Neural cryptography with feedback

Andreas Ruttor and Wolfgang Kinzel
Institut für Theoretische Physik, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany

Lanir Shacham and Ido Kanter
Department of Physics, Bar Ilan University, Ramat Gan 52900, Israel

(Dated: 26 November 2003)

Neural cryptography is based on a competition between attractive and repulsive stochastic forces. A feedback mechanism is added to neural cryptography which increases the repulsive forces and improves the security of the system. In addition, a network with feedback generates a pseudorandom bit sequence which can be used to encrypt and decrypt a secret message.

PACS numbers: 84.35.+i, 87.18.Sn, 89.70.+c

I. INTRODUCTION

Neural networks learn from examples. When a system of interacting neurons adjusts its couplings to a set of externally produced examples, this network is able to estimate the rule which produced the examples. The properties of such networks have successfully been investigated using models and methods of statistical physics [1, 2].

Recently this research program has been extended to study the properties of interacting networks [3, 4]. Two networks which learn the examples produced by their partner are able to synchronize. This means that after a training period the two networks achieve identical time dependent couplings (synaptic weights). Synchronization by mutual learning is a novel phenomena which has been applied to cryptography [5, 6].

To send a secret message over a public channel one needs a secret key, either for encryption, decryption or both. In 1976, Diffie and Hellmann have shown how to generate a secret key over a public channel without exchanging any secret message before. This method is based on the fact that—up to now—no algorithm is known which finds the discrete logarithm of large numbers by feasible computer power [7].

Recently it has been shown how to use synchronization of neural networks to generate secret keys over public channels [5]. This novel algorithm, called neural cryptography, is not based on number theory but it contains a physical mechanism: The competition between stochastic attractive and repulsive forces. When this competition is carefully balanced, two partners A and B are able to synchronize whereas an attacking network E has only a very low probability to find the common state of the communicating partners.

The security of neural cryptography is still being debated and investigated [8–12]. In this paper we introduce a new mechanism which is based on the generation of inputs by feedback. This feedback mechanism increases the repulsive forces between the participating networks, and the amount of the feedback, the strength of this force, is controlled by an additional parameter of our model.

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 [8] for different model parameters and search for scaling properties of the synchronization time as well as for the security measure. It turns out that feedback improves the 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.

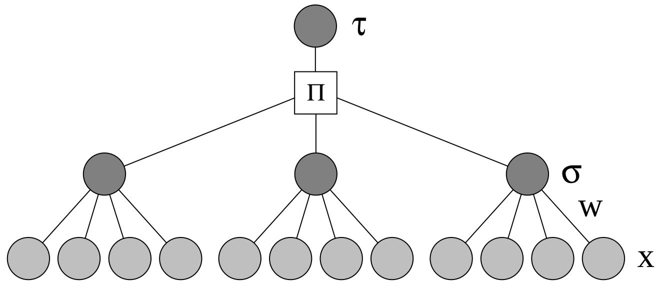

FIG. 1: A Tree Parity Machine with $K = 3$ and $N = 4$.

II. REPULSIVE AND ATTRACTIVE STOCHASTIC FORCES

The mathematical model used in this paper is called a Tree Parity Machine (TPM), sketched in Fig. 1. It consists of K different hidden units, each of them is a perceptron with an N-dimensional weight vector $\mathbf{w}_k$. When a hidden unit k receives an N-dimensional input vector $\mathbf{x}_k$ it produces the output bit

$$\sigma_k = \text{sign}(\mathbf{w}_k \cdot \mathbf{x}_k) . \qquad (1)$$

The K hidden units σ_k define a common output bit τ of the total network by

$$\tau = \prod_{k=1}^{K} \sigma_k . \qquad (2)$$

In this paper we consider binary input values $x_{k,j} \in \{-1, +1\}$ and discrete weights $w_{k,j} \in \{-L, -L+1, ..., L-$

$1, L\}$, where the index j denotes the N components and k the K hidden units.

Each of the two communicating partners A and B has an own network with an identical TPM architecture. Each partner selects random initial weight vectors $\mathbf{w}_k^A(t=0)$ and $\mathbf{w}_k^B(t=0)$.

Both of the networks are trained by their mutual output bits τ^A and τ^B. At each training step, the two networks receive common input vectors $\mathbf{x}_k$ and the corresponding output bit τ of its partner. We use the following learning rule:

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

  (i) anti-Hebbian learning

$$\mathbf{w}_k^+ = \mathbf{w}_k - \tau \mathbf{x}_k \Theta(\sigma_k \tau)\Theta(\tau^A \tau^B)\,; \tag{3}$$

  (ii) Hebbian learning

$$\mathbf{w}_k^+ = \mathbf{w}_k + \tau \mathbf{x}_k \Theta(\sigma_k \tau)\Theta(\tau^A \tau^B)\,; \tag{4}$$

  (iii) random walk

$$\mathbf{w}_k^+ = \mathbf{w}_k + \mathbf{x}_k \Theta(\sigma_k \tau)\Theta(\tau^A \tau^B)\,. \tag{5}$$

  If any component $w_{k,j}$ moves out of the interval $-L, \dots, L$, it is replaced by $\mathrm{sign}(w_{k,j})L$.

