Advances in Science and Technology
Research Journal
Volume 9, No. 26, June 2015, pages 137–142
DOI: 10.12913/22998624/2380

Research Article

DISTANCE OF THE INITIAL WEIGHTS OF TREE PARITY MACHINE DRAWN FROM DIFFERENT DISTRIBUTIONS

Michał Dolecki[1], Ryszard Kozera[1, 2]

[1] The John Paul II Catholic University of Lublin, Konstantynów 1H, 20-708 Lublin, Poland, e-mail: michal.dolecki@kul.pl

[2] Warsaw University of Life Sciences – SGGW, Nowoursynowska 159, 02-776 Warsaw, Poland, e-mail: ryszard.kozera@gmail.com; ryszard_kozera@sggw.pl

Received: 2015.04.12
Accepted: 2015.05.08
Published: 2015.06.01

ABSTRACT

It is well-known that artificial neural networks have the ability to learn based on the provisions of new data. A special case of the so-called supervised learning is a mutual learning of two neural networks. This type of learning applied to a specific networks called Tree Parity Machines (abbreviated as TPM networks) leads to achieving consistent weight vectors in both of them. Such phenomenon is called a network synchronization and can be exploited while constructing cryptographic key exchange protocol. At the beginning of the learning process both networks have initialized weights values as random. The time needed to synchronize both networks depends on their initial weights values and the input vectors which are also randomly generated at each step of learning. In this paper the relationship between the distribution, from which the initial weights of the network are drawn, and their compatibility is discussed. In order to measure the initial compatibility of the weights, the modified Euclidean metric is invoked here. Such a tool permits to determine the compatibility of the network weights' scaling in regard to the size of the network. The proper understanding of the latter permits in turn to compare TPM networks of various sizes. This paper contains the results of the simulation and their discussion in the context of the above mentioned issue.

Keywords: neural networks, neurocryptography.

INTRODUCTION

Most of the stored or transmitted data often carry sensitive information and therefore it is vital to protect them from potential threats [6]. Such a protection relates, among all, to data integrity or authorization of access and confidentiality of communication. One of the main goals here is to transfer the information so that it is impossible to decipher it by any unauthorized and unwanted person. In order to achieve this objective one resorts to various cryptographic algorithms [17].

The message containing information is here encrypted from the plain text to the ciphered text. According to the Kerckhoffs's principle [14], most of the converting algorithms incorporate additional input, i.e. the so-called cryptographic keys [1]. In this approach, the keys generating, distributing and storing methods become equally important issues. One of the options to generate the cryptographic keys is to invoke a well-known Diffie-Helmann protocol [14] which establishes a common key in an open communication channel. However, another interesting approach stems from artificial intelligence, where artificial neural networks for cryptographic key exchange procedure [8, 11] can be applied. This method is based on the phenomenon of two networks' synchronization by their mutual learning [10] described first by Kanter and Kinzel [9].

Artificial neural network learning scheme with incorporated teacher variant relies on the

availability of sample inputs supplemented with corresponding expected results of the network. These impulses are then sent to the taught network which in sequel calculates its output value. Next, the latter result is compared with the expected result and the weights are modified accordingly so that the outcome obtained from the network matches the expected results. In the two networks mutual learning variant is slightly different. First of all, no input-output set is constructed at the preliminary step. Here, the input and output values are generated during the learning process. Both networks receive the same inputs and compute their output values. Then they exchange their respective calculated results. Both networks treat the received results of the counterpart network as the expected values and modify their weights according to the commonly adopted learning algorithm.

Upon imposing some restrictions on the networks weight and their topological structure [15], which is called TPM network (the same for both networks), this learning procedure leads to the synchronization of the common weights in both networks. Such a process is called network synchronization in which a pair of adjusted weights change together yielding different but the same pairs of vectors representing in fact the desired cryptographic keys. An important fact is that learned/taught networks have a dynamic influence on each other by exchanging their outputs. Evidently, the other "eavesdropping" network can intercept communications and gain access to the inputs and outputs of the self-taught networks. However, this can be only passively accomplished since the outputs of the third network are not fed-back to the synchronization process of the other two parties involved.

Consequently, as experimentally verified and statistically proven [10, 11] the passive network is not able to establish compatible weights as quickly as two actively synchronizing networks. Such a phenomenon of establishing compatible network weights during their mutual learning can be therefore exploited in cryptography to construct relatively secure [12, 16] and computationally feasible key exchange protocol [2]. This new method based on TPM networks' synchronization forms an alternative to the computationally difficult key exchange protocol problem which originally is tackled upon applying various advanced NP-contemporary number theory techniques [14].

