LETTER

Dynamical Associative Memory: The Properties of the New Weighted Chaotic Adachi Neural Network**

Guangchun LUO[†], Member, Jinsheng REN[†], and Ke QIN^{†*a)}, Nonmembers

SUMMARY A new training algorithm for the chaotic Adachi Neural Network (AdNN) is investigated. The classical training algorithm for the AdNN and its variants is usually a “one-shot” learning, for example, the Outer Product Rule (OPR) is the most used. Although the OPR is effective for conventional neural networks, its effectiveness and adequateness for Chaotic Neural Networks (CNNs) have not been discussed formally. As a complementary and tentative work in this field, we modified the AdNN’s weights by enforcing an unsupervised Hebbian rule. Experimental analysis shows that the new weighted AdNN yields even stronger dynamical associative memory and pattern recognition phenomena for different settings than the primitive AdNN.

key words: chaotic neural networks, chaotic pattern recognition, dynamical associative memory, Adachi Neural Network

1. Introduction

As a type of memory, an Associative Memory (AM) is a well-trained brain-like distribute memory closely associated with artificial Neural Networks (NNs). Association could take one of two forms: autoassociation and heteroassociation. In autoassociation, a NN is required to store a set of patterns. When one pattern or its distorted version is presented to the NN, the system should be able to recall the particular pattern correctly. Heteroassociation differs from autoassociation in that an arbitrary set of input patterns is paired with another arbitrary set of output patterns. If we let $\mathbf{x}_i$ denote a pattern applied to the network and $\mathbf{y}_i$ denote a memorized pattern, an AM can be described by $\mathbf{x}_i \rightarrow \mathbf{y}_i$, $i = 1, 2, \dots, n$. In an autoassociative memory, $\mathbf{y}_i = \mathbf{x}_i$, while in a heteroassociative memory, $\mathbf{y}_i \neq \mathbf{x}_i$.

Both of the two types of AM have a common feature: “static” output. In other words, there exists a “deterministic” mapping between the input and output: the network corresponds a *fixed* input with a *stable* output. In fact, all classic NNs strongly emphasize the system “stability” since it is very useful for computer information processing, however, it is much different from human memory – when we try to retrieve something, one thing reminds us of another, which reminds us of yet another. There is no obvious corre-

spondence between the input and output. Human memory is apparently “nondeterministic”! Neurobiologists have been trying to uncover the underlying memory mechanisms for decades, and they have even made some progress. Now we realize that chaos universally exists in biological systems. Freeman and his co-authors did clinical work on the large-scale collective behavior of neurons in the perception of olfactory stimuli and concluded that the quiescent state of the brain is chaos [1]. It also has been clarified experimentally that squid giant axons responses of a resting nerve membrane to periodic stimulation are not always periodic and the apparently nonperiodic responses can be understood as deterministic chaos. Thus, if we could build an NN possessing chaotic characteristic on the basis of biological models, it may be helpful for mimicking the phenomena of brain’s information retrieval activities. Subsequently, it can lead to a new type of AM, which motivates this letter.

1.1 Rationale for This Letter

Let $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ denote a set of patterns that the network must memorize. Our aim is to design a Chaotic Neural Network (CNN) which is able to correspond a input with a set of outputs instead of a “stable” one during information processing. Mathematically, if $\mathbf{x}_i$ is presented to the network, the entire set of $\mathbf{x}$ could be recalled. The relationship between input and output can be described as $\mathbf{x}_i \rightarrow \{\mathbf{x}_j\}$, $i, j \in [1, n]$. We must emphasize that from the whole system view, unlike conventional AM, this relationship is *not* a mapping since for each single input $\mathbf{x}_i$ there exist multiple outputs. However, if we observe the network as a dynamical system with feed-back, it can be described as $\mathbf{x}_i(t) \rightarrow \mathbf{x}_j(t)$, $i, j \in [1, n]$ satisfying $\mathbf{x}_i(t+1) = \mathbf{x}_j(t)$, which can be understood as an iterated-function.