Note that for the last rule, the dynamics of each component is identical to a random walk with reflecting boundaries. The only difference to usual random walks is that the dynamics is controlled by the $2K$ global signals $\sigma_k^{A/B}$ which, in turn, are determined by the ensemble of random walks. Two corresponding components of the weights of A and B receive an identical input $x_{k,j}$, hence they move into the same direction if the control signal allows both of them the move. As soon as one of the two corresponding components hits the boundary their mutual distance decreases. This mechanism finally leads to complete synchronization, $\mathbf{w}_k^A(t) = \mathbf{w}_k^B(t)$ for all $t \geq t_{sync}$.

On average, a common step leads to an attractive force between the corresponding weight vectors. If, however, only the weight vector of one of the two partners is changed the distance between corresponding vectors increases, on average. This may be considered as an repulsive force between the corresponding hidden units.

A learning step in at least one of the K hidden units occurs if the two output bits are identical, $\tau^A = \tau^B$. In this case, there are three possibilities for a given pair of hidden units:

- an attractive move for $\sigma_k^A = \sigma_k^B = \tau^{A/B}$;
- a repulsive move for $\sigma_k^A \neq \sigma_k^B$;
- and no move at all for $\sigma_k^A = \sigma_k^B \neq \tau^{A/B}$.

We want to calculate the probabilities for repulsive and attractive steps [8, 13]. The distance between two hidden units can be defined by their mutual overlap

$$\rho_k = \frac{\mathbf{w}_k^A \cdot \mathbf{w}_k^B}{\sqrt{\mathbf{w}_k^A \cdot \mathbf{w}_k^A}\,\sqrt{\mathbf{w}_k^B \cdot \mathbf{w}_k^B}}\,. \tag{6}$$

The probability ϵ_k that a common randomly chosen input $\mathbf{x}_k$ leads to a different output bit $\sigma_k^A \neq \sigma_k^B$ of the hidden unit is given by [2]

$$\epsilon_k = \frac{1}{\pi} \arccos \rho_k\,. \tag{7}$$

The quantity ϵ_k is a measure of the distance between the weight vectors of the corresponding hidden units. Since different hidden units are independent, the values ϵ_k determine also the conditional probability P_r for a repulsive step between two hidden units given identical output bits of the two TPMs. In the case of identical distances, $\epsilon_k = \epsilon$, one finds for $K = 3$

$$\begin{aligned} P_r &= P(\sigma_k^A \neq \sigma_k^B | \tau^A = \tau^B) \\ &= \frac{2(1-\epsilon)\epsilon^2}{(1-\epsilon)^3 + 3(1-\epsilon)\epsilon^2}\,. \end{aligned} \tag{8}$$

On the other side, an attacker E may use the same algorithm as the two partners A and B. Obviously, it will move its weights only if the output bits of the two partners are identical. In this case, a repulsive step between E and A occurs with probability $P_r = \epsilon$ where now ϵ is the distance between the hidden units of E and A.

Note, that for both the partners and the attacker one has the important property that the networks remain identical after synchronization. When one has achieved $\epsilon = 0$ at some time step, the distance remains zero forever, according to the previous equations for P_r. However, although the attacker uses the same algorithm as the two partners, there is an important difference: E can only listen but it cannot influence A or B. This fact leads to the difference in the probabilities of repulsive steps, the attacker has always more repulsive steps than the two partners. For small distances, $\epsilon \ll 1$, the probability P_r increases linear with the distance for the attacker but quadratic for the two partners. This difference between learning and listening leads to a tiny advantage of the partners over an attacker. The subtle competition between repulsive and attractive steps makes cryptography feasible.

On the other side, there is always a nonzero probability P_E that an attacker will synchronize, too [11]. For neural cryptography, P_E should be as small as possible. Therefore it is useful to investigate synchronization for different models and to calculate their properties as a function of the model parameters.

Here we investigate a novel mechanism which decreases P_E, namely we include feedback in the neural networks. The input vectors $\mathbf{x}_k$ are no longer common random numbers, but they are produced by the bits of the corresponding hidden units. Therefore the hidden units of the two partners no longer receive an identical input, but two corresponding input vectors separate with the number of training steps. To allow synchronization, one has to reset the two inputs to common values after some time interval.

For nonzero distance, $\epsilon > 0$, this feedback mechanism creates a sort of noise and increases the number of repulsive steps. After synchronization, $\epsilon = 0$, feedback will produce only identical input vectors and the networks move with zero distance forever [14].

Before we discuss synchronization and several attacking scenarios, we consider the properties of the bit sequence generated by a TPM with feedback.

III. BIT GENERATOR

We consider a single TPM network with K hidden units, as in the previous section. We start with K random input vectors $\mathbf{x}_k$. But now, for each hidden unit k and for each time step t, the input vector is shifted and the output bit $\sigma_k(t)$ is added to its first component [16]. Simultaneously, the weight vector $\mathbf{w}_k$ is trained according to the anti-Hebbian rule Eq. (3). Consequently, the bit sequence $\tau(t)$ generated by the TPM is given by the equation

$$\tau(t) = \prod_{k=1}^{K} \text{sign} \left(\sum_{j=1}^{N} w_{k,j}(t) \sigma_k(t-j) \right) . \qquad (9)$$

Similar bit generators were introduced in [17] and the statistical properties of their generated sequences were investigated [18]. Here we study the corresponding properties for our TPM with discrete weights.