OBJECTIVE AND METHODOLOGY

The synchronization of TPM network is a stochastic process, which depends on the initial network weights and input values generated at each step of learning. This article discusses the impact of the distribution of the initial weights values drawn on their initial compatibility. The uniform and normal distributions with different parameters are applied and examined here in our examples with TPM networks having different topological parameters. For each variant of the distribution involved and TPM parameters fixed 5000 simulations of the selection of initial weights are carried out, which results in the total number of simulations amounting to 375 000. To evaluate the compatibility of the initial weight the cosine [15] and the modified Euclidean measures [4] are used.

TREE PARITY MACHINE

Tree Parity Machine (TPM) is a neural network used in cryptographic key exchange protocol [10, 18]. It has specific topological structure and its operational *modus vivendi* somehow differs from the conventional neural networks. Indeed, the TPM network consists of two layers of artificial neurons and in fact it is a feed-forward network. In the first layer it has K (K = 1, 2, …) artificial neurons constructed in accordance with the McCulloch – Pitts model [7, 13]. These neurons have not overlapping inputs, which imposes on TPM network a tree-like structure. In the second layer, the network has always one artificial neuron with a specific action, namely it multiplies the results of the first layer neurons. Lastly, the outcome of the output neuron is the result of the entire TPM network. To illustrate the above, a specific topology of the TPM structure is shown in the Figure 1.

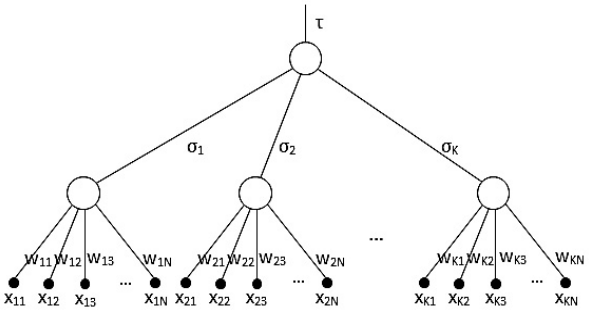

Fig. 1. TPM structure

Each neuron of the first layer has an inputs with values either –1 or 1. The synaptic weights are integers ranging within the interval from $-L$ to L. The summation value of the neuron is calculated without any bias, according to the following formula:

$$\varphi_i = \sum_{j=1}^{N} w_{ij} x_{ij}.$$

The activation function (acting on the summation) of these neurons is a bipolar threshold function, which at the discontinuity is assumed to return the value 1 for argument 0. Therefore, $\sigma_i \in \{-1, 1\}$, for i = 1, 2, …, K and the whole TPM's output as a product of the first layer neuron outputs is either –1 or 1. The Tree Parity Machine has tree-like structure and indicates the parity of negative outputs of first layer neurons.

TPM network learning is performed according to the algorithm, which *de facto* constitutes an extension of the network learning with the teacher. More specifically, we deal here with the case of two networks' mutual learning (and in fact teaching), in which both networks play simultaneously a role of a teacher and a role of a student, interchangeably. The two networks involved iteratively exchange their results as a teacher for the second network and modify their weights as a student. The corresponding network weights are updated according to one of three adopted methods: the anti-Hebbian rule, Hebbian rule or Random Walk rule:

1) Anti-Hebbian rule – in this case neuron weights are modified if the outputs of both networks are different. Weights' modification complies here with the following formula:
$$w_{ki}^{(t+1)} = w_{ki}^{(t)} - x_{ki}\sigma_k .$$
2) Hebbian rule – here weights are modified if the both TPM's results are equal. In this method, weights are modified according to:
$$w_{ki}^{(t+1)} = w_{ki}^{(t)} + x_{ki}\sigma_k .$$
3) Random Walk rule – similar to normal Hebbian rule, where modification occurs only if both results of the networks are equals. The respective weights' adjustments depends here only on the input signal determined by the formula below:
$$w_{ki}^{(t+1)} = w_{ki}^{(t)} + x_{ki} .$$

During entire learning process only the weights of these neurons are modified, for which the outcome is equal to the result of the whole network. If the new weight value $w_{ki}^{(t+1)}$ is greater than L, it is replaced by L and similarly if the weight value is less than $-L$, it's substituted by $-L$, accordingly. As a result of such learning procedure, upon some number of steps, the TPM reaches a compatible weight values i.e. a synchronization state. If the weights' modification is carried out according to either Hebbian or Random Walk rule the corresponding synchronized weights have the same values. On the other hand if anti-Hebbian method is applied then these values are opposite upon synchronization is reached.

