Growing Greedy Search and Its Application to Hysteresis Neural Networks

Kei Yamaoka and Toshimichi Saito(✉)

Hosei University, Koganei, Tokyo 184-8584, Japan
tsaito@hosei.ac.jp

Abstract. This paper presents the growing greedy search algorithm and its application to associative memories of hysteresis neural networks in which storage of desired memories are guaranteed. In the algorithm, individuals correspond to cross-connection parameters, the cost function evaluates the number of spurious memories, and the set of individuals can grow depending on the global best. Performing basic numerical experiments, the algorithm efficiency is investigated.

Keywords: Greedy search · Hysteresis neural nets · Associative memories

1 Introduction

The hysteresis neural network (HNN [1]) is a continuous-time recurent-type network characterized by binary hysteresis activation function and ternary connection parameters. Depending on parameters, the HNN can exhibits various phenomena: co-existing equilibrium points, synchronization, chaos, and bifurcation [2]. The dynamics is described by a piecewise linear differential equation and he phenomena can be analyzed precisely. The HNN has been applied to associative memories, analog-to-digital converters, and combinatorial optimization problem solvers [1–5]. In the associative memories, we have several theoretical results for storage and stability of desired memories [1]. However, it is hard to suppress spurious memories.

This paper presents a simple evolutionary algorithm based on the greedy search [6,7] and applies it to the suppression of the spurious memories. In the greedy based algorithm, individuals correspond to cross-connection parameters of the HNN and the cost function evaluates the number of spurious memories. The initial individual is given by the correlation based learning [1] that guarantees storage of desired memories. Bit-inversion and elite strategy are applied and the results are evaluated. Depending on the evaluation, the number of individuals can increase. This growing structure is the major difference from the classic greedy search. We refer to this algorithm as growing greedy search (GGS). Performing numerical experiments for basic examples of associative memories,

T. Saito—This work is supported in part by JSPS KAKENHI#15K00350.

we have confirmed that the GGS operates effective to suppress the spurious memories. Note that this is the first paper of the GGS. Although the GGS is applied to the HNN in this paper, the GGS will be applied to various systems [8–10].

2 Hysteresis Neural Networks

The dynamics of hysteresis neural network (HNN) is described by

$$\begin{aligned} &\dot{x}_i = -x_i + \sum_{l=1}^{N} w_{ij} y_i + d_i \equiv -(x_i - p_i),\ i = 1 \sim N \\ &y_i = h(x_i) = \begin{cases} +1 \text{ for } x_i > -Th \\ -1 \text{ for } x_i < Th \end{cases} \end{aligned} \tag{1}$$

where $x \equiv (x_1, \cdots, x_N)$ is the inner state vector and $y \equiv (y_1, \cdots, y_N)$, $y_i \in \{-1, 1\}$, is the binary output vector. $h(x)$ is the hysteresis activation function as shown in Fig. 1: $h(x)$ is switched from -1 to 1 (respectively, 1 to -1) if x reaches the right threshold $Th > 0$ (respectively, the left threshold $-Th$). w_{ij} is connection parameters and d_i is offset parameters. For simplicity, we assume that the cross connection parameters are ternary and symmetric: $w_{ij} = w_{ji} \in \{-1, 0, 1\}$. We also assume that $d_i = 0$ and $Th = 0$. An equilibrium point of the HNN is given by

$$p(y) \equiv (p_1, \cdots, p_N),\ p_i = \sum_{l=1}^{N} w_{ij} y_i \tag{2}$$

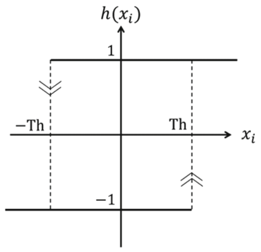

Fig. 1. Binary hysteresis activation function

For stability of the equilibrium point $p(y)$, we have

$$\begin{aligned} &p(y) \text{ is stable} \quad\ \text{if } p_i y_i > -Th \text{ for any } i \\ &p(y) \text{ is unstable if } p_i y_i \le -Th \text{ for some } i \end{aligned} \tag{3}$$