The TPM network has 2^{KN} possible input and $(2L+1)^{KN}$ weight vectors. Therefore our deterministic finite state machine can only generate a periodic bit sequence whose length l is limited by $(4L+2)^{KN}$.

Our numerical simulations show that the average length $\langle l \rangle$ of the period indeed increases exponentially fast with the size KN of the network, but it is much smaller than the upper bound. For $K = 3$ and $L > N$ we find $\langle l \rangle \propto (2.69)^{3N}$, independent of the number L of weight values.

The network takes some time before it generates the periodic part of the sequence. We find that this transient time also scales exponentially with the system size KN. This means that, for sufficiently large values of N, say $N \geq 100$, any simulation of the bit sequence remains in the transient part and will never enter the cycle.

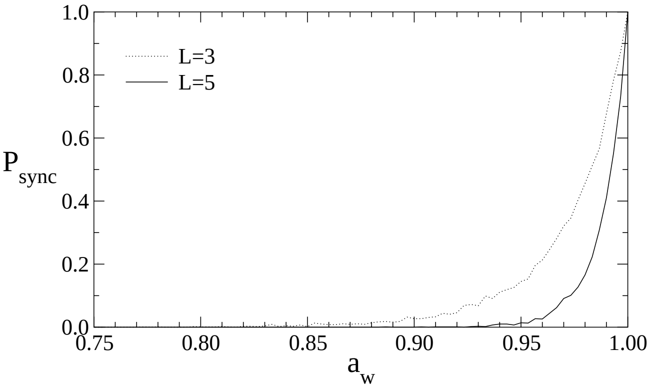

FIG. 2: Probability P_{sync} as a function of the fraction a_w of initially known weights, calculated from 1000 simulations with $K = 3$ and $N = 100$.

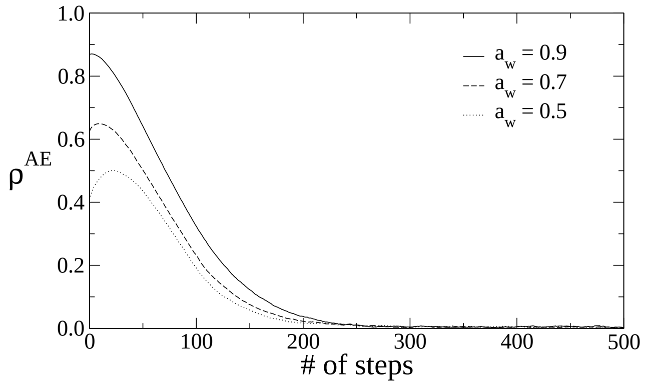

FIG. 3: The average overlap between student and generator as a function of the number of steps for $K = 3$, $L = 5$ and $N = 100$.

The bit sequence generated by a TPM with $K > 2$ cannot be distinguished from a random bit sequence. For $K = L = 3$ we have numerically calculated its entropy and found the value ln 2 as expected from a truly random bit sequence. In addition, we have performed several tests on randomness as described by Knuth [19]. We did not find any correlations between consecutive bits, the bit sequence passed all tests on randomness within strict confidence levels.

Although the bit sequence passed many known tests on random numbers we know that it is generated by a neural network. Does this knowledge help to estimate correlations of the sequence and to predict it? In fact, for a sequence generated by a perceptron (TPM with $K = 1$), another perceptron trained on the sequence could achieve an overlap to the generator [3].

Consider a bit sequence generated by a TPM with the anti-Hebbian rule. Another TPM (the "student") is trained on this sequence using the same rule. In addition, if the output bit disagrees with the corresponding bit of the sequence, we use the geometric method of [8] to perform a training step.

Figures 2 and 3 show that for $K = 3$ hidden units, it is not possible to obtain an overlap to the generating TPM by learning the sequence. Only if the initial overlap between the generator and the student is very large there is a nonzero probability P_{sync} that the student will synchronize with the generator. If it does not synchronize, the overlap between student and generator decays to zero.

Summarizing, a TPM network generates a pseudorandom bit sequences which cannot be predicted from part of the sequence. As a consequence, for cryptographic applications, the TPM can be used to encrypt and decrypt a secret message after it has generated a secret key.

IV. SYNCHRONIZATION

As shown in the previous section, a TPM cannot learn the bit sequence generated by another TPM since the two input vectors are completely separated by the feedback mechanism. This also holds for synchronization by mutual learning: With feedback, two networks cannot be attracted to an identical time dependent state. Hence, to achieve synchronization, we have to introduce an additional mechanism which occasionally resets the two inputs to a common vector. This reset occurs whenever the system has produced R different output bits, $\tau^A(t) \neq \tau^B(t)$. For $R = 0$ we obtain synchronization without feedback, which has been studied previously, and for large values of R the system does not synchronize. Accordingly, we have added a new parameter in our algorithm which increases the synchronization time as well as the difficulty to attack the system. In the following two sections, we investigate synchronization and security of the TPM with feedback quantitatively.