1.2 The Chaotic Adachi Neural Network Model

The Adachi Neural Network (AdNN) [2], [3] is a network of neurons with weights associated with the edges, a well-defined Present-State/Next-State function, and a well-defined State/Output function. It is composed of N neurons which are topologically arranged as a completely connected graph. A neuron identified by the index i is characterized by two internal states $\eta_i(t)$ and $\xi_i(t)$ at time t , whose values are defined by Eq. (1) and (2) respectively. The output of the i^{th} neuron, $x_i(t)$, is given by a sigmoid function (3), which is defined by Eq. (4)

Manuscript received December 12, 2011.

Manuscript revised March 11, 2012.

[†]The authors are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, 611731 China.

*Corresponding author. Member of IEEE. He was a visiting scholar at Carleton University, Ottawa, Canada.

**This work is supported under the Fund of New Century Excellent Talents in Chinese Universities, Grant NO. NCET-10-0298

a) E-mail: qinkesci@ieee.org

DOI: 10.1587/transinf.E95.D.2158

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

$$\xi_i(t+1) = k_r \xi_i(t) - \alpha x_i(t) + a_i \quad (2)$$

$$x_i(t+1) = f(\eta_i(t+1) + \xi_i(t+1)) \quad (3)$$

$$f(y) = \frac{1}{1 + e^{-y/\varepsilon}} \quad (4)$$

where k_f, k_r, ε are constants, α is the refractory scaling parameter and a_i is the bias term with threshold and temporally constant external input. In Adachi and Aihara's paper [2], [3], they set $\alpha = 10$, $a_i = 2$ for obtaining AM properties and $a_i = 2 + 6x_i$ for Pattern Recognition (PR) properties respectively. The weights $\{w_{ij}\}$ are defined by the Outer Product Rule (OPR):

$$w_{ij} = \frac{1}{4} \sum_{s=1}^4 (2p_{is} - 1)(2p_{js} - 1). \quad (5)$$

where p_{is} denotes the i -th component (or pixel) of the s -th trained pattern. In this letter, we consider the weights carefully and claim the Eq. (5) is not good enough, as will be demonstrated in Sect. 2.2.

As one of the most fascinating CNNs, the AdNN shows a spectrum of chaotic, AM and PR properties under certain settings. Again, we emphasize that these properties could be enhanced by modifying the inappropriate connection weights, as will be illustrated in Sect. 3.

2. Designing the New Weighted AdNN

2.1 The Topology of the Network

The topology of a classical NN usually is a fully connected or neighbor-coupled graph, which is easy to be implemented. However, there are two problems associated with such a structure. One is the excessive computational cost (quadratic function of the neuron number) and the other one is that such topologies have few biological meaning. In [4] it is reported that a real neural system used to has a linear-approximation connectivity. Anatomists have proved that the topology of a real neural system probably satisfies small-world and scale-free characteristics [5]–[7]. The pioneering work in regard of modifying CNNs' topology can be found in our previous work [8], [9]. We succeeded in minimizing the computational burden of the AdNN to be linear. Our that idea is novel but still neither consistent with real neural systems. In Ke Qin's doctoral thesis [10], he managed to modify the AdNN's topology to random, small-world and scale-free graph and obtained the so-called Random Adachi, Small-world Adachi and Scale-free Adachi. According to his experimental results, he claimed the three new NNs also show chaotic, AM and PR properties. While all the above work are very valuable, however, in this letter, we still use a fully-connected graph as the basic topology. This is essentially because, as applied CNN researchers, we still do not know how to design a CNN whose topology possesses

complex network characteristics. This future work could be collaboratively done with physic and neuroanatomy researchers.

2.2 The Weights of the Network

We generally have two modes to train classical recurrent NNs: epoch-wise and continuous training. These two methods have some common features, e.g., they are both based on the method of gradient descent. Thus, neither of them can be used for CNNs because CNNs' behaviors cannot be characterized by gradient descent. So far, almost all of current CNN models use the OPR to fix network connection weights. Although the OPR is the simplest algorithm and has yet been successfully applied to Hopfield NN to obtain AM properties, its effectiveness for CNN has not yet been studied or verified. Therefore, we must design another learning algorithm for CNN training.

