Recognition of Spatiotemporal Patterns Using a Nonmonotone Neural Network with Hidden Neurons

Satoshi Murakami †, *Masahiko Morita* ††*and Naoto Sakamoto* ††
Email:mare@bcl.esys.tsukuba.ac.jp

†Doctoral Program in Engineering, University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan
††Institute of Information Sciences and Electronics, University of Tsukuba
Tsukuba, Ibaraki 305-8573, Japan

ABSTRACT

It has been shown that a nonmonotone neural network model can recognize spatiotemporal patterns without expanding them into spatial patterns. We improve the recognition ability of this model by introducing hidden neurons. We also show a simple method of training the hidden neurons. Computer simulation shows that this model can recognize complicated spatiotemporal patterns.

KEYWORDS: recognition, spatiotemporal pattern, nonmonotone neural network, hidden neuron.

1. INTRODUCTION

We previously proposed a nonmonotone neural network model which recognizes spatiotemporal patterns without expanding them into spatial patterns [1]. This model has simple structure and learning algorithm and can recognize learned patterns even if they are temporally extended or contracted.

However, this model works inadequately unless the input patterns have simple structures, that is, their trajectories in the pattern space are rather short and not so much intertwined. This is because the input and output patterns cannot be directly associated when they are too much different in structure.

To solve this problem, we introduce hidden neurons to the model. Due to the hidden neurons making an intermediary state transition between the input and output neurons, it is expected that the model can recognize long complicated spatiotemporal patterns.

We also develop a new method of training hidden neurons, since conventional methods such as the backpropagation algorithm are not applicable to this model.

2. PRINCIPLE

2.1 Structure of the Model

The structure of the model is shown in Fig. 1. The network in the upper half learns spatiotemporal patterns and recognizes them, which we call the recognition network. This network has a simple structure composed of n nonmonotone neurons with mutual connections same

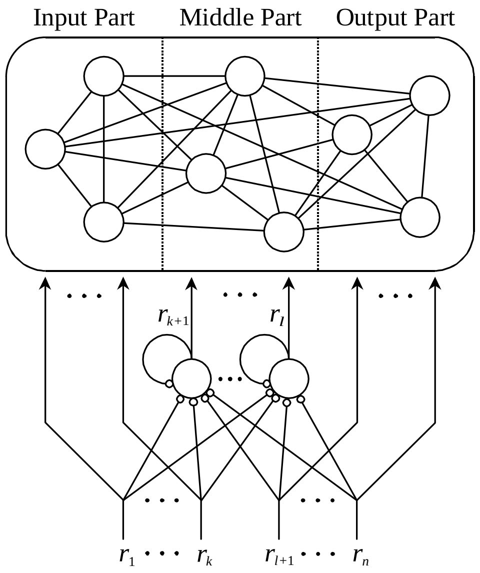

Fig. 1: Structure of the model.

as the previous model [1]. These neurons are divided into three groups, input, middle and output parts, though all the neurons obey the same dynamics and learning rule. The input part receives a spatiotemporal pattern to be recognized, and the output part represents the result of recognition; the other is the middle part.

For convenience, we give serial numbers to the neurons such that neurons 1 to k are the input part, $k+1$ to l are the middle part, and $l+1$ to n are the output part.

The network in the lower half of the figure generates learning signals for the middle part. This network consists of $l-k$ ordinary monotone neurons, each of which corresponds to a hidden neuron. The specific structure will be described in Section 2.4.

2.2 Dynamics of the Recognition Network

Dynamics of the recognition network are expressed by

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

where u_i is the potential of neuron i, and w_{ij} is the synaptic weight from neuron j. z_i is the external input,

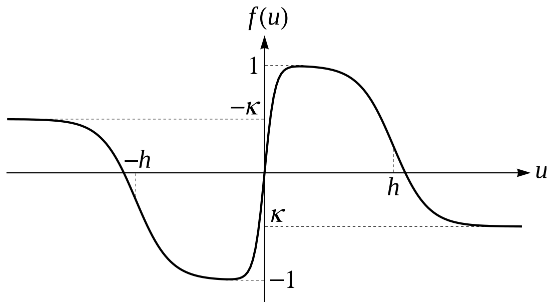

Fig. 2: Nonmonotonic output function.