We consider two TPMs A and B which start with different random weights and common random inputs. The feedback mechanism is defined as follows:

(i) After each step t the input is shifted, $x_{k,j}(t+1) = x_{k,j-1}(t)$ for $j > 1$.

(ii) If the output bits agree, $\tau^A(t) = \tau^B(t)$, the output of each hidden unit is used as a new input bit, $x_{k,1}(t+1) = \sigma_k(t)$, otherwise all K pairs of input bits $x_{k,1}(t)$ are set to common public random values.

(iii) After R steps with different output, $\tau^A(t) \neq \tau^B(t)$, all input vectors are reset to public common random vectors, $x^A_{k,j}(t+1) = x^B_{k,j}(t+1)$.

Feedback creates correlations between the weights and the inputs. Therefore the system becomes sensitive to the learning rule. We find that only for the anti-Hebbian rule, Eq. (3), the components of the weights have a broad distribution. The entropy per component is larger than 99% of the maximal value $\ln(2L+1)$. For the Hebbian or random walk rule, the entropy is much smaller, because the values of the weights are pushed to the boundary values $\pm L$. Therefore the network with the anti-Hebbian rule offers less information to an attack than the two other rules.

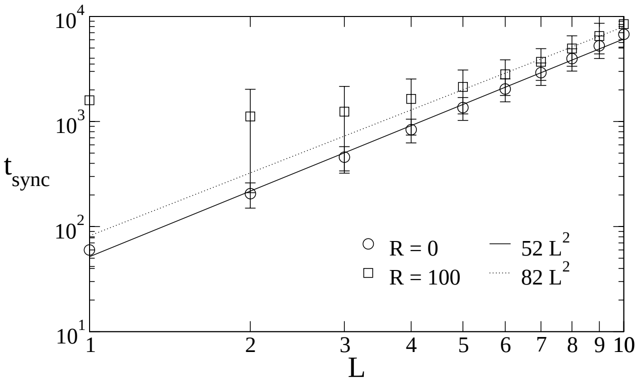

FIG. 4: Average synchronization time t_{sync} and its standard deviation as a function of L, found from 10000 simulation runs with $K = 3$ and $N = 10000$. The line $52L^2$ is a result of linear regression for $R = 0$.

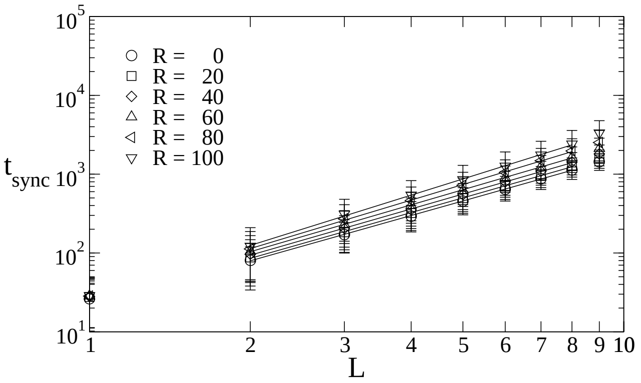

FIG. 5: The synchronization time t_{sync} and its standard deviation as a function of L, averaged over 10000 runs of the iterative equations for $K = 3$.

In Fig. 4 we have numerically calculated the average synchronization time as a function of the number L of components for the anti-Hebbian rule. Obviously, there is a large deviation from the scaling law $t_{sync} \propto L^2$ as observed for $R = 0$. Our simulations for larger values of N, which are not included here, show that there exist strong finite size effects which do not allow to derive a reliable scaling law from the numerical data.

Fortunately, the limit $N \to \infty$ can be performed analytically. The simulation of the KN weights is replaced by a simulation of an $(2L+1) \times (2L+1)$ overlap matrix $f^k_{a,b}$ for each hidden unit k which measures the fraction of weights which are in state a for the TPM A and in state b for B [13, 20].

We have extended this theory to the case of feedback. A new variable $\lambda_k(t)$ is introduced which is defined as the fraction of input components $x_{k,j}$ which are different be-

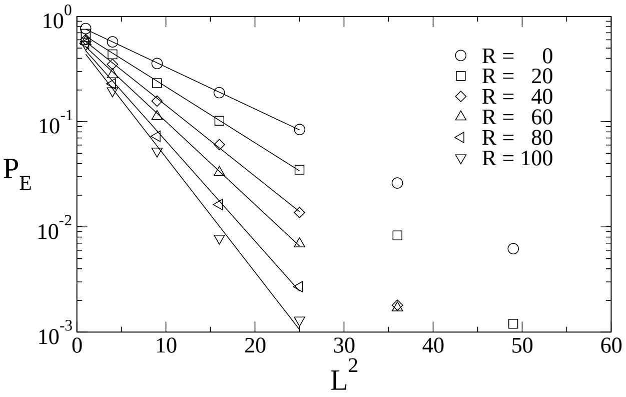

FIG. 6: The success probability P_E as a function of L, averaged over 10000 simulations with $K = 3$ and $N = 1000$.

