International Journal of Soft Computing 3 (3): 177-181, 2008
ISSN: 1816-9503

Neural Cryptography with Multiple Transfers Functions and Multiple Learning Rule

[1]N. Prabakaran, [1]P. Loganathan and [1]P. Vivekanandan
[1]Department of Mathematics, Anna University, Chennai, India

Abstract: The goal of any cryptographic system is the exchange of information among the intended users. We can generate a common secret key using neural networks and cryptography. In the case of neural cryptography is based on a competition between attractive and repulsive forces. A feedback mechanism is added to neural cryptography which increased the repulsive forces. The partners A and B have to use a cryptographic key exchange protocol in order to generate a common secret key over the public channel. This can be achieved by two Tree Parity Machines (TPMs), which are trained on their mutual output synchronize to an identical time dependent weight vector. The proposed TPMs, each output vectors are compare then updates from hidden unit using Hebbian Learning Rule and dynamic unit using Random Walk Rule with feedback mechanism. We can enhance the security of the system using different learning rule with different units. A network with feedback generates a pseudorandom bit sequence which can be used to encrypt and decrypt a secret message. The advanced attacker presented here, named the Majority Flipping Attacker is the first whose does not decay with the parameters of the model. The probability of a successful attack is calculated for different model parameters using numerical simulations.

Key words: Neural cryptography, tree parity machine, neural synchronization, feedback mechanism, majority flipping attacks

INTRODUCTION

The study of neural networks was originally driven by its potential as a powerful learning and memory machine (Rozen-Zvi *et al*., 2002). To send a secret message over a public channel one needs a secret key either for encryption, decryption or both. In 1976, Diffie and Hellman have shown how to generate a secret key over a public channel for exchange of secret message. Recently it has been shown how to use synchronization of neural network to generate secrete keys over public channel. This algorithm called neural cryptography is not based on number theory but it contains a physical mechanism (Ruttor *et al*., 2004).

Neural cryptography (Kanter and Kinzel, 2002; Kinzel, 2002) is based on the effect that two neural networks are able to synchronize by mutual learning (Ruttor *et al*., 2006). In each step of this online learning procedure they receive a common input pattern and calculate their output. Then, both neural networks use those outputs present by their partner to adjust their own weights. This process leads to fully synchronized weight vectors.

Synchronization of neural networks is, in fact, a complex dynamical process. The weights of the networks perform random walks, which are driven by a competition of attractive and repulsive stochastic forces. Two neural networks can increase the attractive effect of their moves by cooperating with each other. But, a third network which is only trained by the other two clearly has a disadvantage, because it cannot skip some repulsive steps. Therefore, bidirectional synchronization is much faster than unidirectional learning (Ruttor *et al*., 2004).

Two partners A and B want to exchange a secret message over a public channel. In order to protect the content against an attacker E, who is listening to the communication, A encrypts the message, but B needs A's secret key for decryption. Without an additional private channel A and B have to use a cryptographic key exchange protocol in order to generate a common secret key over the public channel (Kinzel, 2002). This can be achieved by synchronizing two TPMs (Tree Parity Machines), one for A and one for B, respectively. In this study we introduce a mechanism, which is based on the generation of inputs by feedback.

A measure of the security of the system is the probability P_E that an attacking network is successful. We calculate P_E obtained from the best known attack for different model parameters and search for scaling properties of the synchronization time as well as for the security measure. It turn out that feedback improves the

Corresponding Author: N. Prabakaran, Department of Mathematics, Anna University, Chennai, India

security significantly but it also increases the effort to find the common key when this effort is kept constant, feedback only yields a small improvement of security (Ruttor, 2004).

Neural cryptography: The proposed Tree Parity Machine is used by partners and an attacker in neural cryptography consists of K-hidden units and K-dynamic units, each of them being a perceptron with an N-dimensional weight vector w_k (Engel and Van Den, 2001) When a hidden and dynamic unit receives an N-dimensional input vector x_k it produces the output bit (Volkmer and Wallner, 2005).

