Fast Neural Synchronization Using Geometrical Analysis for Random Walk in Search Space

Naji M. Sahib
Diyala University
Naji_e2006@yahoo.com

Haithem Kareem Abass
Mansour Collage/University
Haithem_72@yahoo.com

Abstract

Neural synchronization is a phenomenon of mutual learning between two or more neural networks. This phenomenon has tremendous applications especially in cryptography where it can be efficiently used in key exchange over public and un secure networks.

This biggest problem facing neural synchronization is the number of rounds needed to accomplish the synchronization; where the output of each neural is sent to other party over network. Since the synchronization is a stochastic behavior then it could be established anytime along the synchronization session, but no current approach to verify establishing the synchronization once it happened.

This paper has deployed geometric analysis to investigate random walk of weights in plane during the synchronization session, the outcome is an enhancement of the convergence of neural networks to synchronization points and another outcome is the verification of establishment the synchronization which is the focus of this paper.

Keywords: Neural network, synchronization, Geometrical analysis, Mutual learning

المستخلص:

التزامن في الشبكة العصبية هو ظاهرة التعلم المتبادل بين شبكتين عصبيتين او اكثر. هذه الظاهرة لها تطبيقات كثيرة وخاصة في التشفير حيث يمكن استخدامها بشكل كفوء جدا في عملية تبادل المفاتيح عبر شبكة عامة غير امينة. ان المشكلة الاكبر التي تواجه ظاهرة التزامن هي عدد الدورات اللازمة لتحقيق التزامن حيث ان مخرج كل شبكة يرسل الى الشبكة المقابلة في الدورة الواحدة وعبر الشبكة. وبما ان عملية التزامن لها تصرف عشوائي فان التزامن قد يحدث في اي وقت خلال جلسة التزامن ولكن لا توجد اي طريقة حالية للتحقق من حصول التزامن ساعة حدوثه. هذا البحث قام بتوظيف التحليل الهندسي لدراسة السير العشوائي للاوزان في مستوي خلال عملية التزامن والنتيجة كانت الوصول الى طريقة تعجل الوصول الى حالة الاتزان اولا وكذلك امكانية التحقق من وصول حالة التزامن.

1. Introduction

Neural networks can synchronize by learning from each other. For that purpose they receive common inputs and exchange their outputs. Adjusting discrete weights according to a suitable learning rule then leads to full synchronization in a finite number of steps. In the case of Tree Parity Machines the dynamics of both processes is driven by attractive and repulsive stochastic forces. Thus it can be described well by models based on random walk, which represent either the weights themselves or order parameters of their distribution. Scaling laws for the number of steps needed for full synchronization and successful learning are derived using analytical models. They indicate that the difference between both processes can be controlled by changing the synaptic depth. In the case of bidirectional interaction the synchronization time increase

proportional to the square of this parameter, but it grows exponentially, if information is transmitted in one direction only [1].

Because of the nature of the synchronization establishment session and its requirement of the mutual interactions, neural synchronization can be used efficiently to construct a cryptographic key-exchange protocol. Where two parties are voluntary engaged in a mutual interaction session to exchange information implicitly, the passive attacker can't learn the key being exchanged due to lacking the interactivity with other parties[1].

Verification of neural synchronization is the crucial point after which the security session can really initiate, otherwise, data would not deciphered properly. The majority of methods to measure neural synchronization are based on spectral representations as Fourier, Hilbert or Wavelet transform which are then used to derive synchronization measures as coherence or phase coupling. From a mathematical point of view these approaches are similar though there is a performance measure that could be used to discriminate between them.[2]

This paper presents synchronization measure of two chaotic systems (i.e., two neural networks), the measure is derived from the geometric analysis of the random walk produced by the mutual learning phase of the neural network. The synchronization of couple chaotic systems has been received considerable attention in the last decade due to its potential applications including secure communication systems and signal-processing systems.

Synchronization scheme is basically depends on a state variable of a given chaotic drive system which is to be the input to drive other chaotic system, the response of the derived chaotic system should be synchronized with the deriving system. [3,4]

2. Perceptron Neural Synchronization

This paper uses the perception neural network to be synchronized due to its features that match the tree parity structure, which is needed to accomplish secure synchronization environment. Perceptron has been chosen for simplicity and ease of implementation.

The structure of the neural network used by this proposed system is composed of one hidden layer and 12 input nodes as it is shown in Figure (1).