and τ is a time constant. The output y_i is given by

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

where $f(u)$ is a nonmonotonic function as shown in Fig. 2. We use, as the nonmonotonic output function,

$$f(u) = \frac{1-e^{-cu}}{1+e^{-cu}} \cdot \frac{1+\kappa e^{c'(|u|-h)}}{1+e^{c'(|u|-h)}}, \tag{3}$$

where c, c', h and κ are constants (we substitute $c = 50$, $c' = 10$, $h = 0.5$, $\kappa = -1$ in the experiments described later).

Since the polarity of u_i is important in nonmonotone neural networks, we consider $x_i = \mathrm{sgn}(u_i)$ and treat the vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ as the network state, where $\mathrm{sgn}(u) = 1$ for $u > 0$ and -1 for $u \le 0$.

The network state $\boldsymbol{x}$ at an instant is represented by a point in the state space consisting of 2^n possible states. When $\boldsymbol{x}$ changes, it almost always moves to an adjacent point in the state space because x_i changes asynchronously. Consequently, $\boldsymbol{x}$ leaves a track with time, which we call the trajectory of $\boldsymbol{x}$. Similarly, we call $\boldsymbol{x}_{\mathrm{in}} = (x_1, \ldots, x_k)$, $\boldsymbol{x}_{\mathrm{mid}} = (x_{k+1}, \ldots, x_l)$ and $\boldsymbol{x}_{\mathrm{out}} = (x_{l+1}, \ldots, x_n)$ the states of the input, middle and output parts, respectively, and consider the trajectories of $\boldsymbol{x}_{\mathrm{in}}$, $\boldsymbol{x}_{\mathrm{mid}}$ and $\boldsymbol{x}_{\mathrm{out}}$ in the state space of each part.

2.3 Learning Algorithm

Let $\boldsymbol{s}^1(t), \ldots, \boldsymbol{s}^m(t)$ be m spatiotemporal patterns to be recognized, where $\boldsymbol{s}^\mu = (s_1^\mu, \ldots, s_k^\mu)$. We assume that the elements s_i^μ are ± 1 and change asynchronously. Then we can consider m trajectories corresponding to $\boldsymbol{s}^\mu$ in the k-dimensional pattern space regarded in the same light as the state space of the input part. These trajectories may intersect or overlap with one another.

We perform learning so that the state $\boldsymbol{x}_{\mathrm{out}}$ of the output part becomes a target state $S^\mu = (s_{l+1}^\mu, \ldots, s_n^\mu)$ when the spatiotemporal pattern $\boldsymbol{s}^\mu$ is given to the input part. The learning algorithm is as follows.

First, we create a learning signal vector $\boldsymbol{r} = (r_1, \ldots, r_n)$ with binary elements ($r_i = \pm 1$). The learning signal $\boldsymbol{r}_{\mathrm{in}} = (r_1, \ldots, r_k)$ corresponding to the input part is $\boldsymbol{s}^\mu$, that is, $r_i = s_i^\mu$ for $i \le k$. The learning signal $\boldsymbol{r}_{\mathrm{out}} = (r_{l+1}, \ldots, r_n)$ corresponding to the output part is a spatiotemporal pattern which changes gradually from a static pattern O to S^μ, where we assume $O = (-1, \ldots, -1)$ without losing generality. The learning signal $\boldsymbol{r}_{\mathrm{mid}}$ for the middle part is also a spatiotemporal pattern described later.

Since $\boldsymbol{r}$ is an n-dimensional binary vector, similarly to $\boldsymbol{x}$, $\boldsymbol{r}$ is regarded as moving in the state space of the network from $(\boldsymbol{s}^\mu(0), \boldsymbol{r}_{\mathrm{mid}}^\mu(0), O)$ to $(\boldsymbol{s}^\mu(T), \boldsymbol{r}_{\mathrm{mid}}^\mu(T), S^\mu)$, where T is the temporal length of $\boldsymbol{s}^\mu$.