In the case where the HNN has equilibrium point, almost all solutions converges to either equilibrium point if

$$w_{ii} + Th > 0 \text{ for } i = 1 \sim N \tag{4}$$

Proofs of Eqs. (2)–(4) can be found in [1]. Hereafter, we consider an application to the associative memory, First, we define the desired memories

$$s^1, \cdots, s^M,\ s^l \equiv (s_1^l, \cdots, s_N^l),\ l = 1 \sim M$$

If we can determine parameters such that the equilibrium point of some desired memory is stable, the desired memory is said to be stored into the HNN. Storage of all the desired memories is guaranteed in the CL-based learning defined as the following [1]. First, the cross connection parameters are given by ternalizeing correlation matrix:

$$w_{ij} = \begin{cases} -1 \text{ for } c_{ij} \le 0 \\ +1 \text{ for } c_{ij} > 0 \end{cases},\ c_{ij} = \sum_{l=1}^{M} s_i^l s_j^l, \quad i \neq j \tag{5}$$

The auto connection parameter w_{ii} is goven by the minimum integer that satisfies

$$0 < w_{ii} + Th + Q_i,\ Q_i = \min_k \sum_{i \neq j} w_{ij} s_i^k s_j^k \tag{6}$$

If equilibrium point of some output vector is stable and is not any desired memory, the output vector is said to be a spurious memory. Even if all the desired memories are stored, there usually exist spurious memories. Suppression of the spurious memories is an important problem in the associative memories.

3 Growing Greedy Search Algorithm

We present a novel evolutionary algorithm: the growing greedy search method (GGS). First, we give basic definitions. Let t denote evolution steps and let $P^i(t) \equiv (p_1^i(t), \cdots p_M^i(t))$ be the i-th individual at step t where M is the dimension of the individual and $i = 1 \sim k(t)$. Note that the number of individuals $k(t)$ can vary. An individual corresponds to half of the cross-connection parameters:

$$P^i(t) \equiv \{w_{ij}\},\ i > j,\ i,j = 1 \sim N$$

where $M = N(N-1)/2$. The other half cross-connection parameters are given by the symmetricity, $w_{ji} = w_{ij}$. The cost function is defined by

$$F(P^i(t)) = \text{ the number of spurious memories}$$

Substituting $P^i(t)$ into w_{ij} and substituting the output vectors into Eqs. (2)–(4), this cost function can be calculated where w_{ii} is given by Eq. (6). The GGS consists of the following two sub-programs.

3.1 GGS1: Bit Inversion

Step 1: Initialization. Let $t = 0$ and let $k(t) = 1$. The initial individual $P^i(t)$, $i = 1 \sim k(t)$ is given by the CL-based learning of Eq. (5). The cost function is also initialized, $F(P^i(t))$. Let Gb denote the global best and let $Gb(t) = F(P^i(t))$

Step 2: One bit inversion. For $P^i(t)$, $i = 1 \sim k(t)$, one of the elements is inversed. After the inversion, we obtain $N \times k(t)$ individuals $Q^j(t)$, $j = 1 \sim N \times k(t)$.

Step 3: Evaluation. $\{Q^j(t)\}$ are evaluated by the cost function F. Let $F(Q_b(t))$ be the best value of $F(Q^j(t))$ for $j = 1 \sim N \times k(t)$. The global best is updated:

$$Gb(t) \leftarrow \begin{cases} F(Q^j(t)) & \text{if } F(Q_b(t)) < Gb(t) \\ Gb(t) & \text{otherwise} \end{cases} \tag{7}$$

If plural individuals $Q^j(t)$ have the same best value of F, then the individuals are preserved in the next step and the number of individuals varies. If the number of the best individuals exceeds the number limit M_a then M_a of the best individuals are selected randomly. For example, if $Q^1(t)$, $Q^7(t)$, and $Q^9(t)$ have the same best value, these three individuals are preserved as $P^1(t)$, $P^2(t)$, and $P^3(t)$, respectively; and let $k(t) \leftarrow 3$.