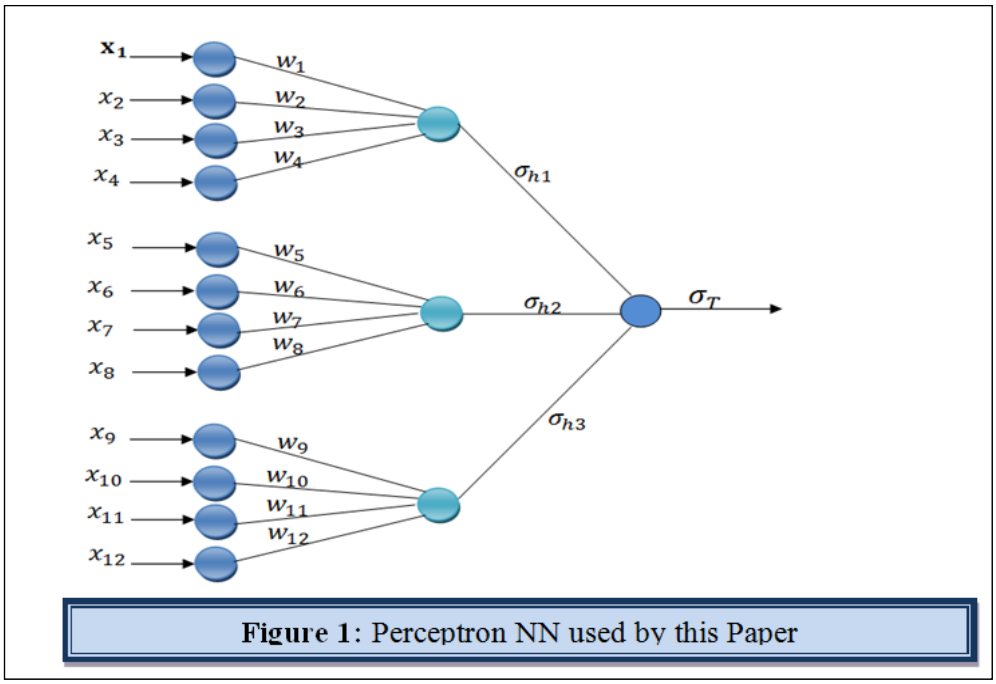

Figure 1: Perceptron NN used by this Paper

This structure has to be possessed by all parties which want to exchange keys based on synchronization of NNs.

Each party, for example A and B uses its own tree parity machine to get its weights to be the key received remotely. Synchronization of the TPMs is achieved in steps presented by Algorithm (1):

1. **Initialize random weight values for all nodes within the tree parity.**
2. **Execute these steps until the full synchronization is achieved:**

1. Generate random input vector X.
2. Compute the values of the hidden neurons.
3. Compute the value of the output neuron.
4. Compare the values of both tree parity machines.
    1. Outputs are others: go to step 2.1.
    2. Outputs are same: one of the suitable learning rules is applied to the weights.

Algorithm 1: Perceptron network synchronization algorithm

After the full synchronization is achieved (the weights w_{ij} of both TPMs are same), A and B can use their weights as keys. This method is known as bidirectional learning. Random walk rule has been used in this proposed system to achieve synchronization.

Random walk rule is given by the following equation

$$w_{t+1} = w_i + x_i\,\theta(\alpha\sigma)\theta(\sigma_A\sigma_B) \qquad \text{eq.1}$$

where w_{t+1} : the next weight

w_i is the previous weight, x_i is the input element, α is the learning rate

σ_A and σ_B are outputs of tree parity in A and B.

2.1 Building Neuron Structure

The basic structure in NN is the neuron which represents the architectural and processing kernel component of the NN. Conceptually, a neuron has three components, as it is presented in Figure (2), where the output is the outcome of the activation components (i.e., $\sigma_i \in [-1, 1]$).