Next, we give an initial state $\boldsymbol{x} = (\boldsymbol{s}^\mu(0), \boldsymbol{r}_{\mathrm{mid}}^\mu(0), O)$ and input $\boldsymbol{r}$ in the form $z_i = \lambda_i r_i$ to the network while it acts according to Equation 1. Here, λ_i denotes the input intensity of r_i, which is a constant λ_{in} for the input part ($i \le k$) and variables λ_{mid} and λ_{out} decreasing with the process of learning for the middle ($k < i \le l$) and output ($i > l$) parts.

We simultaneously modify all synaptic weights w_{ij} according to

$$\tau' \frac{dw_{ij}}{dt} = -w_{ij} + \alpha\, r_i y_j, \tag{4}$$

where τ' denotes a time constant of learning ($\tau' \gg \tau$) and α is a learning coefficient. Since performance of learning is better when α is a decreasing function of $|u_i|$, we put $\alpha = \alpha' x_i y_i$, where α' is a positive constant.

When $\boldsymbol{r}$ is moving in the state space, $\boldsymbol{x}$ follows slightly behind, roughly along the same trajectory. When $\boldsymbol{r}$ reaches the end, we keep $\boldsymbol{r} = (\boldsymbol{s}^\mu(T), \boldsymbol{r}_{\mathrm{mid}}^\mu(T), S^\mu)$ and continue modifying w_{ij} until $\boldsymbol{x}$ comes close enough to $\boldsymbol{r}$.

We apply this procedure for all μ, and repeat it over some circles, gradually decreasing λ_{mid} and λ_{out}. If $\boldsymbol{x}_{\mathrm{out}}$ can reach a state near S^μ even when $\lambda_{\mathrm{mid}} = \lambda_{\mathrm{out}} = 0$, then the learning is completed.

2.4 Generation of the Learning Signal

The learning signal $\boldsymbol{r}_{\mathrm{mid}}$ for the middle part should satisfy the following conditions: (1) its trajectory is shorter and simpler than that of $\boldsymbol{r}_{\mathrm{in}}$; (2) it reflects the change of $\boldsymbol{r}_{\mathrm{in}}$ to some extent; (3) it starts at a point near $O' = (-1, \ldots, -1)$ which we use as the initial state of the middle part.

These conditions are satisfied if the trajectory of $\boldsymbol{r}_{\mathrm{mid}}$ has a middle property between those of $\boldsymbol{r}_{\mathrm{in}}$ and $\boldsymbol{r}_{\mathrm{out}}$. It is therefore thought that we can generate a desired learning signal $\boldsymbol{r}_{\mathrm{mid}}$ by mixing $\boldsymbol{r}_{\mathrm{in}}$ and $\boldsymbol{r}_{\mathrm{out}}$ using a randomly connected network.

Concretely, we use the network as shown in the lower half of Fig. 1. Each neuron receives $\boldsymbol{r}_{\mathrm{in}}$ and $\boldsymbol{r}_{\mathrm{out}}$ through random synaptic weight a_{ij} and b_{ij}, and outputs the element r_i of $\boldsymbol{r}_{\mathrm{mid}}$ $(k < i \leq l)$. To generate a smooth trajectory without sharp fluctuations, we introduce a self connection of a positive strength ρ to each neuron. In mathematical terms,

$$r_i = \mathrm{sgn}\left(\sum_{j=1}^{k} a_{ij} r_j + \sum_{j=l+1}^{n} b_{ij} r_j + \rho r_i\right), \qquad (5)$$

where $r_i = -1$ for $t < 0$.

The synaptic weights a_{ij} and b_{ij} are randomly determined, but b_{ij} should have a positive average so that $\boldsymbol{r}_{\mathrm{mid}}$ may be near O' at the initial state when $\boldsymbol{r}_{\mathrm{out}} = O$. In the following experiments, b_{ij} are normally distributed random numbers with mean $1/(n-l)$ and variance $1/(n-l)$, a_{ij} are those with mean 0 and variance $1/k$, and $\rho = 1$; these values are determined by several trials.