The general structure of the networks is shown in Fig. 1. All inputs values are binary,

$$x_{ij} \in \{-1, +1\} \text{ and } x_{im} \in \{-1, +1\} \quad (1)$$

and the weights are discrete number between -L and +L,

$$w_{ij} \in \{-L, -L+1, \ldots\ldots, L-1, L\} \text{ and } w_{im} \in \{-L, -L+1, \ldots\ldots, L-1, L\}. \quad (2)$$

The index i = 1,….., K denotes the ith hidden unit of TPM and m = 1,….., K dynamic unit of the TPM and j = 1,…., N denotes the N components (Ruttor *et al*., 2006). We mainly consider two transfer functions as given:

$$\sigma_i = \text{sign}\left(\sum_{j=1}^{N} w_{ij} \bullet x_{ij}\right) \quad (3)$$

$$\delta_i = \tanh\left(\sum_{m=1}^{N} w_{im} \bullet x_{im}\right) \quad (4)$$

Where Eq. 3 is the hidden unit and the Eq. 4 is the dynamic unit.

The K hidden and dynamic units σ_i and δ_i define a common output bit τ of the total network by

$$\beta_a = \prod_{i=1}^{K} \sigma_i \quad \text{for hidden units} \quad (5)$$

$$\beta_b = \prod_{i=1}^{K} \delta_i \quad \text{for dynamic units} \quad (6)$$

The two TPMs (Volkmer and Wallner, 2005) compare the output bits then update the values between hidden units and dynamic units as well as 2 parties A and B that are trying to synchronize their weights

$$\phi_i = \text{comp}(\beta_a, \beta_b) \quad (7)$$

Where

$$\phi_i^A = w_{ij}^A x_{ij} \tau^B \sigma_i \quad \text{and} \quad (8)$$

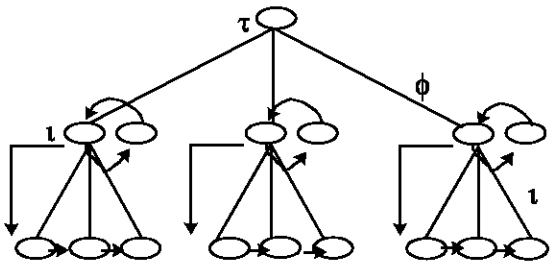

Fig. 1: A structure of tree parity machine

$$\phi_i^B = w_{ij}^A x_{ij} \tau^B \delta_i \quad (9)$$

Each of the 2 communication parties A and B has their own network with an identical TPM architecture. Each party selects a random initial weight vectors $w_i(A)$ and $w_i(B)$ at t = 0.

Both of the networks are trained by their mutual output bits τ^A and τ^B. At each training step, the 2 networks receive common input vectors x_i and the corresponding output bit τ of its partner (Godhavari *et al*., 2005). We use the following learning rule.

- If the output bits are different, $\tau^A \neq \tau^B$, nothing is changed.
- If $\tau^A = \tau^B \equiv \tau$ the hidden and dynamic units are trained which have an output bit identical to the common output $\Phi_i^{A/B} = \tau^{A/B}$.
- To adjust the weights we consider two different learning rules.

(a) Hebbian Learning rule for hidden units

$$w_i^A(t+1) = w_i^A(t) + x_i \tau^A \Theta(\tau^A \phi_i^A) \Theta(\tau^A \tau^B)$$
$$w_i^B(t+1) = w_i^B(t) + x_i \tau^B \Theta(\tau^B \phi_i^B) \Theta(\tau^A \tau^B) \quad (10)$$

(b) Random walk learning for dynamic units

$$w_i^A(t+1) = w_i^A(t) + x_i \Theta(\tau^A \phi_i^A) \Theta(\tau^A \tau^B)$$
$$w_i^B(t+1) = w_i^B(t) + x_i \Theta(\tau^B \phi_i^B) \Theta(\tau^A \tau^B) \quad (11)$$

If any component w_i moves out of the interval {-L, L}, it is replaced by sign (w_i) L. This can treated as a random walk with reflecting boundaries. Using this algorithm the 2 neural networks synchronize to a common secret key.

We analyze the process of synchronization using simulation of finite systems as well as iterative calculations corresponding hidden units i are described by 2L + 1 variables $P^i_{a,b}$, (t), which are defined as the probability to find a weight with W^A_{ij} (t) = a in A's TPM and W^B_{ij} (t) = b in B's TPM at time t.

$$p^i_{a,b}(t) = p\,(w_{ij}^A(t) = a \wedge w_{ij}^B(t) = b \quad (12)$$

While these quantities are approximately given by the frequency of the weight values W^{A}_{ij} (t) and W^{B}_{ij} (t) in simulations, we use the for simple attack as given:

$$w_i^{E+} = w_i^E + x_i \Theta (\sigma_i^E \tau^A) \Theta (\tau^A \tau^B) \quad (13)$$

To determine the time evolution of P^i_{ab} (t) directly in the limit $N \to \infty$.

