Mutual learning in a tree parity machine and its application to cryptography

Michal Rosen-Zvi[1], Einat Klein[1], Ido Kanter[1] and Wolfgang Kinzel[2]
[1]*Minerva Center and Department of Physics, Bar-Ilan University, Ramat-Gan, 52900 Israel, and*
[2]*Institut für Theoretische Physik, Universität Würzbur, Am Hubland 97074 Würzbur, Germany*

Mutual learning of a pair of tree parity machines with continuous and discrete weight vectors is studied analytically. The analysis is based on a mapping procedure that maps the mutual learning in tree parity machines onto mutual learning in noisy perceptrons. The stationary solution of the mutual learning in the case of continuous tree parity machines depends on the learning rate where a phase transition from partial to full synchronization is observed. In the discrete case the learning process is based on a finite increment and a full synchronized state is achieved in a finite number of steps. The synchronization of discrete parity machines is introduced in order to construct an ephemeral key-exchange protocol. The dynamic learning of a third tree parity machine (an attacker) that tries to imitate one of the two machines while the two still update their weight vectors is also analyzed. In particular, the synchronization times of the naive attacker and the flipping attacker recently introduced in [9] are analyzed. All analytical results are found to be in good agreement with simulation results.

PACS numbers: 87.18.Sn, 89.70.+c

I. INTRODUCTION

Artificial neural networks are known for their ability to learn [1, 2]. They produce an output from a given input according to some weight vector and a transfer function. Traditionally, there are two types of learning. One type is unsupervised learning where a network receives input and tries to learn about the input distribution. The other type is the teacher-student scenario, when the so-called teacher receives inputs, produces outputs and gives another machine, the so-called student, both the inputs and their assigned outputs. In such a scenario the teacher is static, i.e., its weight vector does not change during the learning, and the student tries to imitate the teacher so as to produce the same output in a new unknown example by dynamically updating its weight vector. The state in which the student achieves the same weight vector as that of the teacher and can therefore perform the same output as that of the teacher is referred to as perfect learning.

During the last few years a new type of learning scenario has been introduced and is under discussion: the *mutual learning* procedure. In the mutual learning procedure there is no distinction between the teacher role and the student role; both networks function the same way. They receive inputs, calculate the outputs and update their weight vector according to the match between their mutual outputs [3, 4]. This is an online learning procedure where in each step one input vector is given, the output in both machines is calculated and the resulting increment of each weight vector is added accordingly. It was found that perceptrons that undergo *mutual learning* might end up in a synchronized state when the weight vectors of both machines are either parallel - exactly the same, or anti-parallel - exactly the opposite (depending on their specific updating rule). The stationary synchronized solution is equivalent to the stationary perfect learning solution in the teacher-student scenario. We extend the analysis of mutual learning between perceptrons to mutual learning between parity machines . We introduce a generic method of analyzing mutual learning in feedforward tree multi-layer networks where we concentrate on the tree parity machine (TPM)[5, 6, 7]. The method is based on a mapping procedure that maps the mutual learning in TPMs onto mutual learning in noisy perceptrons.

A novel cryptosystem composed of two parity machines that synchronize has recently attracted much attention [8, 9, 10, 11]. A host of simulation results show that discrete TPMs can synchronize very fast and a third machine that tries to learn their weight vector achieves only partial success. These properties make mutual learning in TPMs attractive for applications in secure communications, as an information-bearing message can be hidden within a complicated structure of the TPM's weight vectors and still be reconstructed at the receiver using another TPM whose parameters are exactly matched to those of the first one. This type of cryptosystem can provide a new basis for security much different from currently used cryptosystems that involve large integers and are based upon number theory [12].

The discrete machines studied carried out an updating procedure different from the conventional learning procedures analyzed in neural networks. In the discrete machine procedure the increment of the weight vector in each step is finite and not infinitesimally small. Since the methods of analyzing discrete on-line learning in contemporary research, see [13, 14, 15, 16, 17], are not applicable to this case, we introduce here a novel method for analyzing mutual learning in networks with discrete weight vectors and a learning process that is based on a finite increment. First, we describe mutual learning with discrete *perceptrons*, and then we exploit the method of mapping mutual learning between TPMs onto mutual learning between noisy perceptrons and analyze mutual learning in discrete TPMs.

In cryptography, one of the most important aspects of the channel is its security. Therefore, potential algorithms of eavesdroppers are included in our analysis. Such algorithms are actually sophisticated learning procedures where the parties are the teachers and their weights are time dependent, and the eavesdropper is the student. In the following we name this time-dependent-teacher-student scenario *dynamic learning.*

In this Paper we analyze mutual learning and dynamic learning in TPMs of two kinds: machines with continuous weight vectors (the spherical constraint - see Eq. (2) below) and with discrete weight vectors and finite increment (see Eq. (3) below). We introduce a method that maps mutual learning in two layered parity machines onto mutual learning in noisy perceptrons. The spherical tree parity machine is studied using the same tool box used for studying mutual learning in the perceptron [3]. The interesting behavior of full synchronization for a certain regime in the learning rate space and partial synchronization in the other regime is also found in the mutual learning of TPMs. Mutual learning in a TPM when the weight vectors are continuous is described by equations of motion that reveal the evolution of the order parameters in time. The derivation of the equations of motion is based on the assumption that the order parameters are self-averaging quantities [18, 19]. This assumption is violated when the increment of the weight vectors in each step is finite and not infinitesimally small, as in the case of the discrete weight vector studied here. Therefore we develop different analytical tools for the case of discrete weight vectors.

