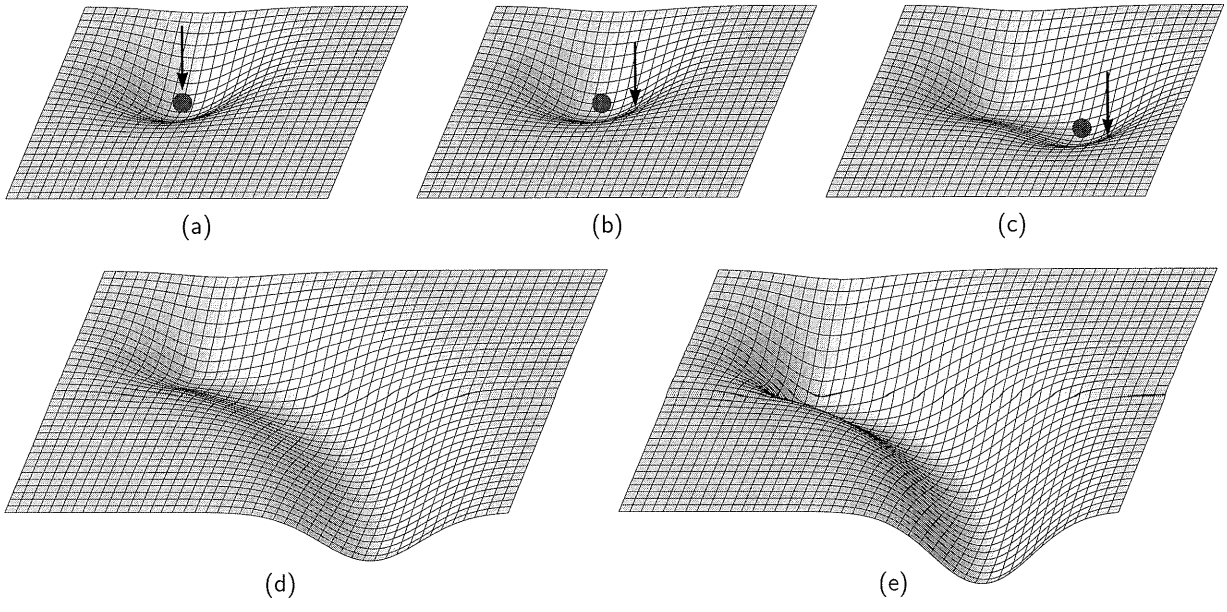


Fig. 5. Illustration of the learning process. Change in the imaginary energy landscape is depicted in a–e. Each point on the surface corresponds to a state or an activity pattern of the network, and the depth represents energy or stability at the state. The current network state $X$ and input pattern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

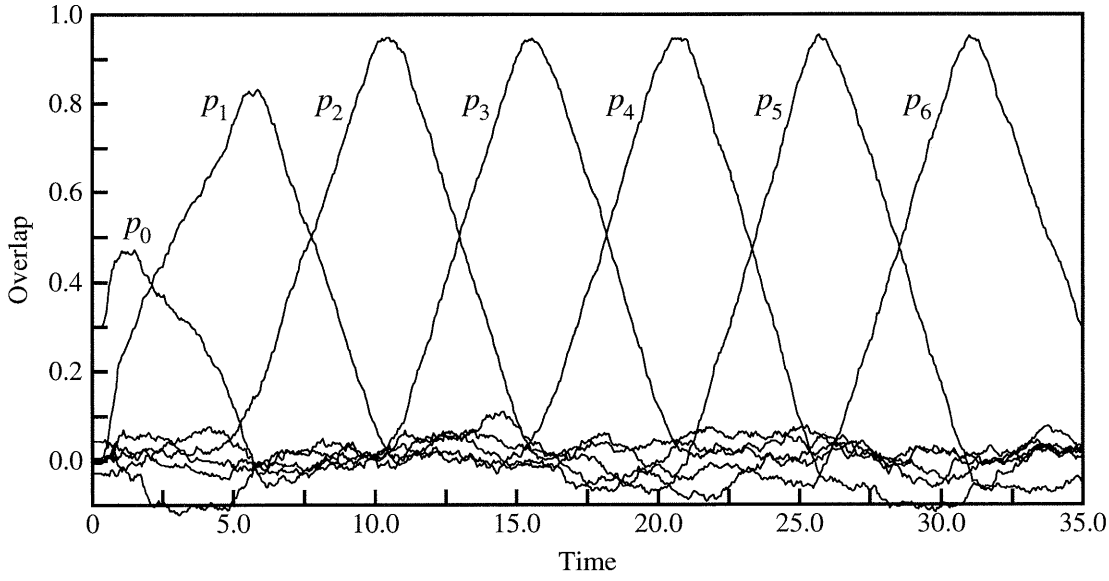


Fig. 6. Recall of the sequence. The time course of the overlaps $p_\mu$ between the network state $X$ and the intermediate patterns $S^\mu$ after learning is plotted. Time is scaled by the time constant $\tau$.

graved to some extent. Accordingly, it is best to decrease $\lambda$ 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 $S^0 \rightarrow S^1 \rightarrow \cdots \rightarrow S^{99} \rightarrow S^0$ was input to the network. Intermediate patterns $S^\mu$ were selected at random, and the learning signal $R$ was varied gradually from $S^\mu$ 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.