In both cases the standard order parameters (Kinzel *et al.*, 2000), which are commonly used for the analysis of online learning can be calculated as functions of P^i_{ab} (t)

$$Q_i^A = \frac{1}{N} W_i^A W_i^A = \sum_{a=-L}^{L} \sum_{b=-L}^{L} a^2 p^i_{a,b} \quad (14)$$

$$Q_i^B = \frac{1}{N} W_i^B W_i^B = \sum_{a=-L}^{L} \sum_{b=-L}^{L} b^2 p^i_{a,b} \quad (15)$$

$$R_i^{A,B} = \frac{1}{N} W_i^A W_i^B = \sum_{a=-L}^{L} \sum_{b=-L}^{L} a\, b\, p^i_{a,b} \quad (16)$$

The level of synchronization between 2 corresponding hidden units and dynamic units is given by the normalized overlap

$$p_i = \frac{w_i^A \bullet w_i^B}{\sqrt{w_i^A \bullet w_i^B}\sqrt{w_i^A \bullet w_i^B}} = \frac{R_i^{A,B}}{\sqrt{Q_i^A Q_i^B}} \quad (17)$$

Uncorrelated weight vector at the beginning of the synchronization process have $P_i = 0$ while value $P_i = 1$ is reached for fully synchronized weights (Ruttor *et al.*, 2006).

SYNCHRONIZATION WITH FEEDBACK

The TPMs A and B which start with different random weights and common random inputs (Ruttor *et al.*, 2004) The feedback mechanism is defined as follows:

- After each step 't' the input is shifted, $x_{i,j}(t+1) = x_{i,j-1}(t)$ for $j > 1$.
- If the output bits agree, $\tau^A(t) = \tau^B(t)$, the output of each hidden unit is used as a new input bit, $x_{i,j}(t+1) = \phi_i(t)$ are set to common public random values.
- After R steps with different outputs, $\tau^A(t) = \tau^B(t)$, all input vectors are reset to public common random vectors, $x^A_{i,j}(t+1) = x^B_{i,j}(t+1)$.

The feedback creates correlations between the weights and the inputs therefore the system becomes sensitive to the learning rule. The Hebbian or Random walk rule, the entropy is much smaller, because the values of the weights are pushed to the boundary values ±L. In Fig. 2, we have numerically calculated the averaged time as a function of the number L of components for the Hebbian rule of hidden units and Random walk rule of dynamic units. There is a large deviation from the scaling law $t_{sync} \propto L^2$ as observed for R = 0.

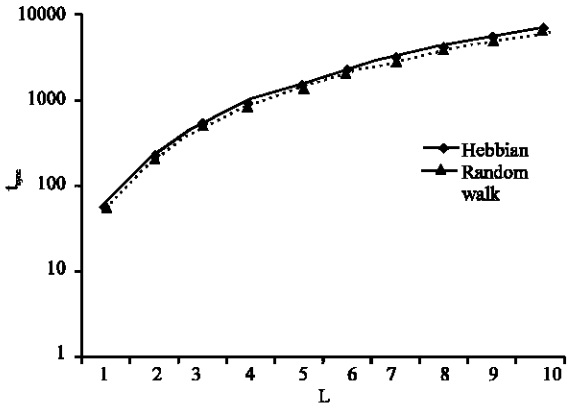

Fig. 2: Average Synchronization time between Hebbian and Random walk of t_{sync} and its standard deviation as a function of L, from TPM with K = 3. Simulation results obtained using $N = 10^3$

RESULTS

The attacker E tries to learn the weight vector of one of the two machines (Mislovaty *et al.*, 2004). The values of N and L are public as well as all the transmission through the channel: input x_{ij} and output $\tau^{A/B}$. The information E lacks in each learning step are the values $\{\sigma_i\}$ of A's hidden units and $\{\delta_i\}$ of A's dynamic units (i.e.) which of the $2^{K-1} * 2^{K-1}$ possible updating scenarios A performs.