This Paper is an extension of [10]. It contains a full, detailed description of the analytical methods and discussions that were not included in [10]. An advanced attack suggested recently by Shamir et al [9] - the flipping attack - is also analyzed. The paper is organized as follows: in section II we introduce the TPM model. We employ a general framework to present its application to Cryptography in II A. The dynamics studied are presented in II B and the order parameters and local field distributions are discussed in II C. The mapping procedure is detailed in III. The learning in continuous TPMs is given in IV, where we divided the section into mutual learning (section IV A), and dynamic learning (section IV B). The section is summarized and the results are discussed in IV C. Discrete learning is presented in section V. We first describe mutual learning in perceptrons in V A. The extension to mutual learning in parity machines is given in V B. Two dynamic learning attacks are studied, the naive attacker (in V C), and the flipping attacker (in V D). A discussion and an overview are given in V E. All analytical results are found to be in good agreement with simulation results as indicated in each section.

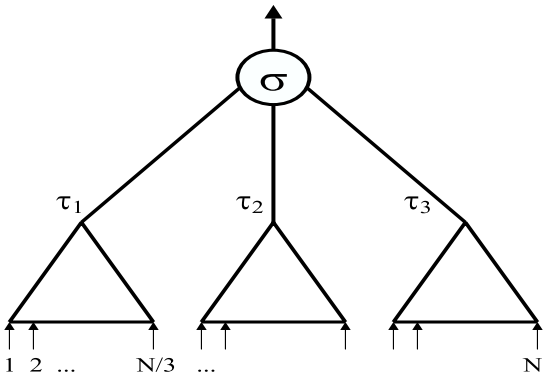

Figure 1: A tree parity machine $N:3:1$

II. THE MODEL

We consider a TPM with K binary hidden units $\tau_i = \pm 1, \quad i = 1, ..., K$ feeding a binary output, $\sigma = \prod_{i=1}^{K} \tau_i$, see Figure 1. The networks consist of either a continuous or a discrete coupling vector $\mathbf{w}_i = W_{1i}, ..., W_{Ni}$ and disjointed sets of inputs $\mathbf{x}_i = X_{1i}, ..., X_{Ni}$ containing N elements each. The input elements are random variables with zero mean and unit variance. We confine the input components to $x_{ji} = \pm 1$ without losing generality. The local field in the ith hidden unit is defined as

$$h_i = \frac{1}{\sqrt{N/3}} \mathbf{w}_i \mathbf{x}_i, \tag{1}$$

and the output in the ith hidden unit is derived by taking the sign of the local field. The output of the tree parity machine is therefore given by

$$\sigma = \prod_{i=1}^{K} sign(h_i) = \prod_{i=1}^{K} \tau_i.$$

Our analysis is limited to TPMs with three hidden units, $K = 3$, merely for simplicity of the representation of the analysis. The extension of the formalism to any number of hidden units is straightforward.

The weight vectors of the TPMs are initiated at random according to a certain constraint. We studied two different cases: the case when the weight vectors are confined to a sphere,

$$\sum_{j=1}^{N} W_{ji}^2 = N, \tag{2}$$

and are initiated randomly according to a Gaussian distribution; and the case when there are a finite number

of available integer values that each component of the weight vector can take,

$$W_{ji} = \pm L, \pm(L-1), ..., \pm 1, 0, \tag{3}$$

and the weight vector components are initiated at random from a flat distribution with equal probability for each value. These two scenarios are referred to as the continuous case and the discrete case.

We studied the mutual and dynamic learning of such TPMs in various scenarios where the initial random selected weight vector is the unknown secret information. Two machines A and B, perform mutual learning and try to synchronize by updating their weights according to the match between their output such that at the end they achieve full synchronization. The third machine, C, performs dynamic learning by trying to learn the weight vectors of one of the two machines, say A, and uses an attack strategy to update its weight vectors such that at the end of the procedure they will be identical to the weight vector of player A. The application of these procedures to the field of Cryptography is discussed in the following section.

A. Cryptography Based on Synchronization: General Framework

Before we develop the detailed equations for mutual learning in TPMs, we introduce the general concept of synchronization and learning in discrete parity machines in terms of a mean-field-like approach, and discuss the qualitative ability to construct an ephemeral key-exchange protocol based on mutual learning between TPMs.

First, let us consider two parties A and B who wish to agree on a secret key over a public channel. The weight vectors, $\mathbf{w}_i^{A/B}$, are the parameters of each unit which are changed during the training procedure. Both parties start with secret initial parameters $\mathbf{w}$ which may be generated randomly. After a number of training steps, the set of parameters is synchronized and becomes the *time-dependent* common key. At each training step a common random input $\mathbf{x}_i$ is generated for both of the parties; it is public and known to possible eavesdroppers.

Each party of the secure channel consists of three hidden units with corresponding three parameter vectors. For a given input $\mathbf{x}_i$ each party calculates an output bit $\sigma^{A/B}$ and sends it over the public channel. A training step is performed only if the two output bits disagree and only for the hidden units which agree with their output

$$\Delta \mathbf{w}^{A/B} = g\left(\sigma^{A/B}\mathbf{x}_i\right)\theta\left(-\sigma^A\sigma^B\right)\theta\left(\sigma^{A/B}\tau_i^{A/B}\right), \tag{4}$$

where g is an odd function. As an example consider the following configuration of the hidden units: $+++$ for TPM A and $-++$ for TPM B. The output bits have the values $\sigma^A = 1, \sigma^B = -1$. Hence A trains all of its units according to $\mathbf{x}_i$, while B changes only the weight vector of its first unit according to $-\mathbf{x}_i$.