Step 4: Update check of Gb. If the $Gb(t)$ has not been updated M_b times then the GGS1 is terminated and is switched to GGS2.

Step 5: Let $t \leftarrow t + 1$, go to Step 2, and repeat until the maximum time limit t_{max1}.

3.2 GGS2: Zero Insertion

Step 1: Initialization. Let $t \leftarrow t_1 + 1$ where t_1 is the last step in GGS1. Let $k(t) = 0$. $P^i(t)$, $i = 1 \sim k(t)$ is given by the best individuals after GGS1. If plural best individuals exist, one of them is selected randomly. The cost function and Gb are initialized by the $P^i(t)$.

Step 2: Zero insertion. For $P^i(t)$, $i = 1 \sim k(t)$, zero is inserted into one of the elements. After the insertion, we obtain $N \times k(t)$ individuals $R^j(t)$, $j = 1 \sim N \times k(t)$.

Step 3: Evaluation. $\{R^j(t)\}$ are evaluated by the cost function F. Let $F(R_b(t))$ be the best value of $F(R^j(t))$ for $j = 1 \sim N \times k(t)$. The global best is updated:

$$Gb(t) \leftarrow \begin{cases} F(R^j(t)) & \text{if } F(R_b(t)) < Gb(t) \\ Gb(t) & \text{otherwise} \end{cases} \tag{8}$$

If plural individuals $Q^j(t)$ have the same best value of F, then the individuals are preserved in the next step and the number of individuals varies. If the number of the best individuals exceeds the number limit M_a then M_a of the best individuals are selected randomly.

Step 4: Update check of Gb. If the $Gb(t)$ has not been updated M_b times then the search is terminated.

Step 5: Let $t \leftarrow t + 1$, go to Step 2, and repeat until the maximum time limit t_{max2}.

Table 1. Parameters after the CL-based learning (1)

j	1	2	3	4	5	6	7	8
w_{1j}	+3	−1	−1	−1	+1	+1	+1	−1
w_{2j}	−1	+1	−1	−1	−1	−1	−1	+1
w_{3j}	+1	−1	+3	−1	−1	−1	+1	−1
w_{4j}	−1	−1	−1	+1	−1	−1	−1	+1
w_{5j}	+1	−1	−1	−1	+3	+1	−1	−1
w_{6j}	+1	−1	−1	−1	+1	+1	−1	+1
w_{7j}	+1	−1	+1	−1	−1	−1	+3	−1
w_{8j}	−1	+1	−1	+1	−1	+1	−1	+1

Table 2. Parameters after the CL-based learning (2)

j	1	2	3	4	5	6	7	8
w_{1j}	+1	+1	+1	−1	−1	−1	+1	−1
w_{2j}	+1	+3	+1	−1	−1	−1	−1	−1
w_{3j}	+1	+1	+1	+1	−1	−1	−1	−1
w_{4j}	−1	−1	+1	+1	+1	+1	−1	−1
w_{5j}	−1	−1	−1	+1	+3	+1	−1	−1
w_{6j}	−1	−1	−1	+1	+1	+1	−1	+1
w_{7j}	−1	−1	−1	−1	−1	−1	+3	+1
w_{8j}	+1	−1	−1	−1	−1	+1	+1	+3

Table 3. Parameters after GGS1 (1)

j	1	2	3	4	5	6	7	8
w_{1j}	+4	+1	+1	−1	−1	−1	−1	−1
w_{2j}	+1	+6	+1	−1	−1	−1	+1	−1
w_{3j}	+1	+1	+2	+1	−1	−1	−1	−1
w_{4j}	−1	−1	+1	0	−1	−1	−1	+1
w_{5j}	−1	−1	−1	−1	0	+1	−1	−1
w_{6j}	−1	−1	−1	−1	+1	0	−1	+1
w_{7j}	−1	−1	−1	−1	+1	−1	+4	−1
w_{8j}	−1	−1	−1	+1	−1	+1	−1	0