The most successful attack on neural cryptography is the Majority Flipping Attack, which is an extension of the Geometric Attack. The Attacker E uses an ensemble of 'n' TPMs. At the beginning, the weight vectors of all attacking networks are chosen randomly, so that the average initial overlap between them is zero. If the 2 partners A and B use queries for the neural key exchange, the success probability strongly depends on the parameter H. This can be used to regain security against the Majority Flipping Attack (Fig. 3).

$$P_E = \frac{1}{1 + e^{-\beta(H-\mu)}} \quad (18)$$

With 2 parameters β and μ is a suitable fitting function for describing P_E as a function of H. Figure 4 shows the dependence on L of the fit parameter μ for both attacks.

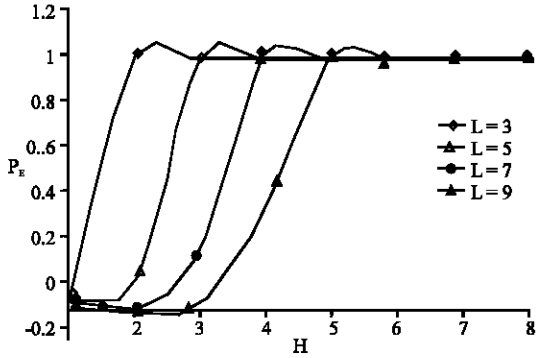

Fig. 3: Success probability of the Majority Flipping Attack as a function of H. Symbols denote the results obtained in 1000 simulations for K = 3, n = 100 and N = 1000

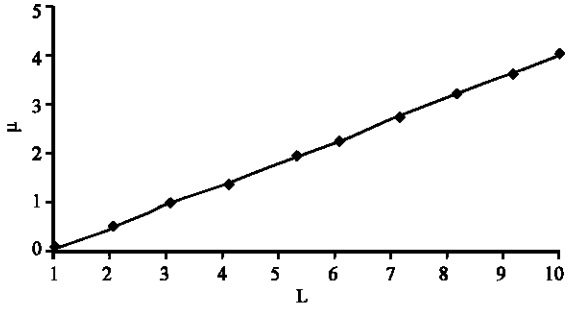

Fig. 4: Parameter μ as a function of synaptic depth L. Symbols denote the results of fits using for the Geometric Attack and Majority Flipping Attack with M = 100

A side from finite size effects for small values of L, this parameter is proportional to the synaptic depth of the TPMs:

$$\mu = \alpha_s L \tag{19}$$

Obviously, the quantity $\alpha < H/L$ not only determines the synchronization time but also the success of attack. Here the probability of Majority Flipping Attack (Kanter *et al.*, 2004). P_{flip} decrease exponentially with increasing synaptic depth in Hebbian Learning Rule (Fig. 5).

$$P_{flip(a)} \alpha e^{-AL} \tag{20}$$

The probability of Majority Flipping Attack P_{flip}, in Random Walk Rule

$$P_{flip(b)} \alpha e^{-BL} \tag{21}$$

The average of Majority Flipping Attack Eq. 20 and Eq. 21 is

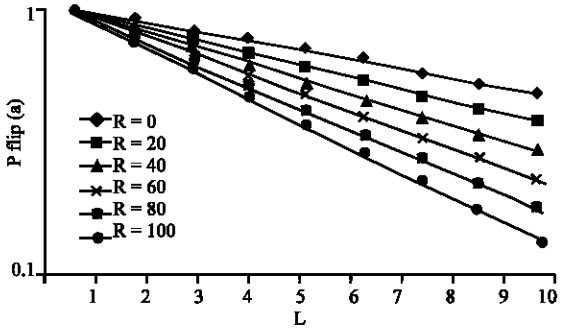

Fig. 5: The Probability P_{flip} as a function of L, averaging 1000 simulation with K = 3 and n = 1000 in Hebbian Learning Rule

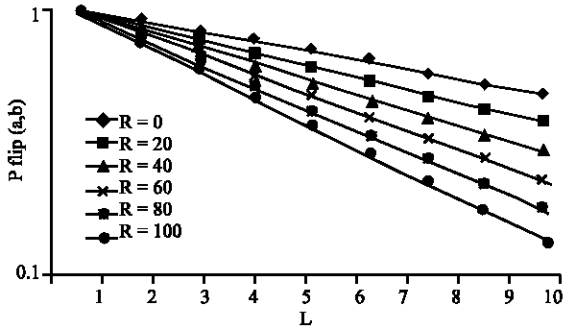

Fig. 6: The Probability P_{flip} as a function of L, found from 10000 runs of the iterative equations for K = 3

$$P_{flip(a,b)} \alpha e^{-((A+B)/2)L} \tag{22}$$

The Eq. 22 shows that, the feedback improves the security of neural cryptography. The synchronization time on the other side also increased (Fig. 6).