Since we have no algorithm at hand to train a CNN, we have to turn to some basic commonly recognized rules, e.g., the Hebb's assumption. According to Hebb's assumption: When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased. This is the fundamental of association learning. The Hebb rule has two forms: supervised and unsupervised. In fact, Aihara and Adachi *et al.* adopted the supervised form. But it's too rough because unlike traditional AM systems, the dynamical AM is able to retrieve "all" the trained patterns instead of a single paired static one, as we stated in Sect. 1.1. Besides, we don't have any input-target samples to use. Thus the unsupervised form, as an alternative, is much better theoretically than the supervised one. The basic unsupervised Hebb rule has the expression:

$$w_{ij}(t+1) = w_{ij}(t) + \beta x_{jq}(t) p_{iq}(t) \quad (6)$$

where β is a learning rate, $x_{jq}(t)$ is the j -th neuron's output corresponds with input pattern P_q at time t , P_q is a column vector whose i -th component is p_{iq} . The physical meaning of Eq. (6) is very obvious: the weight w_{ij} will increase if the j -th neuron's output has the same sign as the i -th neuron's input. Otherwise, the weight will decrease. By such a rule, a NN is able to memorize trained patterns. However, it could lead $\{w_{ij}\}$ to be infinity, which is apparently not right. Thus we introduce a decay item associated with output like "Instar" rule [11]. The modified learning rule is:

$$w_{ij}(t+1) = w_{ij}(t) + \beta x_{jq}(t) p_{iq}(t) - \gamma x_{jq}(t) w_{ij}(t) \quad (7)$$

In our experiments, the output $x_{jq}(t)$ is either 1 or 0. If $x_{jq}(t) = 0$, then $w_{ij}(t+1) = w_{ij}(t)$. Otherwise, the Eq. (7) can be rewritten if we set $\beta = \gamma$.

$$w_{ij}(t+1) = (1 - \beta) w_{ij}(t) + \beta p_{iq}(t) \quad (8)$$

which means w moving towards the trained pattern if the output is active. Repeat this training process using the entire

Fig. 1 The 2-norm of $w(t+1) - w(t)$ during the training process for the Adachi data set. We note that the 2-norm of $w(t+1) - w(t)$ converges to zero after merely about 25 time steps.

training set until the weights converge. We may characterize the training phase as follows:

Algorithm 1 New Weights of the AdNN

- 1: Choose a pattern sequence to train the network. In this letter, the first four patterns presented in Fig. 2 are used, thus $q = 0, 1, 2, 3$ and the neuron number is $N = 100$, $i, j = 1, 2, \dots, N$;
- 2: Set the initial conditions of the network at time $t = 0$ as follows: $q = 0$, $k_f = 0.2$, $k_r = 0.9$, $\eta_i(0) = 0$, $\xi_i(0) = 0$, $x_{jq}(0) = p_{jq}(0)$, $\alpha = 10$, $a_i = 2$, $\varepsilon = 0.015$, $\Delta w(0) = 0$ and $\beta = \gamma = 0.3$. The weight $w_{ij}(0)$ is computed according to Eq. (5)
- 3: At time t , compute the output $x_{jq}(t)$ of the network according to AdNN's definition, as described per Eqs. (1)-(4).
- 4: Update the weight $w_{ij}(t+1)$ for each i and j by Eq. (7).
- 5: Compute the difference $\Delta w_{ij}(t+1) = w_{ij}(t+1) - w_{ij}(t)$ and the 2-norm $\|\Delta w(t+1)\|$, where $\Delta w(t+1)$ is the matrix consisted of $\Delta w_{ij}(t+1)$.
- 6: If $\|\Delta w(t+1)\| \approx 0$, let $w_{ij}^{New} = w_{ij}(t+1)$ as the final weight. Otherwise, let $t = t + 1$, $q = (q + 1) \bmod 4$ and go back to Step 3.

End Algorithm New Weights of the AdNN

2.3 Convergence of the Weights

We proceed a numerical experiment for the Adachi data set to verify the algorithm convergence. A word about the parameter β used is not out of place. The choice of the learning rate is always by trial-and-error method. If it is too small, the convergence is sluggish. Otherwise, it may not converge or can oscillate. In all our experiments, we set $\beta = 0.3$ because the scheme converges and yields the best convergence results.