Synchronization between the two machines indicates a full anti-parallel state where each machine produces exactly the opposite output of the other for any given input. The success of synchronization can be measured by the probability of an incoherent state, i.e., the probability of having the same output instead of the opposite one. The probability for an *incoherent state*, ϵ^{in}, that two corresponding hidden units are mistaken and instead of producing exactly the opposite output they agree on a random input, is given by

$$\epsilon^{in} = Prob\left(\tau_i^A\left(\mathbf{x}_i, \mathbf{w}_i^A\right) = \tau_i^B\left(\mathbf{x}_i, \mathbf{w}_i^B\right)\right). \tag{5}$$

The function g used for training must be chosen so that on the average (over random input) ϵ^{in} is decreased. In this section we simplify the presentation by assuming symmetry among the three hidden unit, $\epsilon_i^{in} = \epsilon^{in}$. The full detailed description of the dynamical process beyond this mean-field-like framework is given in V.

It is now easy to see that as soon as the TPMs are synchronized they will remain synchronized, i.e., if $\mathbf{w}_i^A = -\mathbf{w}_i^B$ for all i, then $\sigma^A = -\sigma^B$ and will remain so. A training step in a unit i is performed only if both output bits disagree and if the two τ_i disagree accordingly. Hence, after the synchronization state is achieved they either perform a coherent training step or they do not change their parameters (referred to as a quiet step). A pair of synchronized hidden units performs a kind of random walk in parameter space but remains synchronized.

This is different when the two hidden units are not identical. Let us consider the *first* hidden unit, where there are four distinct cases:

(a) $\sigma^A = \sigma^B$: nothing moves and the next step is performed.

(b) $\tau_1^A = \sigma^A$, $\tau_1^B = \sigma^B$, $\sigma^A = -\sigma^B$: both parameter vectors $\mathbf{w}_1^A$ and $\mathbf{w}_1^B$ are coherently changed.

(c) $\tau_1^A = \sigma^A$, $\tau_1^B \neq \sigma^B$, $\sigma^A = -\sigma^B$ or $\tau_1^A \neq \sigma^A$, $\tau_1^B = \sigma^B$, $\sigma^A = -\sigma^B$: only one parameter vector is changed and moves incoherently, hence ϵ_1^{in} increases.

(d) $\tau_1^A \neq \sigma^A$, $\tau_1^B \neq \sigma^B$, $\sigma^A = -\sigma^B$: both parameter vectors are not changed.

The probability of finding these four cases can be calculated from the knowledge of ϵ^{in}. For example, the probability of finding the configuration shown above, $+++$ and $-++$, is $\frac{1}{8}\left(1-\epsilon^{in}\right)\left(\epsilon^{in}\right)^2$. All 64 configurations can be divided into three categories: the probability of having an attractive step, p_a (case (b)); the probability of having a repulsive step, p_r (case (c)); or the probability of having a quiet step, p_q (cases (a) and (d)). These probabilities are found to be

$$p_a = \frac{1}{2}\left[\left(1-\epsilon^{in}\right)^3 + \left(1-\epsilon^{in}\right)\left(\epsilon^{in}\right)^2\right], \tag{6}$$

$$p_r = 2\left(1-\epsilon^{in}\right)\left(\epsilon^{in}\right)^2,\ p_q = 1 - p_a - p_r.$$

In the remainder of this section the three probabilities above are employed in order to explain the synchroniza-

tion phenomenon, and to demonstrate the superiority of the synchronization process over a possible attacker that also tries to synchronize with A and B.

Close to synchronization, $\epsilon^{in} \sim 0$, the probability of having a repulsive step is proportional to $p_r \sim \left(\epsilon^{in}\right)^2$ whereas the probability of having an attractive step is $p_a \sim \frac{1}{2}$ (quiet steps are always possible). Let us assume that the change of the error, ϵ^{in} depends only on a function of ϵ^{in} itself. Later we will derive the exact equations, which are more complex. Then, the average change in ϵ^{in} in one step is obtained by

$$\Delta\epsilon = a\left(\epsilon^{in}\right) p_a - r\left(\epsilon^{in}\right) p_r. \tag{7}$$

Close to synchronization a repulsive step affects all of the parameters while an attractive step can only synchronize the few parameters which are not yet identical. Hence we expect for small values of ϵ^{in}:

$$a\left(\epsilon^{in}\right) \sim a_0 \epsilon^{in}, \;\; r\left(\epsilon^{in}\right) \sim r_0. \tag{8}$$

Therefore, in the leading order one obtains $\Delta\epsilon \propto a_0 \epsilon^{in}$. Close to synchronization the attractive force is dominate, independent of the detailed mechanism of learning. The parity machine suppresses the repulsive steps by reducing their appearance frequency.