2.5 Method of Recognition

By the above learning, the trajectories of $\boldsymbol{x}$ become attractors of the dynamical system formed by the recognition network [2]; thus, there exist m trajectory attractors in the state space after learning. Using this, spatiotemporal patterns are recognized as follows.

Let us assume that $\boldsymbol{s}' = (s'_1, \ldots, s'_k)$ is an input pattern made by transforming (e.g., adding noises to) $\boldsymbol{s}^1$ and that s'_i is 1 or -1. We input it to the model in the form $z_i = \lambda_{\mathrm{in}} s'_i$ $(i \leq k)$. To the middle and output parts, we give the initial states O' and O, respectively, and input nothing $(z_i = 0)$ thereafter.

When $\boldsymbol{s}'$ is inputted in this way, $\boldsymbol{x}$ is attracted to the nearest trajectory attractor that is considered to correspond to $\boldsymbol{s}^1$. Consequently, it is expected that the output state $\boldsymbol{x}_{\mathrm{out}}$ becomes nearly equal to S^1 when we finish inputting $\boldsymbol{s}'$, which means the model recognizes the input pattern $\boldsymbol{s}'$ as $\boldsymbol{s}^1$.

3. COMPUTER SIMULATION

To examine the behavior of the model, we carried out computer simulations with 400 input, 600 hidden and 200 output neurons.

We prepared 21 spatiotemporal patterns shown in Table 1. These patterns are formed by connecting 7 static patterns A–G which are 400-dimensional binary vectors selected at random; $\{ABECD\}$ represents the shortest path from A via B, E, C to D, and $\{ABECD\}_T$ is the spatiotemporal pattern whose trajectory is $\{ABECD\}$ and temporal length is T. We set $T = 80\tau$ in the experiment. The target states $S^1, \ldots, S^{21}$ of the output part

Table 1: Spatiotemporal patterns used in the experiment.

$\{ABECD\}_T$	$\{ADACG\}_T$	$\{AEGFA\}_T$
$\{BACAF\}_T$	$\{BEFGF\}_T$	$\{BGCGF\}_T$
$\{CEABD\}_T$	$\{CEACA\}_T$	$\{CGBAB\}_T$
$\{DAFGE\}_T$	$\{DBAFB\}_T$	$\{DCDEC\}_T$
$\{EDGEC\}_T$	$\{EGABD\}_T$	$\{EGEFE\}_T$
$\{FBDAE\}_T$	$\{FBGFC\}_T$	$\{FGFCB\}_T$
$\{GCECF\}_T$	$\{GDADE\}_T$	$\{GFAEF\}_T$

are selected at random; however, for $\boldsymbol{s}^{\mu_1}$ and $\boldsymbol{s}^{\mu_2}$ having the same trajectory in the first ν quarters, we selected S^{μ_1} and S^{μ_2} so that they would have a correlation of $\nu/4$.

After 15 cycles of learning, we inputted various patterns and examined the behavior of the model. The parameters were $\alpha' = 2$, $\tau' = 40000\tau$ and $\lambda_{\mathrm{in}} = 0.2$; λ_{mid} and λ_{out} were decreased by degrees from 0.2 to 0.

3.1 Recognition Process

Figure 3 displays a process of recognition when a spatiotemporal pattern $\{C'E'A'C'A'\}_T$ is inputted with intensity $\boldsymbol{r}_{\mathrm{in}} = 0.2$, where A', C' and E' are vectors containing 50% noise (100 out of 400 elements selected at random are reversed).

Similarities (direction cosines) between $\boldsymbol{x}_{\mathrm{out}}$ and S^μ denoted by $d_{\mathrm{out}}(S^\mu)$ are plotted in the top graph, and those between $\boldsymbol{x}_{\mathrm{in}}$ and A–G denoted by $d_{\mathrm{in}}(A)$–$d_{\mathrm{in}}(G)$ are in the bottom one. The middle graph shows a change of $\boldsymbol{x}_{\mathrm{mid}}$, where similarities between $\boldsymbol{x}_{\mathrm{mid}}(t)$ and $\boldsymbol{r}_{\mathrm{mid}}(t)$ used in learning are plotted. The abscissa is time scaled by the time constant τ.

