2005 International Symposium on Nonlinear
Theory and its Applications (NOLTA2005)
Bruges, Belgium, October 18-21, 2005

Hetero Chaotic Associative Memory for Successive Learning with Give Up Function

Makoto IDEGUCHI, Nobuo SATO and Yuko OSANA

School of Computer Science,
Tokyo University of Technology
1404-1 Katakura-cho, Hachioji, Tokyo, 192-0982, Japan
Email: osana@cc.teu.ac.jp

Abstract—In this paper, we propose a Hetero Chaotic Associative Memory for Successive Learning (HCAMSL) with give up function. The proposed model is based on a Chaotic Associative Memory for Successive Memory (CAMSL). In the proposed HCAMSL, the learning process and the recall process are not divided. When an unstored pattern is given to the network, the HCAMSL can learn the pattern successively.

1. Introduction

Recently, neural networks are drawing much attention as a method to realize flexible information processing. Neural networks consider neuron groups of the brain in the creature, and imitate these neurons technologically. Neural networks have some features, especially one of the important features is that the networks can learn to acquire the ability of information processing.

In the filed of neural network, many models have been proposed such as the Back Propagation (BP) algorithm, the Self-Organizing Map (SOM), the Hopfield network, and the Bidirectional Associative Memory (BAM). In these models, the learning process and the recall process are divided, and therefore they need all information to learn in advance.

However, in the real world, it is very difficult to get all information to learn in advance. So we need the model which the learning process and the recall process are not divided. As such model, Grossberg and Carpenter proposed ART (Adaptive Resonance Theory) [1]. However, the ART is based on the local representation, and therefore it is not robust for damage. While in the field of associative memories, some models have been proposed[2]–[4]. Since these models are based on the distributed representation, they have the robustness for damaged neurons. However, they can deal with only auto-associations.

In this paper, we propose a Hetero Chaotic Associative Memory for Successive Learning (HCAMSL) with give up function. The proposed model is based on a Chaotic Associative Memory for Successive Memory (CAMSL)[4] and can deal with hetero-associations. In the proposed HCAMSL, the learning process and the recall process are not divided. When an unstored pattern is given to the network, the HCAMSL can learn the pattern successively.

2. Hetero Chaotic Associative Memory for Successive Learning

2.1. Outline of HCAMSL

Here, we explain the outline of the proposed Hetero Chaotic Associative Memory for Successive Learning (HCAMSL). The proposed HCAMSL has three stages: (1) Pattern Search Stage, (2) Learning Stage and (3) Free Association Stage.

When an unstored pattern set is given to the network, the proposed HCAMSL distinguishes an unstored pattern set from stored patterns and can learn the pattern set successively. When a stored pattern set is given, the HCAMSL recalls the patterns. When an unstored pattern set is given to the network, the HCAMSL changes the internal pattern for the input pattern set by chaos and presents other pattern candidates (we call this the Pattern Search Stage). When the HCAMSL can not recall the desired patterns, it learns the input pattern set as an unstored pattern set (Learning Stage). After the learning, or when an input pattern set is not given, the HCAMSL has free association (Free Association Stage).

2.2. Structure of HCAMSL

The proposed HCAMSL is a kind of the hetero-associative memories. Figure 1 shows a structure of the HCAMSL. This model has two layers; an Input-Output Layer (I/O Layer) composed of conventional neurons and a Distributed Representation Layer (DR Layer) composed of chaotic neurons[5]. In this model, there are the connection weights between neurons in the Distributed Representation Layer and the connection weights between the Input-Output Layer and the Distributed Representation Layer. As shown in Fig.1, the Input-Output Layer has plural parts. The number of parts is decided depending on the number of patterns included in the pattern set. In the case of Fig.1, the Input-Output Layer consists of two parts corresponding to the pattern 1 and the pattern 2.