CONCLUSION

The Neural Cryptography is based on repulsive and attractive stochastic forces. A feedback mechanism for hidden and dynamic units has been increased the synchronization time of 2 networks and decreases the probability of a successful attack. The synchronization time, feedback yields a small enhancement the security of the system. After synchronization, the system is generates a pseudorandom bit sequence which passes test on random numbers like gap test and poker test. When another network is trained on this bit sequence it is not possible to extract some information on the statistical properties of the sequence. The Tree Parity Machines generate a secret key and also encrypt and decrypt a secret message.

REFERENCES

Aydogan, S., 2007. Multifeedback-Layer Neural Network. IEEE. Trans. Neural Network, 18: 373-384.

Engel, A. and C. Van Den Broeck, 2001. Statistical Mechanics of Learning. Cambridge University Press, Cambridge.

Godhavari, T., N.R. Alamelu and R. Soundarajan, 2005. Cryptography Using Neural Network. IEEE. Indicon Conf., pp: 258-261.

Kanter, I., W. Kinzel, L. Shancham, E. Klein and R. Mislovaty, 2004. Cooperating attackers in neural cryptography. Phys Rev. E, Vol. 69.

Kanter, I. and W. Kinzel, 2002. Neural cryptography. In: Proceedins of the 9th International Conference on Neural Information Processing, Singapore.

Kanter, I., W. Kinzel and E. Kanter, 2002. Secure exchange of information by synchronization of neural network, Eur. Phys. Lett., 57: 141-147.

Kinzel, W. and I. Kanter, 2002. Interacting neural networks and cryptography. Advances in Solid State Physics, by B. Kramer (Springer, Berlin), 42: 383-391.

Kinzel, W., 2002. Theory of Interacting Neural Network. Preprint cond-mat/0204054.

Kinzel, W., R. Metzler and I. Kanter, 2000. Dynamics of Interacting neural networks. J. Phys. A: Math. Gen., 33: 141-149.

Klimov, A., A. Mityagin and A. Shamir, 2002. Analysis of neural cryptography. In: Proc. Asia Crypt 2002. Volume 2501 of LNCS, Queenstown, New Zealand, Springer Verlag, pp: 288-298.

Metzler, R., W. Kinzel and I. Kanter, 2000. Interacting Neural Networks. Phys. Rev. E, 62: 2555-2565.

Mislovaty, R., E. Klein, I. Kanter and W. Kinzel, 2003. Public channel cryptography by synchronization of neural networks and chaotic maps. Phys. Rev. Lett., Vol. 91.

Mislovaty, R., E. Klein, I. Kanter and W. Kinzel, 2004. Security of Neural cryptography. IEEE. Trans., 5: 219-221.

Mislovaty, R., Y. Perchenok, I. Kanter and W. Kinzel, 2002. Secure key-exchange protocol with an absence of injective functions. Phys. Rev. E, 66: 1-4.

Rosen-Zvi, M., E. Kleign, I. Kanter and W. Kinzel, 2002. Cryptography based on neural networks-analytical results. Phys. Rev. E, 35: 703-713.

Rosen-Zvi, M., E. Kleign, I. Kanter and W. Kinzel, 2002. Mutual learning in a tree parity machine and its application to cryptography. Phys. Rev. E, vol. 66(13): 066135.

Ruttor, A., G. Reents and W. Kinzel, 2004. Synchronication of random walk with reflecting boundaries. J. Phys. A: Math.Gen., 37: 8609 [cond-mat/0405369].

Ruttor, A., I. Kanter and W. Kinzel, 2006. Dynamics of neural cryptography [cont-mat/061257/ 21].

Ruttor, A., I. Kanter, R. Naeh and W. Kinzel, 2006. Genetic attack on neural cryptography [cond-mat/0512022 v2/1].

Ruttor, A., W. Kinzel, L. Shacham and I. Kanter, 2004. Neural cryptography with feedback. Phys. Rev. E, 69: 1- 8.

Volkmer, M. and S. Wallner, 2005. Proceedings of the 1st International Workshop on Secure and Ubiquitous Net-works, SUN'05. IEEE. Comput. Soc. Copenhagen, pp: 241-245.

Volkmer, M. and S. Wallner, 2005. Tree Parity Machine Rekeying Architectures. IEEE. Trans. Comput., 54: 421-427.