In the bottom graph, the similarity $d_{\mathrm{in}}(C)$ between $\boldsymbol{x}_{\mathrm{in}}$ and C increases rapidly from the initial value 0.5 to more than 0.8, and then decreases gradually according to the increase of the similarity $d_{\mathrm{in}}(E)$, which indicates that $\boldsymbol{x}_{\mathrm{in}}$ is moving along the path $\{CE\}$. We see that $\boldsymbol{x}_{\mathrm{in}} \simeq A$ at $t = 40\tau$, $\boldsymbol{x}_{\mathrm{in}} \simeq C$ at $t = 60\tau$ and $\boldsymbol{x}_{\mathrm{in}} \simeq A$ at $t > 80\tau$ and that $\boldsymbol{s}^8 = \{CEACA\}_T$ is restored in the input part.

In the middle graph, $\boldsymbol{x}_{\mathrm{mid}}$ moves along a middle trajectory between $\boldsymbol{r}^7_{\mathrm{mid}}$ corresponding to $\boldsymbol{s}^7 = \{CEABD\}_T$ and $\boldsymbol{r}^8_{\mathrm{mid}}$ at $t < 40\tau$. This is because $\boldsymbol{s}^7$ and $\boldsymbol{s}^8$ are the same in their first half and thus $\boldsymbol{r}^7_{\mathrm{mid}}$ and $\boldsymbol{r}^8_{\mathrm{mid}}$ are rather close to each other. Thereafter, $\boldsymbol{x}_{\mathrm{mid}}$ departs from $\boldsymbol{r}^7_{\mathrm{mid}}$ and approaches $\boldsymbol{r}^8_{\mathrm{mid}}$.

In the top graph, we see that $d_{\mathrm{out}}(S^8)$ increases consistently with time, and finally $\boldsymbol{x}_{\mathrm{out}}$ reaches S^8. This indicates that the model has correctly recognized the in-

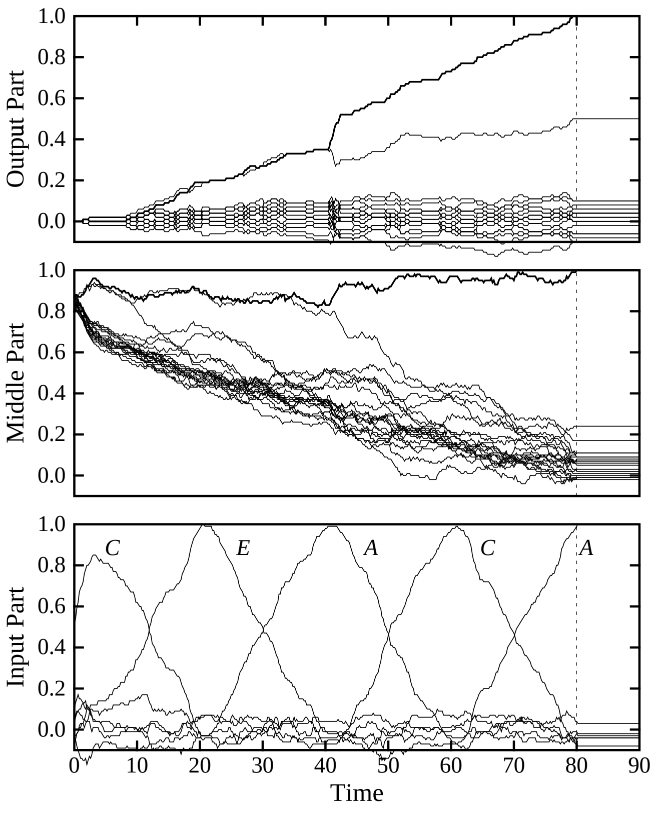

Fig. 3: A process of recognition.