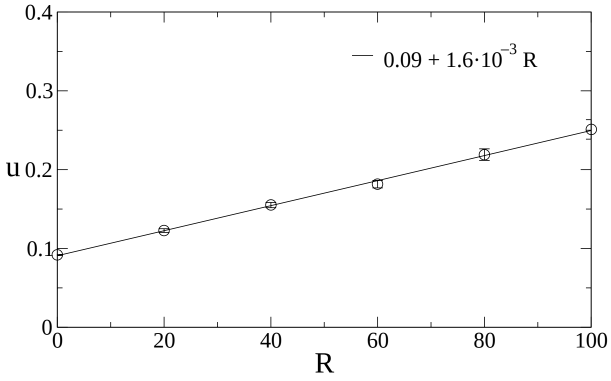

FIG. 7: The coefficient u as a function of the feedback parameter R, calculated from the results shown in Fig. 6.

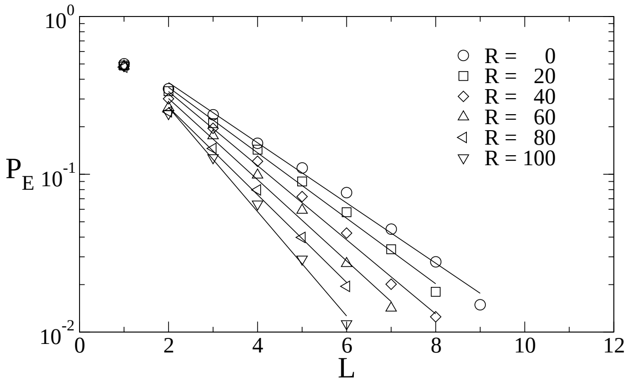

FIG. 8: The success probability P_E as a function of L, found from 10000 runs of the iterative equations for $K = 3$.

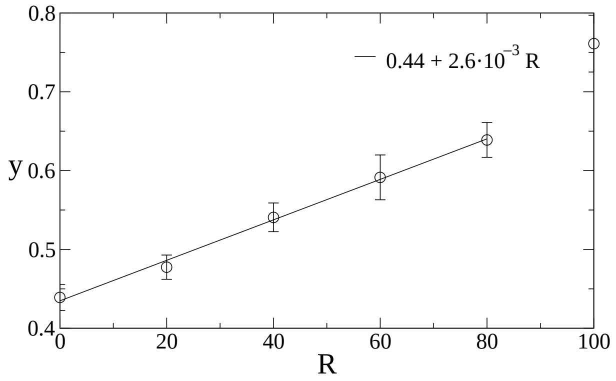

FIG. 9: The coefficient y as a function of the feedback parameter R, calculated from the results shown in Fig. 8.

tween the corresponding hidden units of A and B. This variable changes with time, and it influences the equation of motion for the overlap matrix $f^k_{a,b}(t)$. Details are described in the Appendix.

Figure 5 shows the results of this semi-analytic theory. Now, in the limit of $N \to \infty$, the average synchronization time can be fitted to increase with a power of L, roughly proportional to L^2. The data indicate that only the pre-factor but not the exponent depends on the strength R of the feedback; the pre-factor seems to increase linearly with R.

Hence, if the network is large enough, feedback has only a small effect on synchronization. In the following section we investigate the effect of feedback on the security of the network: How does the probability that an attacker is successful depend on the feedback parameter R?

V. ENSEMBLE OF ATTACKERS

Up to now, the most successful attack on neural cryptography is the geometric attack [8, 11]. The attacker E uses the same TPM with an identical training step as the two partners. That means, only for $\tau^A = \tau^B$ the attacker performs a training step. When its output bit τ^E agrees with the two partners, the attacker trains the hidden units which agree with the common output. For $\tau^E \neq \tau^{A/B}$, however, the attacker first inverts the output bit σ_k for the hidden unit with the smallest absolute value of the internal field and then performs the usual training step.

For the geometric attack the probability P_E that an attacker synchronizes with A and B is nonzero. Consequently, if the attacker uses an ensemble of sufficiently many networks there is a good chance that at least one of them will find the secret key.

We have simulated an ensemble of attackers using the geometric attack for the two TPMs with feedback and anti-Hebbian learning rule. Of course, each attacking network uses the same feedback algorithm as the two partner networks. Figure 6 shows the results of our numerical simulations. The success probability P_E decreases with the feedback parameter R. For the model parameters shown in Fig. 6 we find that P_E can be fitted to an exponential decrease with L^2,

$$P_E \propto e^{-uL^2} \,. \tag{10}$$

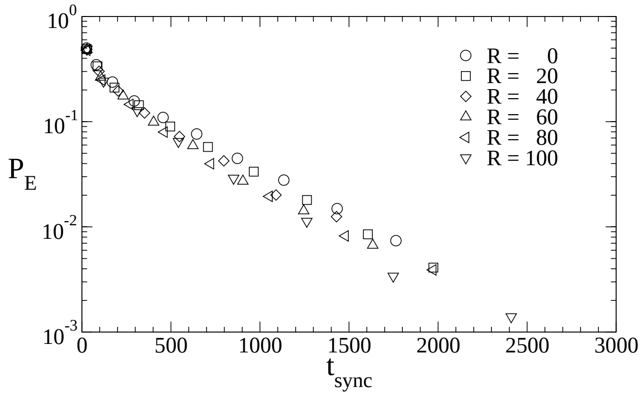

FIG. 10: The success probability P_E as a function of the average synchronization time t_{sync}, calculated from the results shown in Fig. 5 and Fig. 8.