In this model, when a pattern set is given to the Input-Output Layer, the internal pattern corresponding to the input patterns is formed in the Distributed Representation Layer. Then, in the Input-Output Layer, an output pattern

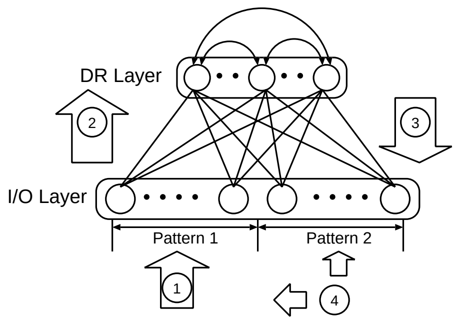

Figure 1: Structure of proposed HCAMSL.

set is generated from the internal pattern. The HCAMSL distinguishes an unstored pattern set from stored patterns by comparing the input patterns with the output patterns.

In this model, the output of the ith neuron in the Distributed Representation Layer at time $t+1$, $x_i^D(t+1)$ is given by the following equations.

$$x_i^D(t+1) = \begin{cases} \phi^D(\xi_i(t+1)), \\ \qquad \text{when a new pattern set is given} \\ \phi^D(\xi_i(t+1) + \eta_i(t+1) + \zeta_i(t+1)), \\ \qquad \text{otherwise} \end{cases} \quad (1)$$

$$\xi_i(t+1) = k_s \xi_i(t) + \sum_{j=1}^{M} v_{ij} A_j(t) \quad (2)$$

$$\eta_i(t+1) = k_m \eta_i(t) + \sum_{j=1}^{N} w_{ij} x_j^D(t) \quad (3)$$

$$\zeta_i(t+1) = k_r \zeta_i(t) - \alpha(t) x_i^D(t) - \theta_i(1-k_r) \quad (4)$$

In Eqs.(1)–(4), M is the number of neurons in the Input-Output Layer, v_{ij} is the connection weight between the jth neuron in the Input-Output Layer and the ith neuron in the Distributed Representation Layer, N is the number of neurons in the Distributed Representation Layer, $A_j(t)$ is the jth external input to the Input-Output Layer at time t, w_{ij} is the connection weight between the ith neuron and the jth neuron in the Distributed Representation Layer, $\alpha(t)$ is the scaling factor of the refractoriness at time t, k_s, k_m, k_r are the damping factors. $\phi^D(\cdot)$ is the following output function:

$$\phi^D(u_i) = \tanh(u_i/\varepsilon) \quad (5)$$

where ε is the steepness parameter.

The output of the jth neuron in the Input-Output Layer at time t, $x_j^{IO}(t)$ is given as follows.

$$x_j^{IO}(t) = \phi^{IO}\left(\sum_{i=1}^{N} v_{ij} x_i^D(t)\right) \quad (6)$$

$$\phi^{IO} = \begin{cases} 1 & , \quad u \geq 0 \\ -1 & , \quad u < 0 \end{cases} \quad (7)$$

2.3. Pattern Search Stage

In the Pattern Search Stage, when an input pattern set is given, the HCAMSL distinguishes the pattern set from stored patterns. When an unstored pattern set is given, the HCAMSL changes the internal pattern for the input pattern by chaos and presents the other pattern candidates. Until the HCAMSL recalls the desired patterns, the following procedures are repeated. If the HCAMSL can not recall the desired patterns, when the stage is repeated certain times, the HCAMSL finishes the stage.

2.3.1. Pattern Assumption

In the proposed HCAMSL, only when the input patterns are given to all parts of the Input-Output Layer, the patterns are judged. When the input pattern $A(t)$ is similar to the recalled pattern $x^{IO}(t)$, the HCAMSL can assume that the input pattern is one of the stored patterns. The HCAMSL outputs the pattern formed by the internal pattern in the Distributed Representation Layer. The similarity $s(t)$ is defined by :

$$s(t) = \frac{1}{M} \sum_{j=1}^{M} g(A_j(t), x_j^{IO}(t)) \quad (8)$$

$$g(a,b) = \begin{cases} 1 & , \quad a = b \\ 0 & , \quad a \neq b. \end{cases} \quad (9)$$

The HCAMSL regards the input patterns as a stored pattern set, when the similarity rate $s(t)$ is larger than the threshold s^{th} $(s(t) \geq s^{th})$.