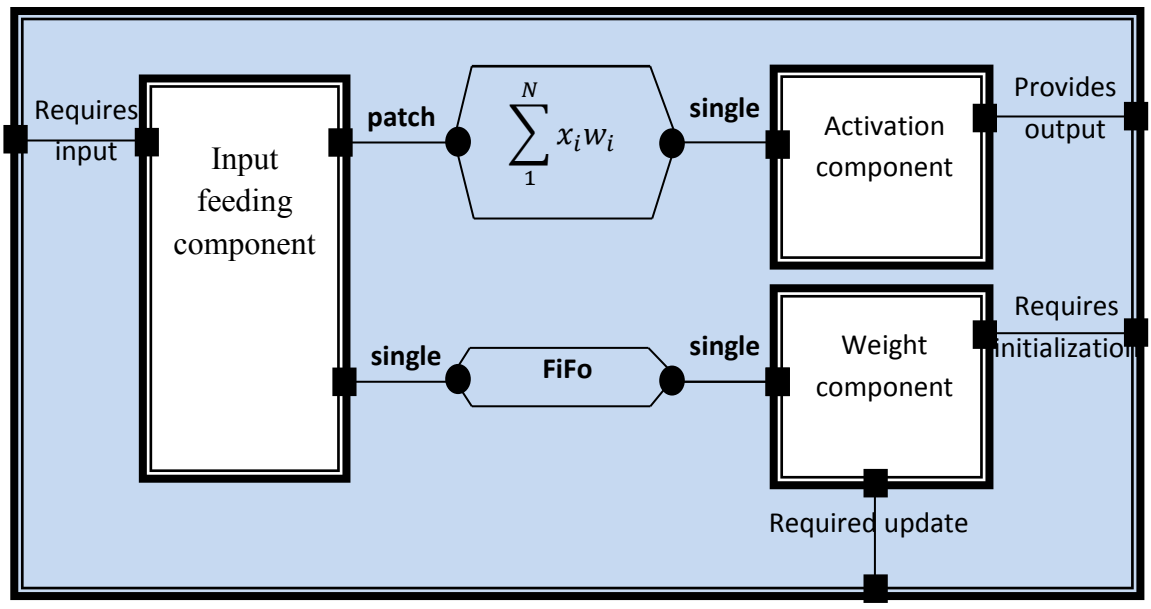

Figure 2: Conceptual view of neuron

After initializing the neuron; tree parity neural network is composed and the output of the neuron is calculated in the same manner at all levels. The method used to calculate the output is:

$$\text{sgn}(y) = \begin{cases} -1 & \text{if } y < 0 \\ 0 & \text{if } y = 0 \\ 1 & \text{if } y > 0 \end{cases}$$

As in other neural networks the weighted sum over the current input values is used to determine the output of the hidden units. Therefore the full state of each hidden neuron is given by its local field.

$$h_i = \frac{1}{\sqrt{N}} w_i . x_i = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} w_{i,j} \; x_{i,j} \qquad \text{eq.2}$$

The output σ_i of the i-th hidden unit is then defined as the sign of h_i, $\sigma_i = \text{sgn}(h_i)$

eq.3

but the special case $h_i = 0$ is mapped to $\sigma_i = -1$ in order to ensure a binary output value. Thus a hidden unit is only active, $\sigma_i = +1$, if the weighted sum over its inputs is positive, otherwise it is inactive, $\sigma_i = -1$.

Then the total output τ of a Tree Parity Machine is given by the product (parity) of the hidden units, τ

$$\tau = \prod_{i=1}^{k} \sigma_i$$

eq.4

So that τ only indicates, if the number of inactive hidden units, with $\sigma_i = -1$, is even ($\tau = +1$) or odd ($\tau = -1$). Consequently, there are 2^{k-1} different internal representations ($\sigma_1, \sigma_2, \ldots, \sigma_k$), which lead to the same output value τ .

If there is only one hidden unit, τ is equal to σ_1. Consequently, a Tree Parity Machine with K = 1 shows the same behavior as a perceptron, which can be regarded as a special case of the more complex neural network.

$$\text{sgn}(y) = \begin{cases} -1 & \text{if } y < 0 \\ 1 & \text{if } y >= 0 \end{cases} \qquad \text{eq.5}$$

3- Proposed fast Synchronization and Verification

Synchronization has to be verified before you start sending encrypted data, otherwise the data would not decrypt in the right way, and this is a crucial challenge in implementing NN synchronization in real world. Since tree parity produces only one value as output and sends it to other side; it will not be enough to get analogous output after certain count of iterations to confirm reaching synchronization state, even if the iteration is huge, thus this proposed system is presenting a technique to confirm synchronization establishment.

The technique used in this proposed system is based on simple vector mathematics as it is presented in figure (3):.