The coefficient u increases linearly with R, as shown in Fig. 7. The scaling [Eq. (10)], however, is a finite size effect. For large system sizes N, the success probability decreases exponentially with L instead of L^2,

$$P_E \propto e^{-yL} \, . \tag{11}$$

This can be seen from the limit $N \to \infty$ which can be performed with the analytic approach of the previous section. Now the dynamics of the system is described by a tensor $f^k_{a,b,e}$ for the three networks A, B and E and corresponding variables $\lambda^A_k, \lambda^B_k, \lambda^E_k$. Details are given in the Appendix.

Figure 8 indicates the exponential scaling behavior [Eq. (11)] for several values of R. The coefficient y increases linearly with R, as shown in Fig. 9.

These results show that feedback improves the security of neural cryptography. The synchronization time, on the other side, increases, too. Does the security of the system improve for constant effort of the two partners?

This question is answered in Fig. 10 which shows the probability P_E as a function of the average synchronization time, again for several values of the feedback parameter R. On the logarithmic scale shown for P_E, the security does not depend much on the feedback. For constant effort to find the secret key, feedback yields a small improvement of security, only.

VI. CONCLUSIONS

Neural cryptography is based on a delicate competition between repulsive and attractive stochastic forces. A new feedback mechanism has been introduced which amplifies the repulsive part of these forces. We find that feedback increases the synchronization time of two networks and decreases the probability of an successful attack.

The numerical simulations up to $N = 10^5$ do not allow to derive reliable scaling laws, neither for the synchronization time not for the success probability. But the limit $N \to \infty$ which can be performed analytically indicates that the scaling laws with respect to the number L of component values are not changed by the feedback, only the respective coefficients are modified. The average synchronization time increases with L^2 while the probability P_E of an successful attack decreases exponentially with L, for huge system sizes N.

Accordingly, the security of neural cryptography is improved by including feedback in the training algorithm. But simultaneously the effort to find the common key rises. We find that for a fixed synchronization time, feedback yields a small improvement of security, only.

After synchronization, the system is generating a pseudorandom bit sequence which passed all tests on random numbers applied so far. Even if another network is trained on this bit sequence it is not able to extract some information on the statistical properties of the sequence. Consequently, the neural cryptography cannot only generate a secret key, but the same system can be used to encrypt and decrypt a secret message, as well.

APPENDIX: SEMI-ANALYTICAL CALCULATION FOR SYNCHRONIZATION WITH FEEDBACK

In this appendix we describe our extension of the semi-analytic calculation [13, 20] to the case of feedback.

The effect of the feedback mechanism depends on the fraction Λ of newly generated input elements $x_{k,j}$ per step and hidden unit. In the numerical simulations presented in this paper Λ is equal to N^{-1}. In this case the effect of the feedback mechanism vanishes in the limit $N \to \infty$. But it is also possible to generate several input elements $x_{k,j}$ per hidden unit and step. For that purpose one can multiply the output bit σ_k with ΛN random numbers $z \in \{-1, +1\}$. As we want to compare the results of the semi-analytical approach with simulations for $N = 1000$, we set $\Lambda = 10^{-3}$ in the following calculations.

In the case of two TPMs the development of the input noise λ_k is given by

$$\lambda_k^+ = (1 - \Lambda)\,\lambda_k + \Lambda\,\Theta(-\sigma^A_k \sigma^B_k)\,\Theta(\tau^A \tau^B) \, . \tag{A.1}$$

At the beginning and after R steps with $\tau^A \neq \tau^B$ all variables λ_k are set to zero (according to the algorithm described in Sec. IV).

The input noise generated by the feedback mechanism affects the output of the hidden units. An input element with $x^B_{k,j} = -x^B_{k,j}$ causes the same output σ^B_k as a change of sign in $w^B_{k,j}$ together with equal inputs for both A and B. Therefore the probability $\epsilon_{k,\text{eff}}$ that two hidden units with overlap ρ_k and input error λ_k disagree on the output bit is given by

$$\epsilon_{k,\text{eff}} = \frac{1}{\pi} \arccos(1 - 2\lambda_k)\rho_k \, . \tag{A.2}$$

The distance $\epsilon_{k,\text{eff}}$ between the hidden units of A and B is used to choose the output bits σ^A_k and σ^B_k with the correct probabilities in each step [13, 20].