Fig. 6 shows the time course of the overlaps $p_\mu$ between the network state $X$ and the intermediate patterns $S^\mu$ 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 $p_\mu$ for $\mu \geq 2$ successively reach a peak value of more than 0.95. This graph indicates that $X$ changes continuously from $S^\mu$ 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.

Fig. 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.

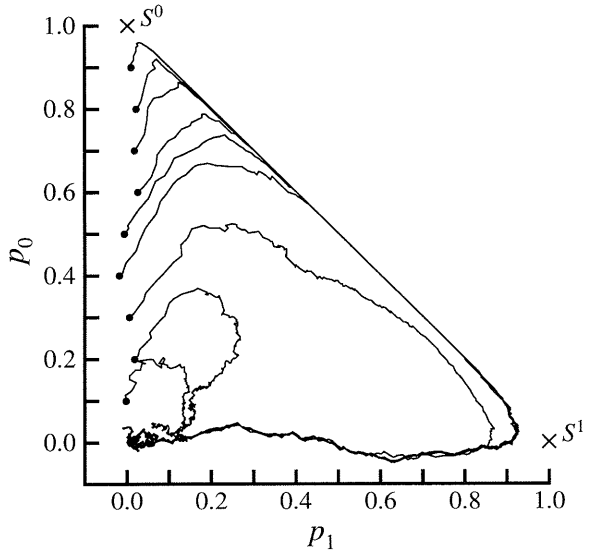


Fig. 7. Retrieval process. Trajectories of $X(t)$ $(0 \leq t \leq 20\ \tau)$ for various initial states (represented by dots) are projected onto the $p_1 - p_0$ plane. Recall is successful when the initial overlap is $p_0 \geq 0.3$.

## 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 non-monotonic 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 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 infero-temporal 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.

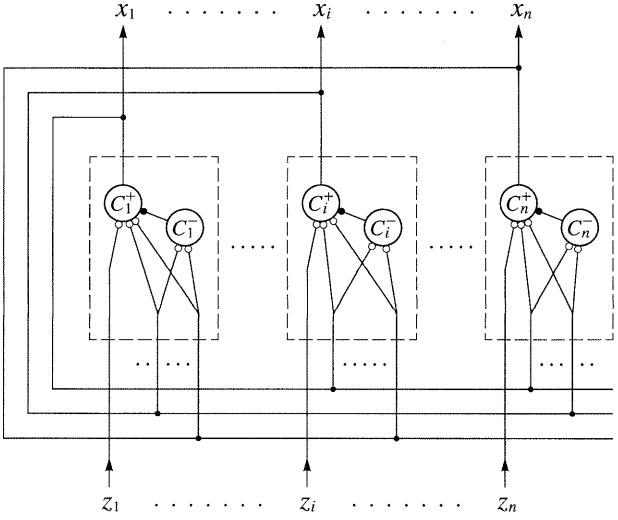


Fig. 8. Structure of the memory network. A pair of excitatory and inhibitory cells compose a unit corresponding to a single element in the previous model.

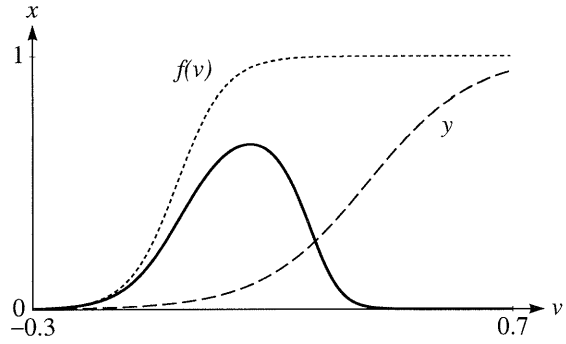


Fig. 9. Input-output characteristics of the unit. Without the inhibition cell, the output $x$ monotonically increases with the total input $v$ (dotted line), but $x$ is a non-monotonic function of $v$ (solid line) because of the output $y$ of the inhibition cell (broken line).

### 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 terms,

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

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

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

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 $\theta$ 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 non-monotonic 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 non-monotonic 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