From Fig. 1 we observe that the norm of the matrix $\Delta w(t+1) = w(t+1) - w(t)$ converges to 0 which indicates the weight $w(t)$ converges very rapidly after merely 25 time steps. That is, after 25 repeated training epoches, the weight $w(t)$ will not change. We must also note that the converged weight $w(t)$ is not equal to w^{old} — the initial weight of the AdNN calculated by OPR.

3. New Properties of the New-AdNN

Now we are in the position of checking whether the newly obtained network, named the New-AdNN, possesses stronger dynamical AM properties. The numerical experiment is conduct on the Adachi data sets as shown in Fig. 2.

Fig. 2 The pattern set used by Adachi. The first four patterns are used to train the network. The fifth pattern includes 15% noise of the forth. The sixth pattern is the untrained pattern.

Table 1 The AM properties comparison. In each case, the pattern in the most left column is the input. The patterns in the second column are those associated ones corresponding to the input. The numbers are the frequencies of each pattern being retrieved within 1,000 iterations.

		Original Patterns		Inverse Patterns	
		New-AdNN	AdNN	New-AdNN	AdNN
P1	P1	18	4	2	2
	P2	33	4	6	2
	P3	17	4	1	5
	P4	16	8	1	5
	Sum	84	20	10	14
P2	P1	17	6	1	4
	P2	20	10	3	7
	P3	16	3	0	4
	P4	23	3	4	2
	Sum	76	22	8	17
P3	P1	13	2	0	1
	P2	31	6	4	2
	P3	27	7	4	3
	P4	21	0	4	3
	Sum	92	15	12	9
P4	P1	12	6	0	3
	P2	25	6	1	5
	P3	23	3	2	4
	P4	20	5	4	4
	Sum	80	20	7	12

3.1 Comparative Associative Memory Properties

Table 1 is the AM properties comparison between the New-AdNN and AdNN in four cases. From the Italic numbers we can see clearly the New-AdNN is able to retrieve all stored patterns with *much* higher frequency. Thus we say the New-AdNN indeed performs better than the AdNN in AM ability.

3.2 Quasi-Energy Function and Long Term Behaviors

We define the quasi-energy function of the New-AdNN as Adachi *et al* did in [3]:

$$QE(t) = -\frac{1}{2} \sum_{i \neq j} \sum_j w_{ij} x_i(t) x_j(t) - \sum_i a_i x_i(t) \quad (9)$$

As shown in Fig. 3, the New-AdNN doesn't converge even after 150,000 iterations. While the AdNN converges shapely to a periodic attractor with period 20 after 24,310 iterations. Thus we conclude the enhanced AdNN has a larger transient phase than the AdNN. Besides, we observe that the New-AdNN's behavior is not strictly chaotic, instead, it has a periodicity 11888! On the contrary, this periodicity phenomenon during the transient phase is not observed for the AdNN.

Fig. 3 The time series of the quasi-energy function of the New-AdNN during the first 150,000 iterations.

Fig. 4 The Hamming distance between the output and the stored patterns (P2 and P4) when external stimulus a corresponding to P4. In the interest of brevity, we just choose the figure of P2 and P4 as a comparison.

3.3 Comparative External Stimulus Dependence and Pattern Recognition Properties

The behavior of the New-AdNN also strongly depends on the external stimulus a , which is possibly be utilized for pattern recognition. We investigated two different external stimulus: a corresponding to a stored pattern (e.g., P4) and an unstored pattern (e.g., P6).

In the first case, during the 6000 iterations, the pattern P4 is retrieved very frequently and seemed non-periodically, as shown in Fig. 4 bottom. However, if we observe the Hamming distance between the output and a stored pattern, P2. We'll see the P2 is periodically recalled after about 292 iterations. We also have confirmed the period is 1372. Thus we summarize the network behavior is indeed periodic after a short transient phase. During each period, the corresponding pattern appears very frequently while the others do not. In the second case, all stored patterns are retrieved a few times with no period and the frequencies are much lower than in the first case. The experimental details are given in Table 2. As we can see clearly, when external stimulus a corresponds to each of those trained patterns (namely, P1, P2, P3 and P4) or even the noisy version of P4, the corresponding pattern can be retrieved with much higher frequency, while the others cannot. The size of transient phase frame and period are also listed in this table. The phenomena that the stored pat-

Table 2 The transient phase frame and period of the network when a corresponding to different stored or unstored patterns.