2.3.2. Pattern Search

When the HCAMSL assumes that the input patterns is an unstored pattern set, the HCAMSL changes the internal pattern $x^D(t)$ for the input patterns by chaos and presents the other pattern candidates.

In the chaotic neural network, it is known that dynamic (chaotic) association can be realized if the scaling factor of the refractoriness $\alpha(t)$ is suitable[5, 6]. Therefore, in the proposed model, $\alpha(t)$ is changed as follows:

$$\alpha(t) = ((\alpha_{max}(t) - \alpha_{min})(1 - s(T)) + \alpha_{min})/\alpha_{DIV} \quad (10)$$

$$\alpha_{max}(t) = M v_{max} + N w_{max} \quad (11)$$

$$w_{max} = \max\{|w_{11}|, \cdots, |w_{ii'}|, \cdots, |w_{NN}|\} \quad (12)$$

$$v_{max} = \max\{|v_{11}|, \cdots, |v_{ij}|, \cdots, |v_{NM}|\} \quad (13)$$

where α_{min} is the minimum of α, $\alpha_{max}(t)$ is the maximum of α at time t, $s(T)$ is the similarity between the input pattern and the output pattern at time T (the time when the Pattern Search Stage was started), α_{DIV} is constant.

2.4. Learning Stage

In the Pattern Search Stage, if the HCAMSL can not recall the desired pattern set, it learns the input pattern set as an unstored pattern set. The Learning Stage has two phases: (1) Hebbian Learning Phase and (2) anti-Hebbian Learning Phase. In the Hebbian Learning Phase, if the signs of the outputs of two neurons are the same, the connection weight between these two neurons is strengthened. By this learning, the connection weights are changed to learn the input patterns, however the Hebbian learning can only learn a new input pattern set. In the proposed HCAMSL, the anti-Hebbian Learning Phase is employed as similar as the original CAMSL[4]. In the anti-Hebbian Learning Phase, the connection weights are changed in the opposite direction in the case of the Hebbian Learning Phase. The HCAMSL can learn a new pattern set without destroying the stored patterns by the anti-Hebbian Learning.

2.4.1. Hebbian Learning Phase

In the Hebbian Learning Phase, until the similarity rate $s(t)$ becomes 1.0, the following procedures ((a) and (b)) are repeated.

(a) Update of Connection Weights

The connection weight between the Input-Output Layer and the Distributed Representation Layer v_{ij} and the connection weight in the Distributed Representation Layer $w_{ii'}$ are updated as follows :

$$v_{ij}^{(new)} = v_{ij}^{(old)} + \gamma_v^+ x_i^D(t) A_j(t) \tag{14}$$

$$w_{ii'}^{(new)} = w_{ii'}^{(old)} + \gamma_w^+ x_i^D(t) x_{i'}^D(t) \tag{15}$$

where γ_v^+ is the learning rate of the connection weight v_{ij} in the Hebbian Learning Phase, and γ_w^+ is the learning rate of the connection wright $w_{ii'}$ in this phase.

(b) Normalization of Connection Weights

When the number of the learnings in this phase becomes more than the threshold n^{th} and the similarity rate $s(t)$ is not equal to 1.0, the connection weights are normalized.

$$v_{ij}^{(new)} = \frac{v_{ij}^{(old)}}{v_{max}} \tag{16}$$

$$w_{ii'}^{(new)} = \frac{w_{ii'}^{(old)}}{w_{max}} \tag{17}$$

where v_{max} is the maximum absolute value of the connection weights v_{ij}, and w_{max} is the maximum absolute value of the connection weight $w_{ii'}$.