The feedback mechanism influences the equation of motion for the overlap matrix $f^k_{a,b}$, too. Here we use additional variables $\Delta^m_k = \Theta(\sigma^m_k \tau^m)\Theta(\tau^A \tau^B)$ to determine if the weights of hidden unit k in the TPM of $m \in \{A, B, E\}$ change ($\Delta^m_k = 1$) or not ($\Delta^m_k = 0$). Therefore we are able to describe the update of elements $f^k_{a,b}$ away from the boundary ($-L < a, b < L$) in only one equation:

$$\begin{aligned} f^{k+}_{a,b} &= \frac{1-\lambda_k}{2} \left(f^k_{a+\Delta^A_k, b+\Delta^B_k} + f^k_{a-\Delta^A_k, b-\Delta^B_k} \right) \\ &+ \frac{1}{2}\lambda_k \left(f^k_{a+\Delta^A_k, b-\Delta^B_k} + f^k_{a-\Delta^A_k, b+\Delta^B_k} \right) . \quad \text{(A.3)} \end{aligned}$$

The second term in Eq. (A.3) which is proportional to λ_k shows the repulsive effect of the feedback mechanism. Similar equations can be derived for elements on the boundary.

In the limit $N \to \infty$ the number of steps required to achieve full synchronization diverges [9]. Because of that one has to define a criterion which determines synchronization in order to analyze the scaling of t_{sync} using semi-analytic calculations. As in [13] we choose the synchronization criterion $\bar{\rho}^{AB} = \frac{1}{3}\sum_{k=1}^{K} \rho^{AB}_k \geq 0.9$.

In order to analyze the geometric attack in the limit $N \to \infty$ one needs to extend the semi-analytical calculation to three TPMs. In this case the development of the input noise is given by the following equations:

$$\begin{aligned} \lambda^{A+}_k &= \Lambda\, \Theta(-\sigma^A_k \sigma^B_k)\, \Theta(-\sigma^A_k \sigma^E_k)\, \Theta(\tau^A \tau^B) \\ &+ (1-\Lambda)\, \lambda^A_k \, , && \text{(A.4)} \\ \lambda^{B+}_k &= \Lambda\, \Theta(-\sigma^B_k \sigma^A_k)\, \Theta(-\sigma^B_k \sigma^E_k)\, \Theta(\tau^A \tau^B) \\ &+ (1-\Lambda)\, \lambda^B_k \, , && \text{(A.5)} \\ \lambda^{E+}_k &= \Lambda\, \Theta(-\sigma^E_k \sigma^A_k)\, \Theta(-\sigma^E_k \sigma^B_k)\, \Theta(\tau^A \tau^B) \\ &+ (1-\Lambda)\, \lambda^E_k \, . && \text{(A.6)} \end{aligned}$$

Analogical to Eq. (A.2) the distance $\epsilon^{mn}_{k,\text{eff}}$ between two hidden units can be calculated from the overlap ρ^{mn}_k and the variables λ^m_k and λ^n_k:

$$\epsilon^{mn}_{k,\text{eff}} = \frac{1}{\pi} \arccos(1 - 2\lambda^m_k - 2\lambda^n_k)\rho^{mn}_k \, . \quad \text{(A.7)}$$

But for the geometric attack the attacker E needs to know the local fields h^E_k. The joint probability distribution of h^A_k, h^B_k and h^E_k is given by [13]

$$P(h^A_k, h^B_k, h^E_k) = \frac{e^{-\frac{1}{2}(h^A_k, h^B_k, h^E_k)\mathcal{C}^{-1}_k (h^A_k, h^B_k, h^E_k)^T}}{\sqrt{(2\pi)^3 \det \mathcal{C}_k}} \, . \quad \text{(A.8)}$$

The covariance matrix in this equation describes the correlations between the three neural networks:

$$\mathcal{C}_k = \begin{pmatrix} Q^A_k & R^{AB}_{k,\text{eff}} & R^{AE}_{k,\text{eff}} \\ R^{AB}_{k,\text{eff}} & Q^B_k & R^{BE}_{k,\text{eff}} \\ R^{AE}_{k,\text{eff}} & R^{BE}_{k,\text{eff}} & Q^E_k \end{pmatrix} . \quad \text{(A.9)}$$

From the tensor $f^k_{a,b,e}$ and the variables λ^m_k one can easily calculate the elements of $\mathcal{C}_k$:

$$\begin{aligned} Q^A_k &= \sum_{a,b,e=-L}^{L} a^2 f^k_{a,b,e} \, , && \text{(A.10)} \\ Q^B_k &= \sum_{a,b,e=-L}^{L} b^2 f^k_{a,b,e} \, , && \text{(A.11)} \\ Q^E_k &= \sum_{a,b,e=-L}^{L} e^2 f^k_{a,b,e} \, , && \text{(A.12)} \\ R^{AB}_{k,\text{eff}} &= (1 - 2\lambda^A_k - 2\lambda^B_k) \sum_{a,b,e=-L}^{L} ab f^k_{a,b,e} \, , && \text{(A.13)} \\ R^{AE}_{k,\text{eff}} &= (1 - 2\lambda^A_k - 2\lambda^E_k) \sum_{a,b,e=-L}^{L} ae f^k_{a,b,e} \, , && \text{(A.14)} \\ R^{BE}_{k,\text{eff}} &= (1 - 2\lambda^B_k - 2\lambda^E_k) \sum_{a,b,e=-L}^{L} be f^k_{a,b,e} \, . && \text{(A.15)} \end{aligned}$$

We use a pseudo random number generator to determine the values of h^A_k, h^B_k and h^E_k in each step. The application of the *rejection method* [21] ensures that the local fields have the right joint probability distribution $P(h^A_k, h^B_k, h^E_k)$. Then the output bits σ^m_k of the hidden units are given by $\sigma^m_k = \text{sign}(h^m_k)$. If $\tau^A = \tau^B \neq \tau^E$ the hidden unit k with the smallest absolute local field $|h^E_k|$ is searched and it's output σ^E_k is inverted (geometric attack). Afterwards the usual training of the neural networks takes place.