This relation does not hold for the committee machine which maps the hidden units to their majority vote, $\sigma = sign\left(\tau_1 + \tau_2 + \tau_3\right)$ [20, 21]. For this case one finds

$$p_a = \frac{3}{4}\left(1-\epsilon^{in}\right)^3 + \left(1-\epsilon^{in}\right)^2\left(\epsilon^{in}\right) + \frac{1}{2}\left(1-\epsilon^{in}\right)\left(\epsilon^{in}\right)^2, \tag{9}$$

$$p_r = \frac{1}{2}\left(1-\epsilon^{in}\right)^2\left(\epsilon^{in}\right) + \left(1-\epsilon^{in}\right)\left(\epsilon^{in}\right)^2.$$

Now, close to synchronization $p_r \sim \epsilon^{in}$ and repulsion and attractive forces are of the same order, Eq. (7). This competition between attraction and repulsion supports possible attackers, as discussed below.

Let us go back to the parity output and consider an attacker C who knows all the details of the algorithm and can listen to the communication between A and B. We know that the initial configurations of the parameters of A and B are unknown. The attacker C has the same architecture (TPM), the same number of hidden units (3) and uses the same learning algorithm, Eq. (4). What is a good algorithm for C to synchronize, i.e., to learn A and to be anti-parallel to B? If C is synchronized then she should remain so. Hence she should use the identical training step in case of agreement with A. Let us consider an attacker C who simulates party A after synchronization between A and B is achieved. C uses the complete algorithm explained above for party A. This means that A always makes some moves of her parameters while C moves her parameters corresponding to the units whose output bit τ_i^C are identical to σ^A (in the following we named this attack *the naive attack* - see V C). This strategy for C generates many repulsion steps between C and A. In fact, assuming the error between all matching units is the same, $\epsilon^{in} = Prob\left(\tau_i^C \neq \tau_i^A\right)$ (where we use the same symbol for ϵ^{in} as in Eq. (5), although seemingly different, in both cases it refers to the error, see II C and Eq. (17) below) and summing up all possibilities yields

$$p_a = \frac{1}{2}\left(1-\epsilon^{in}\right)^3 + \frac{1}{2}\left(1-\epsilon^{in}\right)\left(\epsilon^{in}\right)^2 + \left(1-\epsilon^{in}\right)^2\epsilon^{in}, \tag{10}$$

$$p_r = \left(1-\epsilon^{in}\right)^2\epsilon^{in} + 2\left(1-\epsilon^{in}\right)\left(\epsilon^{in}\right)^2 + \left(\epsilon^{in}\right)^3.$$

The essential difference between party A and attacker C is that the probability of finding a repulsive step scales with $\left(\epsilon^{in}\right)^2$ in the mutual learning between A and B and scales with ϵ^{in} in the dynamic learning between C and A, close to synchronization. A and B react to their mutual output while C cannot influence A; this yields a different behavior for small values of the error ϵ^{in}.

The full scheme of the ratio, p_r/p_a, derived from Eqs. (6) and (10) as a function of ϵ^{in} is presented in Figure 2. It is clear that at any value of ϵ^{in} the performance of the mutual learning is better than the performance of the naive attacker that performs many more repulsive moves compared to hers attractive moves. Therefore, a more sophisticated attacker was recently suggested in [9] - the flipping attacker. Hers performance cannot be measured in the scope of this general framework since hers strategy depends on the local fields in the hidden units and therefor can not be included under the rubric of Eq. (4), where g depends only on $\sigma \mathbf{x}_i$.

In the following, before delving into details we introduce the dynamic (Eq. (4)) more specifically. We discuss some of the relevant order parameters and their distributions. We present the strategy of the flipping attacker and an intuitive explanation for her success.

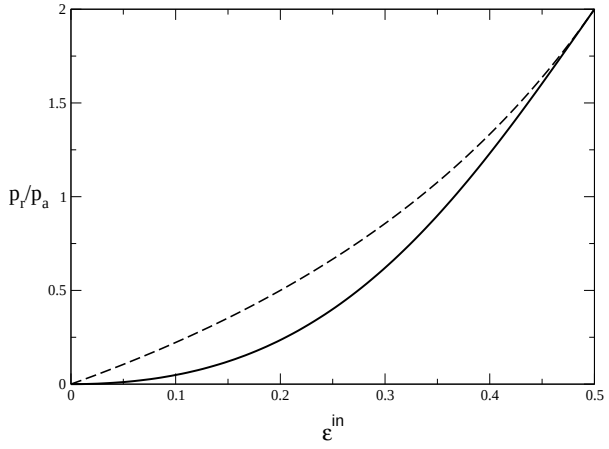

Figure 2: The ratio between p_r and p_a as a function of ϵ^{in} in the case of mutual learning in TPMs, Eq. (6) (solid line) and in case of the naive attack, Eq. (10) (dashed line).

B. The Dynamics

In principle, one can consider the following classes of dynamics that lead to a synchronized state:

(A) The parties update their weight vectors whenever their outputs mismatch ($\sigma^A \neq \sigma^B$, as appears in Eq. (4)), and each unit updates according to the input multiplied by the opposite of its output.

(B) The parties update their weight vectors whenever their outputs mismatch ($\sigma^A \neq \sigma^B$, as appears in Eq. (4)), and each unit updates according to the input multiplied by its output.

(C) The parties update their weight vectors whenever their outputs match ($\sigma^A = \sigma^B$), and each unit updates according to the input multiplied by the opposite of its output.

(D) The parties update their weight vectors whenever their outputs match ($\sigma^A = \sigma^B$), and each unit updates according to the input multiplied by its output.