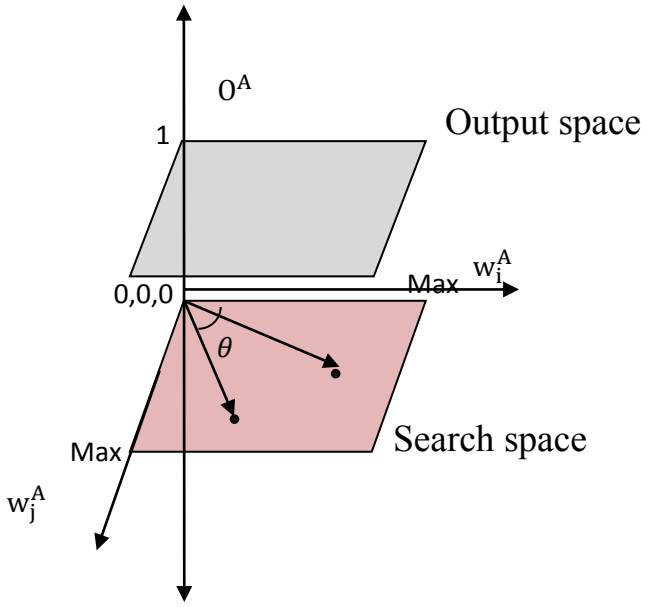

Figure 3: Geometrical representation of two NNs

$$w_{t+1}^A = w_t^A + O^B . X . f(\theta, R) \quad \text{eq.6}$$

$$w_{t+1}^B = w_t^B + O^A . X . f(\theta, R) \quad \text{eq.7}$$

$$= \underset{i}{\rightarrow} w_t^B + \underset{j}{\rightarrow} w_t^B \quad \text{eq.8}$$

$$\theta = \cos^{-1} \frac{w_{t+1}^B \cdot w_t^B}{|w_{t+1}^B||w_t^B|} \quad \text{eq.9}$$

$$R = \sqrt{\underset{i}{\rightarrow} (w_{t+1}^B - w_t^B)^2 + \underset{j}{\rightarrow} (w_{t+1}^B - w_t^B)^2}$$

eq.10

Where

R: The distance between 2 points, each which has been obtained from the vector of weights.

θ: The angle between 2 vectors,

Synchronization will be achieved when:

$$\Delta\theta\ ,\Delta\ R\ =0$$

To increase the convergence speed toward the synchronization point in the plane, the following stepping function is added

$$f(\theta,R)=\ sgn(\Delta\theta).\frac{2}{1+e^{-\Delta\theta.\Delta R}} \qquad \text{eq.11}$$

when this equation has applied to the model, the following results, shown in table (1), from a simulation program written in c++ were obtained:

Table 1: Experimental Results for Regular and Proposed Fast Modifications

I	Input	Neurons	Weight range	Iteration (regular)	Time (regular)	Iteration (fast)	Time (fast)
1	4	3	10	30000	27 sec	27210	13 sec
2	8	6	10	50000	33 sec	34112	22 sec
3	12	9	10	50000	48 sec	31284	27 sec
4	16	12	10	100000	65 sec	77451	41 sec
5	20	15	10	220000	110 sec	143121	67 sec

4- Conclusions

1- **Neural synchronization can be simulated as heuristic random walk in space, and this walk can be guided to make the convergence toward the synchronization point faster than blind walk.**

2- **Synchronization state can be estimated earlier and it is related to the architecture of the neural used to implement synchronization session.**

3- **Geometrical analysis models are efficiently deployed to simulate neural network phenomenon and can be used to enhance other aspect of neural network; this is due to the fact that weights is a vector in space, and neural learning is an optimization for the movement of that vector. Therefore, neural network response to environmental changes can be studied through the geometrical analysis rather than statistical analysis.**

5- References

1- **Andreas Ruttor, "Neural Synchronization and Cryptography", Springer, USA, 2006.**

2- **Lim. Goh, "ICBME 2008- 13th International conference on Biomedical Engineering", volume 23, Singapore, 2008.**

3- **De-Shuang Huang, Laurent heutte, and marco Loog, " Advanced Intelligent Computing Theories and Applications", Proceedings, Qingdao, China, ICIC 2007.**

4- **Frank Spitzer, "Principles of Random Walk", Second edition, Springer, USA,2001**

5- **Daniel S. Yeung, "sensitivity Analysis for Neural Networks", Springer, USA, 2010**