(c) Give Up Function

When the similarity rate $s(t)$ does not become 1.0 even if connection weights are normalized r^{th} times, the HCAMSL gives up to study the patterns. If the HCAMSL gives up to study the patterns, the anti-Hebbian Learning Phase is not performed.

2.4.2. Anti-Hebbian Learning Phase

The anti-Hebbian Learning Phase is performed after the Hebbian Learning Phase. In this phase, the connection weights v_{ij} and $w_{ii'}$ are changed in the opposite direction in the case of the Hebbian Learning Phase. The anti-Hebbian Learning makes the relation between the patterns are learned without destroying the stored patterns.

In this phase, v_{ij} and $w_{ii'}$ are updated by

$$v_{ij}^{(new)} = v_{ij}^{(old)} - \gamma_v^- x_i^D(t) A_j(t) \tag{18}$$

$$w_{ii'}^{(new)} = w_{ii'}^{(old)} - \gamma_w^- x_i^D(t) x_{i'}^D(t) \tag{19}$$

where $\gamma_v^-(\gamma_v^- > \gamma_v^+ > 0)$ is the learning rate of the connection weight v_{ij} in the anti-Hebbian Learning Phase, and $\gamma_w^-(\gamma_w^- > \gamma_w^+ > 0)$ is the learning rate of the connection weight $w_{ii'}$ in this phase.

2.5. Free Association Stage

After the Learning Stage, or when a pattern set is not given to the network, the HCAMSL associates freely. In this stage, distinction between unstored and stored patterns is not carried.

3. Computer Experiment Result

In this section, we show the computer experiment results to demonstrate the effectiveness of the proposed HCAMSL.

3.1. Action Study of Robot using HCAMSL

In this section, we applied the proposed HCAMSL to the action study of a robot. Here, we made the robot (Fig.2) by using LEGO Mindstorms and the relation between the instruction and the action were trained by the proposed HCAMSL (Fig.3).

In this experiment, the input such as the instruction and the evaluation for robot's action are given by voice, and they are recognized by the speech recognition program. The recognized instruction information is sent to the HCAMSL, and the HCAMSL output the action of the robot. The robot acts according to the output of the HCAMSL. The relation between the instruction and the action is learned based on the evaluation which is given to the robot's action.

We carried out some trials and confirmed that the proposed model can memorize the relation between the instruction and the action.

3.2. Storage Capacity

Here, we examined the storage capacity of the proposed HCAMSL. In this experiment, we used the HCAMSL which has 100 neurons (50 neurons for pattern 1 and 50 neurons for pattern 2) in the Input-Output Layer and the

Figure 2: Mindstorms Robot.

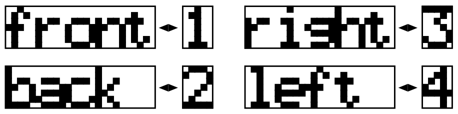

Figure 3: An Example of Stored Pattern.

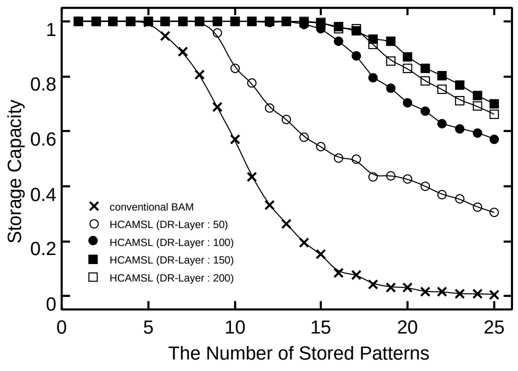

Figure 4: Storage Capacity (I/O Layer : 100).