In all the dynamics mentioned above, the ith hidden unit is updated only if it matches the overall output in that party, if $\tau_i = \sigma$. The two parties that try to synchronize might end up in an anti-parallel state (cases (A) and (B)), or in a parallel state (cases (C) and (D)). Although Eq. (4) does not describe cases (C) and (D), the discussion in section II A is relevant to all cases.

In this Paper we introduce a detailed presentation of case (A). In each step an update is made only if both machines, A and B, disagree, $\sigma_A \neq \sigma_B$, and each unit updates according to the input multiplied by the opposite of its output. In the spherical case we normalize the weight vector after each updating such that its norm does not change. The dependence of the weight vector in a new step on the weight vector in the former one in the continuous case is

$$\mathbf{w}_i^{A+} = \frac{\mathbf{w}_i^A + \frac{\eta}{N}\mathbf{x}_i\theta(-\sigma^A\sigma^B)\theta(\sigma^A\tau_i^B)\sigma^B}{\left\|\mathbf{w}_i^A + \frac{\eta}{N}\mathbf{x}_i\theta(-\sigma^A\sigma^B)\theta(\sigma^A\tau_i^B)\sigma^B\right\|}, \quad (11)$$
$$\mathbf{w}_i^{B+} = \frac{\mathbf{w}_i^B + \frac{\eta}{N}\mathbf{x}_i\theta(-\sigma^A\sigma^B)\theta(\sigma^B\tau_i^B)\sigma^A}{\left\|\mathbf{w}_i^B + \frac{\eta}{N}\mathbf{x}_i\theta(-\sigma^A\sigma^B)\theta(\sigma^B\tau_i^B)\sigma^A\right\|},$$

where $\theta(y)$ is the Heavyside function, i.e., equals zero for $y < 0$ and 1 otherwise, η is the learning rate and $i = 1, ..., K$. The analysis of the dynamic is in the thermodynamic limit where $N \to \infty$ and the weight vectors are updated by an infinitely small quantity in each step.

In the discrete scenario, the update is made in a similar manner, yet there are two important differences from the dynamics point of view. One is that in each step the vectors' components are changed to the next integer value and not by an infinitesimally small one as in the continuous case (Eq. (11)). The second difference is that when there is an update, the components which have reached the boundary value $W_i = \pm L$, and their absolute value should be increased $W_i^+ = \pm(L+1)$, are not changed, and remain with the boundary value. Mathematically, the learning is phrased as follows

$$\mathbf{w}_i^{A+} = \mathbf{w}_i^A + D(\mathbf{w}_i^A\mathbf{x}_i\sigma^B)\mathbf{x}_i\sigma^A\theta(\sigma^A\tau_i^A)\theta(-\sigma^A\sigma^B), \quad (12)$$
$$\mathbf{w}_i^{B+} = \mathbf{w}_i^B + D(\mathbf{w}_i^B\mathbf{x}_i\sigma^A)\mathbf{x}_i\sigma^A\theta(\sigma^B\tau_i^B)\theta(-\sigma^A\sigma^B),$$

where $D(y) = 1 - \delta_{L,y}$ and δ is the Kronecker delta function.

C. Order Parameters and Joint Probability Distributions

The analysis of learning in neural networks with an infinite number of weight vector components is based upon statistical mechanics analysis of several order parameters. The standard order parameters used are

$$Q_i^m = \frac{1}{N/3}\mathbf{w}_i^m \cdot \mathbf{w}_i^m, \quad (13)$$
$$R_i^{m,n} = \frac{1}{N/3}\mathbf{w}_i^m \cdot \mathbf{w}_i^n,$$

where the index i represents the ith hidden unit, $i = 1, ..., K$ and m, n denote the specific party, $m, n \in \{A, B, C\}$. The angle between each pair of weight vectors θ, is given by the normalized overlap between the weight vectors

$$\rho_i^{m,n} = \cos\theta_i^{m,n} = \frac{\mathbf{w}_i^m \cdot \mathbf{w}_i^n}{\|\mathbf{w}_i^m\|\,\|\mathbf{w}_i^n\|}. \quad (14)$$

We assume that there are no direct correlations between different hidden units due to the tree architecture and therefore the overlaps between different units is zero.

In the framework of statistical mechanics analysis of on-line learning the order parameters play an important role in taking the averages over the random inputs, or equivalently over the local field distribution. According to the central limit theorem, the joint probability distribution of the local fields in each triplet of matching hidden units taken from the three different machines depends only on the set of order parameters, $P(h^A, h^B, h^C | \{R, Q\})$ (where we omitted the subscript i from all parameters) and can be found from the correlation matrix. When all weight vectors are normalized, $Q^m = 1$, it is found to be

$$P = \frac{\exp(-\frac{F}{2E})}{(2\pi)^{3/2}\sqrt{E}}, \quad (15)$$

where $F = \left(h^C\right)^2 G^C + \left(h^A\right)^2 G^A + \left(h^B\right)^2 G^B - 2h^A h^B D^C - 2h^A h^C D^B - 2h^C h^B D^A$, $E = 1 - \left(\rho^{A,B}\right)^2 - \left(\rho^{A,C}\right)^2 - \left(\rho^{B,C}\right)^2 + 2\rho^{A,B}\rho^{A,C}\rho^{B,C}$, $G^k = (1 - \rho^{l,m})^2$, $D^k = \rho^{l,m} - \rho^{k,m}\rho^{k,l}$ and $k, l, m \in \{A, B, C\}$. This complicated expression can be much simplified if we assume that the two machines, A and B, are already anti-parallel. In that case the joint probability distribution of the local

fields is given by

$$P = \frac{e^{-\frac{1}{2}\frac{\left(h^C\right)^2+\left(h^A\right)^2-2h^Ah^C\rho^{A,C}}{1-(\rho^{A,C})^2}}}{2\pi\sqrt{1-\rho^{A,C}}}\delta(h^A+h^B), \qquad (16)$$

where $\delta()$ stand for the Dirac delta function.