Once two TPM networks are synchronized, they remain in this state regardless of their further learning and although each pair of respective weights of such networks changes along next iterations, they remain however equal in pairs. Only the network with an even number of neurons in the first layer may achieve the internal results configuration in which no weight will change in the neurons. This is possible when all hidden neurons have output equal to –1. There is therefore no internal neuron with the same result as the entire network so there is no weight modification.

KEY EXCHANGE PROTOCOL

The outlined above phenomenon of TPM synchronization can be used to construct the cryptographic key exchange protocol. More specifically, let A and B be two trusted parties who intend to establish the cryptographic key in order to encrypt their further communication. The proposed protocol proceeds along the following steps:

0. Parties A and B determine (e.g. through an open communication channel) the parameters K, N and L describing the TPM network topology and the range of interval to which the weights of a learned networks belong. In addition, both parties establish a common learning method. Of course, all such information is also available to the potential attacker (called here the third party).
1. Each of the parties (i.e. A and B) create their own TPM, with secretly randomly chosen respective weights w^A and w^B.
2. Parties A and B obtain the same, publicly known input vector (also available to the third party) and calculate their corresponding TPM results τ^A and τ^B.
3. The Parties A and B exchange the calculated results of their networks (via open channel).

4. Party A treats the result τ^B (received from B) as the expected result for its network, and B proceeds similarly with the value τ^A (obtained from A). Thus Parties A and B play simultaneously the roles of teachers and students.
5. Both Parties A and B modify the weights of the corresponding networks according to the mutually pre-selected learning method.
6. This process continues until the synchronization status is reached.

Once the two networks are synchronized, the entire learning process is completed, otherwise, it is continued upon returning to the step 2. The synchronization moment (i.e. adjusting the pairs of compatible weight vectors) can be accurately detected during the simulation program, which has access to the weight vectors of two networks involved. In practice the phenomenon of TPM network weights synchronization for cryptographic key exchange protocol yields weights which are in turn to be confidential. Therefore, the communication partners do not identify exactly a full synchronization moment. However, they can observe the exchanged results of both networks and once they coincide on different input vectors both TPM networks, such networks are considered as already synchronized. More specifically, sufficiently long exchange of consistent networks' results for random inputs indicates that two networks have consistent weights [18]. We experimentally test here the answer to the question of time synchronization for different TPM parameters.

RESULTS

As it turns out, the time required to synchronize the TPM networks depends on their size. The latter involves the number of neurons in the hidden layer, the number of input values entering each of these neurons, the number of weights and finally the weight variation interval. Evidently, the number of input values is equal to the number of weights involved. All of the above-mentioned values are the network parameters K, N and L. In general, the larger the network is, the longer the synchronization takes [15]. As shown in the previous works [5] the distribution of TPM network synchronization time has a characteristic left-hand histogram. The third quartile is nearly half of the longest observed synchronization time and this relationship remains valid for networks with different topologies [3, 4]. Thus, it is important to focus on the fast synchronizations, which occur more frequently than long ones. Obviously, the synchronization time is crucial for the design of cryptographic key exchange protocol using TPM networks. TPM synaptic weight are assumed to be integers within the interval $[-L, L]$. At the beginning of synchronization procedure, both networks weights are chosen randomly. An initial choice and randomly generated inputs determine the time of network learning. The closer to each other both weight vectors are, the more identical results both networks generate and consequently, they synchronize faster. On the other hand, any initial choice of distant weights may result in a long synchronization time increasing the risk of taking over the cryptographic key by the undesired third party.

The analysis of TPM networks with the initial weights randomly chosen from the uniform distribution and normal distribution with different values of the standard deviation is performed in this paper below. Due to the symmetry of interval $[-L, L]$, where the corresponding weights belong, the mean value of normal distribution is continuously equal to 0. The standard deviation s depends on the network parameter L and the corresponding cases analyzed here include:

$$s \in \left\{L, \frac{L}{2}, \frac{L}{5}\right\}.$$

This distribution parameter has an impact on the frequency with which the weights' values are closer to 0 which in turn influences the initial weight vectors compatibility.