		Original Patterns	Inverse Patterns	Transient Phase Frame	Period
P1	P1	3170	58	402	1299
	P2	107	87		
	P3	29	24		
	P4	21	25		
P2	P1	439	582	324	81
	P2	2815	0		
	P3	3	2		
	P4	1	0		
P3	P1	1	0	155	57
	P2	104	0		
	P3	3574	104		
	P4	3	1		
P4	P1	1	0	292	1372
	P2	20	10		
	P3	221	87		
	P4	3194	10		
P5	P1	3	1	Infinity	No period
	P2	10	1		
	P3	19	5		
	P4	242	8		
P6	P1	176	2	Infinity	No period
	P2	17	1		
	P3	2	3		
	P4	334	1		

tern corresponding to a special external stimulus is retrieved much more frequently and the others are retrieved much less frequently indicates the network can be utilized for pattern recognition with nonlinear dynamics.

4. Conclusions

In this letter we have concentrated on the chaotic Adachi Neural Network (AdNN), which has been shown to possess chaotic, Associative Memory (AM) and Pattern Recognition (PR) properties. Because the neurons of the AdNN are interconnected through a conventional auto-associative matrix of synaptic weights, which is rough for CNNs, we modified the connection weights so as to render the network possessing stronger AM or PR properties. This is achieved by using the Outer Product Rule to compute the initial weights and then train the network by unsupervised Hebbian rule. Experimental results show that the new weighted network indeed possesses enhanced AM and PR properties. As far as we know, the theoretical and experimental results presented here are both unreported and novel.

References

- [1] W.J. Freeman, "Tutorial on neurobiology: from single neurons to brain chaos," *Int J. Bif & Chaos in Appl Sci & Eng*, vol.2, pp.451–482, 1992.
- [2] K. Aihara, T. Takabe, and M. Toyoda, "Chaotic neural networks," *Phys. Lett. A*, vol.144, no.6-7, pp.333–340, 1990.
- [3] M. Adachi and K. Aihara, "Associative dynamics in a chaotic neural network," *Neural Netw.*, vol.10, no.1, pp.83–98, 1997.
- [4] E.M. Ullian, S.K. Sapperstein, and K.S. Christopherson, "Control of synapse number by gli," *Science*, vol.291, no.26, pp.657–661, 2001.
- [5] P. Hagmann, L. Cammoun, and X. Gigandet, "Mapping the structural core of human cerebral cortex," *PLoS Biology*, vol.6, no.7, pp.1479–1493, 2008.
- [6] C.J. Stam, A. Hillebrand, H. Wang, and P. van Mieghem, "Emergence of modular structure in a large-scale brain network with in-

- teractions between dynamics and connectivity,” *Frontiers in Comp Neusci*, vol.4, pp.1–13, 2010.
- [7] C.W. Shin and S. Kim, “Self-organized criticality and scale-free properties in emergent functional neural networks,” *arXiv:cond-mat/0408700v2 [cond-mat.dis-nn]*, 2008.
- [8] K. Qin and B.J. Oommen, “Adachi-like chaotic neural networks requiring *Linear*-time computations by enforcing a tree-shaped topology,” *IEEE Trans. Neural Netw.*, vol.20, no.11, pp.1797–1809, 2009.
- [9] K. Qin and B.J. Oommen, “An enhanced tree-shaped Adachi-like chaotic neural network requiring linear-time computations,” *Chaotic Systems: Theory and Applications.*, pp.284–293, 2009.
- [10] K. Qin, *Generic Analysis of Chaotic Neural Networks and Their Applications in Pattern Recognition and Crypto-systems*, Ph.D. Thesis, University of Electronic Science and Technology of China, 2010.
- [11] H. Martin and B. Mark, *Neural Networks Design*, 3rd ed., PWS Pub. Co., Boston, MA, 2002.
-