At this stage it is possible to calculate the probabilities defined in section II A and to show that indeed ϵ^{in} has the same meaning and the same dependency on ρ in the two cases: Eq. (5) and later when the attacker is introduced. Averaging over the local field distributions results in the case of mutual learning in $\epsilon^{in} = 1 - \frac{1}{\pi}\cos^{-1}\rho^{A,B}$ and in the case of dynamic learning we find $\epsilon^{in} = \frac{1}{\pi}\cos^{-1}\rho^{A,C}$. In order to compare these two errors, where in the first one learning is described by negative ρ and in the second by positive, we define $\bar{\rho} = |\rho^{A,B}| = |\rho^{A,C}|$. Substituting $\bar{\rho}$ into both functions above, we get

$$\epsilon^{in} = \frac{1}{\pi}\cos^{-1}\bar{\rho}. \qquad (17)$$

We present in this Paper a flipping attacker, which makes use of the absolute value of the local field. The attacker estimates that the unit with the smallest absolute local field is the one that is most probably wrong - that has different outputs, $\tau_i^C \neq \tau_i^A$. The origin of this assumption can be easily explained by averaging over the local field distribution. The average of the absolute value of the local field, $\langle|h^C|\rangle$, given an overlap $\rho^{A,C}$ between two matching hidden units and norm Q^C of the weight vector in this unit is found to be

$$\langle|h^C|\rangle = \frac{1}{2}\sqrt{\frac{Q^C}{2\pi}}\left(1 \pm \rho^{A,C}\right), \qquad (18)$$

where the sign in the right hand-side of the equation is plus for agreement between the outputs and minus for disagreement. Since ρ varies between -1 and 1 and in a state of partial learning $0 < \rho < 1$, a small absolute local field signals a mistake in the unit's output. The flipping attacker uses this knowledge in her learning procedure, as discussed in section V D.

The analytical study of this attacker includes averages over probability distribution of the local field in the third party, the attacker C, given the local fields of the two machines. This probability is given by

$$P(h^C|h^B,h^A,\{\rho,Q\}) = \frac{P(h^C,h^B,h^C|\{\rho,Q\})}{P(h^A,h^B|\{\rho,Q\})} \qquad (19)$$

where $P(h^C,h^B,h^C|\{\rho,Q\})$ and $P(h^C,h^B|\{\rho,Q\})$ are the joint probability distributions of the three local fields and two local fields respectively, and they are derived from the correlation matrix similar to Eq. (15).

In the discrete case, when the increment is finite (see for instance Eq. (12)), the above order parameters no longer suffice for the macroscopical description of the dynamics even in the thermodynamic limit, $N \to \infty$. However, the distributions above do hold. The dynamic cannot be analyzed with the standard equations of motion based on differential equations of the order parameters with respect to α, the number of examples per input dimension. We introduce a generic method for analyzing the discrete case by extending the macroscopical parameters and deriving macro-dynamical updating equations (see section V).

III. MAPPING PROCEDURE

One can map mutual learning in the parity case onto mutual learning in K perceptrons. The mapping to noisy perceptron introduced for analyzing on-line learning in TPM [22] is inadequate in the case of *mutual* learning where the updating depends on the matching between the outputs but is independent of their specific sign. Nevertheless, a different mapping from TPM to noisy perceptrons can be used for the mutual learning case. The mapping presentation is much simplified in the continuous case since assuming random initial conditions to all hidden units results in the same overlap for all hidden units, $\rho_i = \rho\ \forall i$. Therefore, we first assume that all the overlaps between matching hidden units are the same. Hence, updating K perceptrons is equivalent to one updating in the TPM. The presentation of the mapping below is simplified by the restriction of $K = 3$ and the generalization to any K is straightforward.

We have TPMs that consist of non-overlapping receptive fields with random inputs. Hence in each of the TPMs all 8 internal representations appear with equal probability. A specific hidden unit is updated when the following two conditions are fulfilled; (a) there is a mismatch between the results of the two TPMs, and (b) the state of the hidden unit is the same as the output of the TPM. We make use of ϵ, the probability of having different results in the two hidden units that the overlap between them is ρ and is given by

$$\epsilon = \frac{1}{\pi}\cos^{-1}\rho. \qquad (20)$$

We concentrate on a specific pair of matched hidden units. Given that the outputs of the hidden units are different, there is a probability, P_1, that the TPMs results are different and in one *half* of the cases the TPM output has the same output as its hidden unit and therefore both hidden units in both machines are updated. This probability is given by

$$P_1 = P(\sigma^A \neq \sigma^B|\tau_i^A \neq \tau_i^B) = \epsilon^2 + (1-\epsilon)^2. \qquad (21)$$

Similarly, the probability that there is a mismatch between the two TPMs given that there is agreement between two hidden units, is given by

$$P_2 = P(\sigma^A \neq \sigma^B|\tau_i^A = \tau_i^B) = 2\epsilon(1-\epsilon). \qquad (22)$$

In this case only one of the hidden units has the same sign as the output in its TPM and only that hidden unit is updated.