Table 4. Parameters after GGS1 (2)

j	1	2	3	4	5	6	7	8
w_{1j}	0	+1	+1	−1	−1	+1	+1	−1
w_{2j}	+1	+2	−1	−1	−1	−1	−1	−1
w_{3j}	+1	−1	0	+1	−1	−1	−1	−1
w_{4j}	−1	−1	+1	0	−1	+1	−1	−1
w_{5j}	−1	−1	−1	−1	+2	+1	−1	−1
w_{6j}	+1	−1	−1	+1	+1	0	−1	+1
w_{7j}	+1	−1	−1	−1	−1	−1	+4	+1
w_{8j}	−1	−1	−1	−1	−1	+1	+1	+4

Table 5. Parameters after GGS2 (1)

j	1	2	3	4	5	6	7	8
w_{1j}	+3	0	+1	0	−1	−1	0	−1
w_{2j}	0	+4	0	−1	−1	−1	+1	−1
w_{3j}	+1	0	+2	0	−1	−1	+1	−1
w_{4j}	0	−1	0	0	−1	−1	−1	+1
w_{5j}	−1	−1	−1	−1	+1	+1	0	−1
w_{6j}	−1	−1	−1	−1	+1	0	−1	+1
w_{7j}	0	+1	+1	−1	0	−1	+2	−1
w_{8j}	−1	−1	−1	+1	−1	+1	+1	0

Table 6. Parameters after GGS2 (2)

j	1	2	3	4	5	6	7	8
w_{1j}	+1	+1	0	−1	−1	0	+1	0
w_{2j}	+1	+1	0	0	−1	−1	−1	0
w_{3j}	0	0	+1	+1	0	−1	−1	−1
w_{4j}	−1	0	+1	+1	0	0	−1	−1
w_{5j}	−1	−1	0	0	+1	+1	0	−1
w_{6j}	0	−1	−1	0	+1	+1	0	+1
w_{7j}	+1	−1	−1	−1	0	0	+2	+1
w_{8j}	0	0	−1	−1	−1	+1	+1	+2

4 Numerical Experiments

In order to investigate the algorithm efficiency, we have performed basic numerical experiments. We consider the following two examples of 14 desired memories. Example 1 (s^8–s^{14} are inverse patterns of s^1–s^7, respectively.)

$$\begin{aligned}
s^1 &\equiv (-1,1,-1,-1,1,1,-1,1) = -s^8\\
s^2 &\equiv (1,-1,1,-1,1,-1,1,-1) = -s^9\\
s^3 &\equiv (1,1,-1,-1,1,1,-1,-1) = -s^{10}\\
s^4 &\equiv (1,1,1,-1,-1,-1,1,-1) = -s^{11}\\
s^5 &\equiv (-1,1,1,-1,1,1,-1,-1) = -s^{12}\\
s^6 &\equiv (1,-1,1,-1,1,1,-1,-1) = -s^{13}\\
s^7 &\equiv (-1,1,1,1,-1,-1,-1,1) = -s^{14}
\end{aligned}$$