HCAMSL which has 200 neurons (100 neurons for pattern 1 and 100 neurons for pattern 2) in the Input-Output Layer. We used random patterns to store and Figs.4 and 5 show the average of 100 trials. In these figures, the horizontal axis is the number of stored pattern set, and the vertical axis is the perfect recall rate. As shown in Figs.4 and 5, the storage capacity of the proposed HCAMSL is higher than that of the Bidirectional Associative Memory (BAM)[7]. Moreover, although the conventional BAM can not memorize any pattern when the number of patterns to be stored is large, the proposed HCAMSL can memorize some patterns.

Figure 6 shows the storage capacity of the proposed HCAMSL and the model without give up function. This figure shows that the effectiveness of the give up function in the proposed HCAMSL.

4. Conclusions

In this paper, we have proposed the Hetero Chaotic Associative Memory for Successive Learning (HCAMSL) with give up function and applied it to action study of a robot. The proposed model is based on a Chaotic Associative Memory for Successive Memory (CAMSL). In the proposed HCAMSL, the learning process and the recall process are not divided. When an unstored pattern is given to the network, the HCAMSL can learn the pattern successively. We carried out a series of experiment and confirmed that the proposed HCAMSL can memorize pattern sets successively.

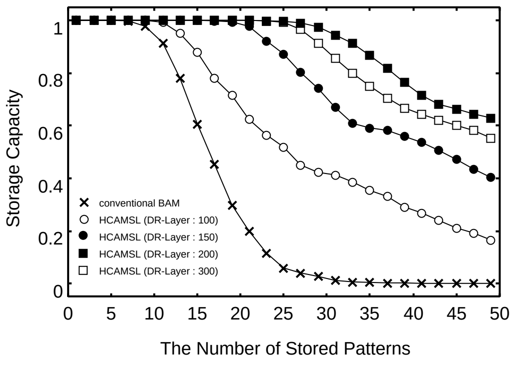

Figure 5: Storage Capacity (I/O Layer : 200).

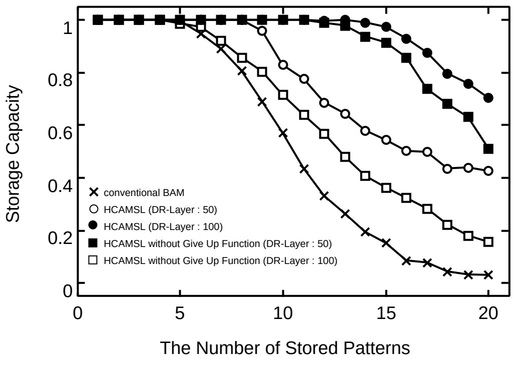

Figure 6: Effectiveness of Give Up Function.

References

[1] G. A. Carpenter, S. Grossberg, Pattern Recognition by Self-organizing Neural Networks, The MIT Press, 1995.

[2] M. Watanabe, K. Aihara, S. Kondo, "Automatic learning in chaotic neural network s," IEICE-A, Vol.J78-A, No.6, pp.686–691, 1995 (in Japanese).

[3] Y. Osana and M. Hagiwara, "Successive learning in chaotic neural network," International Journal of Neural Systems, Vol.9, No.4, pp.285–299, 1999.

[4] N. Kawasaki, Y. Osana and M. Hagiwara, "Chaotic associative memory for successi ve learning using internal patterns," IEEE International Conference on Systems, Man and Cybernetics, 2000.

[5] K. Aihara, T. Takabe and M. Toyoda, "Chaotic neural networks," Physics Letters A, **144**, No. 6,7, pp. 333–340, 1990.

[6] Y. Osana and M. Hagiwara, "Separation of superimposed pattern an d many-to-many associations by chaotic neural networks," IJCNN, Anchorage, **1**, pp. 514–519, 1998.

[7] B. Kosko, "Bidirectional associative memories," IEEE Trans. Syst. Man. Cybrn., **SMC-18**, No. 1, pp. 49–60, 1988.