These probabilities are introduced into the updating procedure of the hidden units - the perceptrons. In the continuous case they affect the form of the equations of motion (see Eq. (23)). In the discrete case they are introduced in a different manner, as described in section V.

IV. CONTINUOUS TREE PARITY MACHINES

Counting on the mapping procedure described above, mutual and dynamic learning in continuous TPMs can be mapped onto learning scenarios in continuous perceptrons. The updating rule can be redefined so that it will be suitable for a perceptron where the kind of updating depends on the above probabilities, P_1 and P_2, Eqs. (21) and (22). The standard on-line equations consist of an average over the order parameters [2], and now contain additional random variables. The average over these additional variables is taken by introducing auxiliary random parameters, as described in the following section.

A. Anti-Parallel Learning

In this scenario the updating rules of the TPMs are given in Eqs. (11) where we have three hidden units, $K = 3$. Mapping the rules onto a perceptron learning by employing the probabilities above is done by introducing auxiliary random parameters, p_α, p_β, p_γ, which are equally distributed between 0 and 1. The updating rule is calculated as a function of these parameters in the following manner,

$$\mathbf{w}^{A+} = \frac{\mathbf{w}^A + \frac{\eta}{N}\mathbf{x}\tau^B\Delta_A}{|\mathbf{w}^A + \frac{\eta}{N}\mathbf{x}\tau^B\Delta_A|},\ \mathbf{w}^{B+} = \frac{\mathbf{w}^B + \frac{\eta}{N}\mathbf{x}\tau^A\Delta_B}{|\mathbf{w}^B + \frac{\eta}{N}\mathbf{x}\tau^A\Delta_B|}, \tag{23}$$

where

$$\Delta_A = \theta(-\tau^A\tau^B)\theta(\frac{P_1}{2} - p_\alpha) - \theta(\tau^A\tau^B)\theta(P_2 - p_\beta)\theta(\frac{1}{2} - p_\gamma)\ ,$$
$$\Delta_B = \theta(-\tau^A\tau^B)\theta(\frac{P_1}{2} - p_\alpha) - \theta(\tau^A\tau^B)\theta(P_2 - p_\beta)\theta(p_\gamma - \frac{1}{2})$$

The introduction of the auxiliary random variables is done according to the following logic: in one half of the cases of disagreement between the units and disagreement between the TPMs, no update occurs in the units (since their sign does not match the TPM's sign) and hence P_1 is divided by 2 in the equation above. The second scenario where updating occurs is when the units have the same sign, the TPMs disagree and therefore one of the units is updated and the other is not. The auxiliary random number p_γ is the one that determines (randomly) which unit of the two is updated.

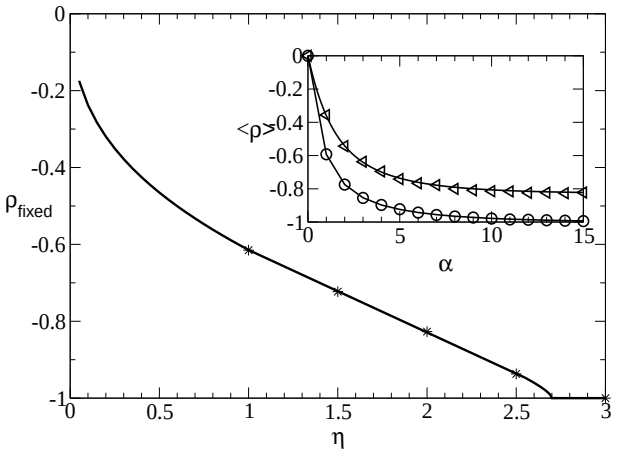

Figure 3: The fixed point ρ_f as a function of η for the continuous TPM as obtained from the solution of Eq. (25) (solid line). Simulation results in some instances of η are presented by stars. Inset: Analytical (solid lines) and simulation results in the case of $\eta = 2$ (triangles) and $\eta = 3$ (circles) for $\langle\rho\rangle$ as a function of α. All simulations are carried out with $N = 5000$ and averaged over 20 samples.

In order to calculate the equations of motion, one has to multiply the updated vectors, Eq. (23), first, and then to perform the two averages; average over the joint probability distributions of the local fields and over the random parameters, p_α, p_β and p_γ. The result of these two averages is an equation over the normalized overlap ρ, that depends only on ρ or equivalently on the angle, θ, (see Eq. (14))

$$\frac{d\rho}{d\alpha} = \eta[\frac{\theta^2}{\pi^2} + (1 - \frac{\theta}{\pi})^2][\frac{1}{\sqrt{2\pi}}(1-\rho) - \frac{\eta\theta}{2\pi}](1+\rho) \tag{24}$$
$$-\frac{2\eta}{\sqrt{2\pi}}(1-\rho^2)\frac{\theta}{\pi}(1-\frac{\theta}{\pi}) - \eta^2\rho\frac{\theta}{\pi}(1-\frac{\theta}{\pi})^2,$$