We tested networks with $K = 3$ and $N \in \{11, 50, 100\}$. Parameter L is related to a standard deviation s and to the uniform and the normal distribution with $s = L$. Its value belongs to the following set $L \in \{2, 3, 4, \ldots, 10\}$. For the normal distribution with $s = L/2$ only even values of L are tested, namely $L \in \{2, 4, 6, 8, 10\}$ and for a standard deviation $s=L/5$ only values for L which are divisible by 5 are considered i.e. $L \in \{5, 10\}$. This gives 75 possible variants of parameters of examined networks. Each network is tested with random initial weights draw for 5000 times and in other experiment with 5000 synchronizations. This results in a total number of 750 000 cases.

In order to analyze the initial weights compatibility a cosine function is applied, originally used by Kanter et al. [15] and recently used in modified reversed Euclidean distance [3, 4]. The cosine is calculated according formula:

$$\rho^{AB} = \cos\theta = \frac{w^A \circ w^B}{\sqrt{w^A \circ w^A}\sqrt{w^B \circ w^B}} = \frac{w^A \circ w^B}{\|w^A\| \cdot \|w^B\|},$$

and the previously used Euclidean distance reads as:

$$dist(A,B)_t = \frac{\max\limits_{1 \le j \le t_{synch}} dist(A,B)_j - dist(A,B)_t}{\max\limits_{1 \le j \le t_{synch}} dist(A,B)_j - \min\limits_{1 \le j \le t_{synch}} dist(A,B)_j},$$

where $\max\limits_{1 \le j \le t_{synch}} dist(A,B)_j$ and $\min\limits_{1 \le j \le t_{synch}} dist(A,B)_j$ are assumed to be known. In this work the distance is calculated in relation to the lowest and highest possible values, accordingly. The lowest distance is simply 0, whereas the biggest can be obtained when the corresponding weights have opposite values L and $-L$. It can be calculated as follows:

$$\max dist(A,B) = 2L\sqrt{KN}.$$

Table 1 presents the results obtained for TPM with the respective topological parameters 3-50-L, where $L \in \{2, 3, 4, \ldots, 10\}$ and the weights are drawn according to either uniform and normal distributions with the standard deviation $s = L$. The average values of cosine and Euclidean distance depending on the maximal possible distance from 5000 randomly chosen initial weights vectors are presented in the Table 1.

The tests performed for different TPM networks yielded similar results. Within a fixed value of N, and upon increasing L the average distance of initial weights slightly decreases and the cosine variations get close to zero without a visible regularity. Given the above observation, a further analysis focuses here on the Euclidean distance.

The choice of the specific distribution (from which the initial weights are drawn) results in the variation of distances between the initial drawn weights. In particular, the normal distribution renders a closer pairs of weights as opposed to the uniform one. Table 2 shows the average distances with respect to the maximal distance for 5000 random draws of the initial weights for all analyzed networks. Both uniform and normal distributions (with standard deviation equal to L or $L/2$) are presented below.

Standard deviation of the used normal distribution depends on the interval to which the weights of the tested network belong. The distances between randomly generated weight vectors are scaled here to the maximal possible distance appearing in a tested TPM network of a given fixed structure. Consequently, all generated results are similar and independent from different topological parameters.

Reducing the standard deviation leads to the further alignment of the initial weight vectors. Indeed, the similarity of the results described above permits to average them, which in turn is summarized in Table 3. The latter visibly indicates that the distance between the weights' vectors is reduced in percentage.

In case when the weights are drawn according to a normal distribution the respective average distance decreases. For the standard deviations either L or $L/2$ or $L/5$, the corresponding calculated distance is respectively 41.4%, 32.6% and 39.8% of the maximal distance and what is interesting this percentage occurs regardless of the size of the TPM network.

CONCLUSIONS

Having comparing the uniform and normal distributions with respect to the weights selection we observe that the latter favors the drawn weight values closer to the mean. Therefore, the distance between weight vector decreases. It is interest-

Table 1. Results for TPM with the topological parameters 3-50-L

TPM			UNIFORM		NORMAL	
K	N	L	COS	DIST	COS	DIST
3	50	2	-0.001754	0.499937	-0.001107	0.456752
3	50	3	-0.000102	0.470752	-0.001080	0.432801
3	50	4	0.002459	0.455380	0.002562	0.419601
3	50	5	0.001128	0.446826	-0.000657	0.412462
3	50	6	0.001672	0.440176	0.001753	0.406837
3	50	7	0.000207	0.436068	0.002923	0.403342
3	50	8	-0.001378	0.432749	-0.001870	0.401124
3	50	9	-0.000908	0.429989	0.000275	0.398876
3	50	10	0.000699	0.427757	0.000988	0.397211