The equation of motion for tensor elements $f^k_{a,b,e}$ away from the boundary ($-L < a, b, e < L$) is given by

$$\begin{aligned} f^{k+}_{a,b,e} &= \frac{1 - \lambda^A_k - \lambda^B_k - \lambda^E_k}{2} f^k_{a+\Delta^A_k, b+\Delta^B_k, e+\Delta^E_k} \\ &+ \frac{1 - \lambda^A_k - \lambda^B_k - \lambda^E_k}{2} f^k_{a-\Delta^A_k, b-\Delta^B_k, e-\Delta^E_k} \\ &+ \frac{1}{2}\lambda^A_k\, f^k_{a-\Delta^A_k, b+\Delta^B_k, e+\Delta^E_k} \\ &+ \frac{1}{2}\lambda^A_k\, f^k_{a+\Delta^A_k, b-\Delta^B_k, e-\Delta^E_k} \\ &+ \frac{1}{2}\lambda^B_k\, f^k_{a+\Delta^A_k, b-\Delta^B_k, e+\Delta^E_k} \\ &+ \frac{1}{2}\lambda^B_k\, f^k_{a-\Delta^A_k, b+\Delta^B_k, e-\Delta^E_k} \\ &+ \frac{1}{2}\lambda^E_k\, f^k_{a+\Delta^A_k, b+\Delta^B_k, e-\Delta^E_k} \\ &+ \frac{1}{2}\lambda^E_k\, f^k_{a-\Delta^A_k, b-\Delta^B_k, e+\Delta^E_k} \, . \quad \text{(A.16)} \end{aligned}$$

Similar equations can be derived for elements on the boundary. An attacker is considered successful if one of the conditions $\bar{\rho}^{AE} \geq 0.9$ or $\bar{\rho}^{BE} \geq 0.9$ is achieved earlier than the synchronization criterion $\bar{\rho}^{AB} \geq 0.9$.

[1] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison Wesley, Redwood City, 1991).

[2] A. Engel and C. Van den Broeck, *Statistical Mechanics of Learning* (Cambridge University Press, Cambridge, 2001).

[3] R. Metzler, W. Kinzel, and I. Kanter, Phys. Rev. E **62**, 2555 (2000).

[4] W. Kinzel, R. Metzler, and I. Kanter, J. Phys. A **33**, L141 (2000).

[5] I. Kanter, W. Kinzel, and E. Kanter, Europhys. Lett. **57**, 141 (2002).

[6] W. Kinzel and I. Kanter, J. Phys. A **36**, 11173 (2003).

[7] D. R. Stinson, *Cryptography: Theory and Practice* (CRC Press, 1995).

[8] A. Klimov, A. Mityagin, and A. Shamir, in *ASIACRYPT* (2002).

[9] W. Kinzel and I. Kanter, *Interacting neural networks and cryptography* (2002), cond-mat/0203011.

[10] W. Kinzel and I. Kanter, *Neural cryptography* (2002), cond-mat/0208453.

[11] R. Mislovaty, Y. Perchenok, W. Kinzel, and I. Kanter, Phys. Rev E **66**, 066102 (2002).

[12] I. Kanter and W. Kinzel, in *Proceedings of the XXII Solvay Conference on physics on the physics of communication*, edited by I. Antoniou, V. A. Sadovnichy, and H. Wather (2002), p. 631.

[13] M. Rosen-Zvi, E. Klein, I. Kanter, and W. Kinzel, Phys. Rev. E **66**, 066135 (2002).

[14] A similar mechanism for a noise which increases the security of the network was recently found by combining synchronization of neural networks and chaotic maps. See Ref. [15].

[15] R. Mislovaty, E. Klein, I. Kanter, and W. Kinzel, Phys. Rev. Lett. **91**, 118701 (2003).

[16] E. Eisenstein, I. Kanter, D. Kessler, and W. Kinzel, Phys. Rev. Lett. **74**, 6 (1995).

[17] H. Zhu and W. Kinzel, Neural Computation **10**, 2219 (1998).

[18] R. Metzler, W. Kinzel, L. Ein-Dor, and I. Kanter, Phys. Rev. E **63**, 056126 (2001).

[19] D. E. Knuth, *Seminumerical Algorithms*, vol. 2 of *The Art of Computer Programming* (Addison-Wesley, Redwood City, 1981).

[20] M. Rosen-Zvi, I. Kanter, and W. Kinzel, J. Phys A **35**, L707 (2002).

[21] A. K. Hartmann and H. Rieger, *Optimization algorithms in physics* (Wiley-VCH, Berlin, 2002).