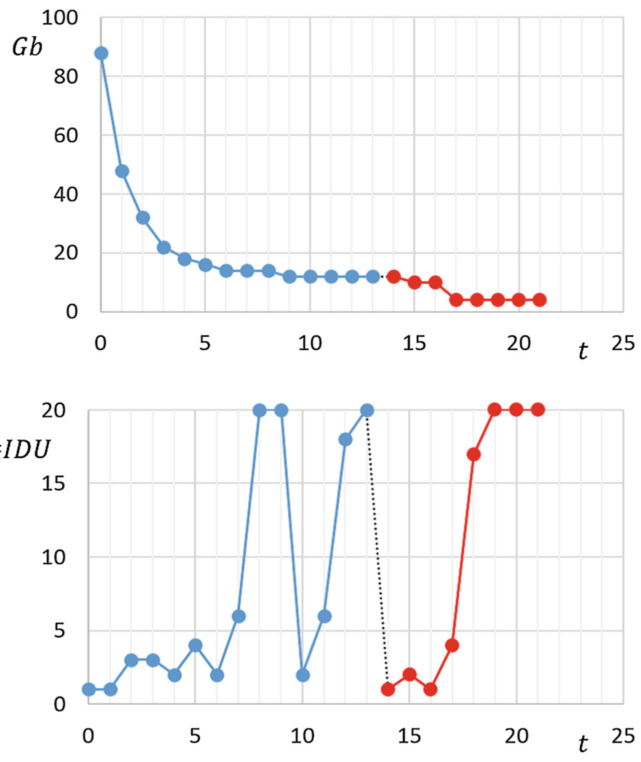

Fig. 2. Evolution process of Gb and #IDU (1)

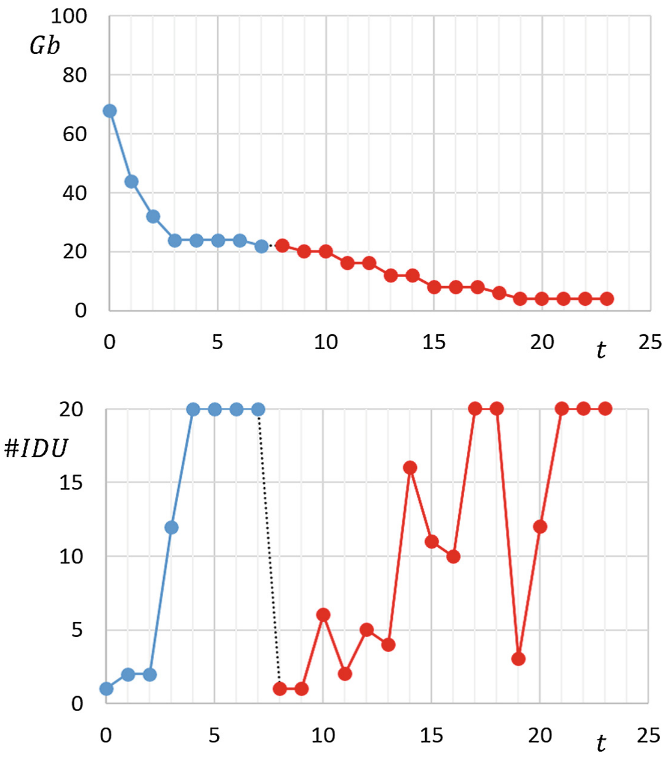

Fig. 3. Evolution process of Gb and #IDU (2)

Example 2 (s^8–s^{14} are inverse patterns of s^1–s^7, respectively.)

$$\begin{aligned}
s^1 &\equiv (-1,-1,-1,-1,1,1,1,1) = -s^8\\
s^2 &\equiv (1,-1,-1,-1,-1,1,1,1) = -s^9\\
s^3 &\equiv (1,1,-1,-1,-1,-1,1,1) = -s^{10}\\
s^4 &\equiv (-1,-1,1,1,-1,-1,1,1) = -s^{11}\\
s^5 &\equiv (-1,1,-1,1,-1,1,-1,1) = -s^{12}\\
s^6 &\equiv (-1,-1,-1,1,1,1,-1,1) = -s^{13}\\
s^7 &\equiv (-1,-1,-1,1,1,1,1,-1) = -s^{14}
\end{aligned}$$