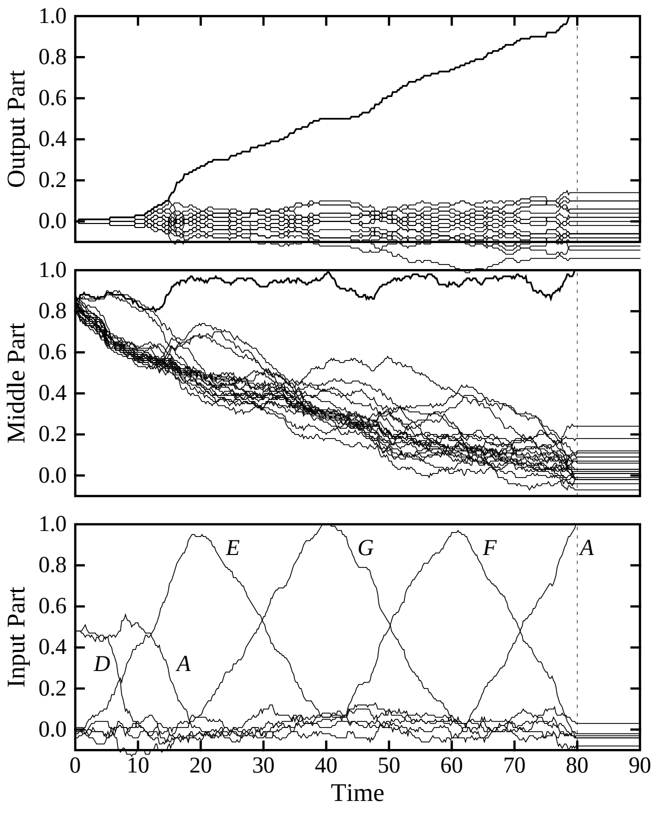

Fig. 4: Behavior when a vague pattern is inputted.

put pattern as $\boldsymbol{s}^8$.

Another example of recognition is shown in Fig. 4, where a vague pattern $\{(AD)E'(AG)F'(AC)\}_T$ is inputted. Here (AD) denotes a vector lying midway between A and D; (AG) and (AC) are also middle vectors.

In the middle graph, $\boldsymbol{x}_{\text{in}}$ lies in between A and D for $t < 10\tau$, when $\boldsymbol{x}_{\text{mid}}$ moves in the middle of trajectories $\boldsymbol{r}^1_{\text{mid}}$, $\boldsymbol{r}^2_{\text{mid}}$ and $\boldsymbol{r}^3_{\text{mid}}$. However, as $\boldsymbol{x}_{\text{in}}$ approaches E, $\boldsymbol{x}_{\text{mid}}$ is attracted to $\boldsymbol{r}^3_{\text{mid}}$ corresponding to $\boldsymbol{s}^3 = \{AEGFA\}_T$, departs from the other $\boldsymbol{r}^{\mu}_{\text{mid}}$ and $\boldsymbol{x}$ moves along the trajectory of $\boldsymbol{r}^3$ thereafter. As a result, $\boldsymbol{x}_{\text{out}}$ reaches S^3, that is, the model recognizes the input pattern as $\boldsymbol{s}^3$. It should be noted that when (AG) and (AC) are inputted, G and A are restored in the input part, respectively, clearly demonstrating an effect of the feedback signals from the middle and output parts.

4. CONCLUDING REMARKS

We have described a nonmonotone neural network model with hidden neurons and proposed a simple method of training the hidden neurons. It has been shown that this model can recognize spatiotemporal patterns even if they are long and complicated or contain much noise.

Moreover, this model does not require special mechanisms such as delay circuits. Accordingly, we think that this model is much more similar to the brain in its working principle than the conventional models of spatiotemporal pattern recognition.

There is much room for further development in this model. For example, this model is thought to be applicable to speech recognition. Theoretical analysis is also a subject for future study.

References

[1] M. Morita and S. Murakami, "Recognition of Spatiotemporal Patterns by Nonmonotone Neural Networks", *Proc. ICONIP'97*, **Vol. 1**, pp.6–9 (1997).

[2] M. Morita, "Memory and learning of sequential patterns by nonmonotone neural networks", *Neural Networks*, **Vol. 9**, 8, pp.1477–1489 (1996).