Table 2. Distances for different TPMs and initial weights' distributions

L	UNIFORM	NORMAL	
		s = L	s = L/2
N = 11			
2	0.49657	0.45431	0.34725
4	0.45363	0.41857	0.32682
6	0.43829	0.40572	0.32137
8	0.42964	0.40001	0.31811
10	0.42527	0.39483	0.31553
N = 50			
2	0.49994	0.45675	0.34976
4	0.45538	0.41960	0.32909
6	0.44018	0.40684	0.32265
8	0.43275	0.40112	0.31979
10	0.42776	0.39721	0.31792
N = 100			
2	0.49986	0.45669	0.34966
4	0.45609	0.42021	0.32951
6	0.44090	0.40776	0.32313
8	0.43252	0.40120	0.31980
10	0.42808	0.39759	0.31796

Table 3. Average distance for TPMs with different weights' interval

N	UNIFORM [%]	NORMAL [%]		
		L	L/2	L/5
11	44.7	41.3	32.6	29.7
50	44.9	41.4	32.8	29.8
100	44.9	41.5	32.8	29.8

ing that expressing this distance, in relation to the maximal possible distance, yields similar results for networks with different parameters and thus renders the result independent from the size of the examined networks. The weights which are drawn closer to each other at the initial phase of the synchronization process should shorten the network synchronization time. This issue will be analyzed in a further research.

REFERENCES

1. Barker E., Barker W., Burr W., Polk W., Smid M.: Recommendation for key management – part 1: general (revision 3). National Institute of Standards and Technology Special Publication, 2012, 800–857.
2. Bisalapur S.: Design of an efficient neural key distribution center. International Journal of Artificial Intelligence & Applications, 2 (1), 2011, 60–69.
3. Dolecki M., Kozera R.: Threshold method of detecting long-time TPM synchronization. Proceedings of the 12th International Conference on Computer Information Systems and Industrial Management Applications, Lecture Notes in Computer Science 8104, Springer – Verlag Berlin, 2013, 241-252.
4. Dolecki M., Kozera R., Lenik K.: The evaluation of the TPM synchronization on the basis of their outputs. Journal of Achievements in Materials and Manufacturing Engineering, 57, 2013, 91–98.
5. Dolecki M., Kozera R.: Distribution of the Tree Parity Machine synchronization time. Advances in Science and Technology Research Journal, 18, 2013, 20–27.
6. Gil A., Karoń T.: Analiza środków i metod ochrony systemów operacyjnych. Postępy Nauki i Techniki, 12, 2012, 149–168.
7. Hassoun M.: Fundamentals of Artificial Neural Networks. MIT Press, 1995.
8. Ibrachim S., Maarof M.: A review on biological inspired computation in cryptology. Jurnal Teknologi Maklumat, 17(1), 2005, 90–98.
9. Kanter I., Kinzel W.: The theory of neural networks and cryptography. The Physics of Communication: Proceedings of the XXII Solvay Conference on Physics, 2002, 631–644.
10. Kanter I., Kinzel W., Kanter E.: Secure exchange of information by synchronization of neural networks. Europhysics Letters, 57, 2002, 141–147.
11. Klein E., Mislovaty R., Kanter I., Ruttor A., Kinzel W.: Synchronization of neural networks by mutual learning and its application to cryptography. Advances in Neural Information Processing Systems, 17, MIT Press, Cambridge, 2005, 689–696.
12. Klimov A., Mityagin A., Shamir A.: Analysis of neural cryptography, [In:] Y. Zheng (ed.), Advances in Cryptology – ASIACRYPT 2002, Springer, 2003, 288–289.
13. McCulloch W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics, 5, 1943, 115–133.
14. Menezes A., Vanstone S., Van Oorschot P.: Handbook of Applied Cryptography. CRC Press, 1996.
15. Ruttor A.: Neural Synchronization and Cryptography. PhD thesis, Wurzburg 2006.
16. Ruttor A., Kinzel W., Naeh R., Kanter I.: Genetic attack on neural cryptography. Physical Review E, 73(3), 2006, 036121.
17. Stokłosa J., Bilski T., Pankowski T.: Bezpieczeństwo danych w systemach informatycznych. PWN, 2001.
18. Volkmer M., Wallner S.: Tree parity machine rekeying architectures. IEEE Transactions on Computers, 54, 2005, 421–427.