Applying the CL-based learning, these desired memories can be stored into 8-dimensional HNN ($N = 8$). Tables 1 and 2 show parameters after the CL-based learning. However, the Examples 1 and 2 have 88 and 68 spurious memories, respectively. In order to suppress the spurious memories, we have applied the GGS1 (bit-inversion) and GGS2 (zero insertion). After trial-and-errors, the algorithm parameters are set as the following.

$$t_{max1} = t_{max2} = 25, M_a = 20, M_b = 5$$

The GGS1 and GGS2 can remuve the spurious memories. Tables 3 and 4 show parameters values after the GGS1. The Examples 1 and 2 have 12 and 22 spurious memories, respectively. Tables 5 and 6 show parameters values after the GGS2. The Examples 1 and 2 have 4 and 6 spurious memories, respectively.

Figure 2 shows Gb and #IDU (the number of individuals) in the evolution process of Example 1. In the figure, we can see that the GGS1 removes spurious memories sufficiently. #IDU varies widely and the variation seems to be helpful for the effective evolution. At $t = 13$, Gb improvement is stagnated and GGS1 is switched to GGS2. The GGS2 removes a few spurious memories and is not so effective. At $t = 21$, the Gb improvement is stagnated and GGS2 is terminated.

Figure 3 shows Gb and #IDU in the evolution process of Example 2. In the figure, we can see that the GGS1 cannot removes spurious memories sufficiently. At $t = 7$, Gb improvement is stagnated and GGS1 is switched to GGS2. The GGS2 works effectively to further removal of spurious memories. #IDU varies and the variation seems to be effective. At $t = 22$, the Gb improvement is stagnated and GGS2 is terminated.

5 Conclusions

A simple evolutionary algorithm, GGS (GGS1 and GGS2) is presented in this paper. The GGS1 and GGS2 operate based on the bit-inversion and zero-insertion, respectively. The individuals is evaluated by a cost function and the set of individuals can grow depending on the evaluation by the cost function. The algorithm is applied to the suppression problem of spurious memories in hysteresis neural networks. In basic experiments. the algorithm efficiency is confirmed.

Future problems include analysis of search process, effective setting of the algorithm parameters, and application to various discrete problems.

References

1. Jin'no, K., Saito, T.: Analysis and synthesis of a continuous-time hysteresis neural network. In: Proceedings of IEEE/ISCAS, pp. 471–474 (1992)
2. Jin'no, K., Nakamura, T., Saito, T.: Analysis of bifurcation phenomena in a 3 cells hysteresis neural network. IEEE Trans. Circuit Syst. I **46**(7), 851–857 (1999)
3. Jin'no, K., Tanaka, M.: Hysteresis quantizer, In: Proceedings of IEEE/ISCAS, pp. 661–664 (1997)
4. Nakaguchi, T., Jin'no, K., Tanaka, M.: Hysteresis neural networks for n-queens problems. IEICE Trans. Fund. **E82–A**(9), 1851–1859 (1999)
5. Nakaguchi, T., Isome, S., Jin'no, K., Tanaka, M.: Box puzzling problem solver by hysteresis neural networks. IEICE Trans. Fund. **E84–A**(9), 2173–2181 (2001)
6. Couvreur, C., Bresler, Y.: On the optimality of the backward greedy algorithm for the subset selection problem. SIAM J. Matrix Anal. Appl. **21**(3), 797–808 (2000)
7. Ertel, W.: Introduction to Artificial Intelligence. Springer, London (2009)
8. Hopfield, J.J.: Neural networks and physical systems with emergent collective computation abilities. Proc. Natl. Acad. Sci. **79**, 2554–2558 (1982)
9. Müezzinoglu, M.K., Güzelis, G.: A Boolean Hebb rule for binary associative memory design. IEEE Trans. Neural Netw. **15**(1), 195–202 (2004)
10. Tarzan-Lorente, M., Gutierrez-Galvez, Dominique Martinez, D., Marco, S.: A Biologically inspired associative memory for artificial olfaction. In: Proceedings of IJCNN, pp. 25–30 (2010)