where α is the number of examples per input dimension. The points $\rho = \pm 1$ are fixed points of the equation of motion above. Both are repulsive when the learning rate, η, is small. As soon as $\eta > \eta_c \sim 2.68$ a phase transition occurs, the $\rho = -1$ fixed point becomes an attractive one and a new phase arises, where the two machines are fully synchronized. The asymptotic decay of ρ to synchronization scales exponentially with α, as can be found by expanding the terms in Eq. (24) around $\theta = \pi$. Apart from the fixed points discussed above, for any η smaller than η_c there is a different attractive fixed point, as can be found by solving numerically Eq. (24). The fixed point $\theta_f(\rho_f)$ is the exact angle(overlap) in a specific learning rate, η, in which the right hand side of equation 24 becomes zero:

$$\eta = \frac{\frac{\sqrt{2\pi}}{\theta_f}\sin^2\theta_f(1-\frac{2\theta_f}{\pi})^2}{(1+\cos\theta_f)(\frac{\theta_f^2}{\pi^2}+(1-\frac{\theta_f}{\pi})^2)+2\cos\theta_f(1-\frac{\theta_f}{\pi})^2}. \quad (25)$$

In Figure 3 we plotted the fixed points as a function of η, as was found numerically from Eq. (25). Simulation results for spherical TPMs with $N = 5000$ and averaged over 20 samples are in agreement with the analysis as indicated by the few tested cases presented by the symbols. Clearly, the system undergoes a phase transition from a partial to a perfect anti-parallel state at $\eta_c \sim 2.68$. One instance for each of the phases is given in the inset of Figure 3. The development of the averaged $\langle\rho\rangle$, averaged over the three hidden units and 20 samples, in the case of partial mutual learning, $\eta = 2$ (triangles), and the case of anti-parallel synchronization, $\eta = 3$ (circles), as a function of α is presented in the inset of Figure 3. Numerical calculations of the analytical equation, Eq. (24), are presented by the solid lines.

B. Dynamic Learning

In the last section we show a procedure that leads to full synchronization. In the following we check the ability of a third TPM, an attacker, to learn the weight vectors of the two parties. The third machine, C, that tries to imitate A, updates its weight vector only when the two parties are updated and only the hidden units that match the output of party A. Mathematically, this is defined as follows

$$\mathbf{w}_i^{C+} = \frac{\mathbf{w}_i^C + \frac{\eta}{N}\mathbf{x}_i\theta(-\sigma^A\sigma^B)\theta(\sigma^A\tau_i^C)\sigma^B}{\left\|\mathbf{w}_i^C + \frac{\eta}{N}\mathbf{x}_i\theta(-\sigma^A\sigma^B)\theta(\sigma^A\tau_i^C)\sigma^B\right\|} \quad . \quad (26)$$

Continuing the same line of introducing probabilities in the mutual learning procedure, one can write a set of updating rules for the dynamic and mutual learning in perceptrons which is equivalent to TPMs learning. This is given by

$$\mathbf{w}^{A+} = \frac{\mathbf{w}^A + \frac{\eta}{N}\mathbf{x}\tau^B\tilde{\Delta}_A}{\left\|\mathbf{w}^A + \frac{\eta}{N}\mathbf{x}\tau^B\tilde{\Delta}_A\right\|}, \quad (27)$$

$$\mathbf{w}^{B+} = \frac{\mathbf{w}^B + \frac{\eta}{N}\mathbf{x}\tau^A\tilde{\Delta}_B}{\left\|\mathbf{w}^B + \frac{\eta}{N}\mathbf{x}\tau^A\tilde{\Delta}_B\right\|} \quad ,$$

$$\mathbf{w}^{C+} = \frac{\mathbf{w}^C + \frac{\eta}{N}\mathbf{x}\tau^B\Delta_C}{\left\|\mathbf{w}^C + \frac{\eta}{N}\mathbf{x}\tau^B\Delta_C\right\|},$$

where

$$\begin{aligned}
\tilde{\Delta}_A &= \theta(-\tau^A\tau^B)\theta(P_1-p_\alpha)\theta(\frac{1}{2}-p_\delta) \\
&\quad +\theta(\tau^A\tau^B)\theta(P_2-p_\beta)\theta(\frac{1}{2}-p_\gamma), \\
\tilde{\Delta}_B &= \theta(-\tau^A\tau^B)\theta(P_1-p_\alpha)\theta(\frac{1}{2}-p_\delta) \\
&\quad +\theta(\tau^A\tau^B)\theta(P_2-p_\beta)\theta(p_\gamma-\frac{1}{2}), \\
\Delta_C &= \theta(-\tau^A\tau^B)\theta(\tau^A\tau^C)\theta(P_1-p_\alpha)\theta(\frac{1}{2}-p_\delta) \\
&\quad +\theta(\tau^A\tau^B)\theta(\tau^A\tau^C)\theta(P_2-p_\beta)\theta(\frac{1}{2}-p_\gamma) \\
&\quad -\theta(-\tau^A\tau^B)\theta(-\tau^A\tau^C)\theta(P_1-p_\alpha)\theta(p_\delta-\frac{1}{2}) \\
&\quad +\theta(\tau^A\tau^B)\theta(-\tau^A\tau^C)\theta(P_2-p_\beta)\theta(p_\gamma-\frac{1}{2}).
\end{aligned}$$

We introduce another random parameter, p_δ, which is redundant when one calculates only the mutual learning, Eq. (23), and it is necessary for deriving equations of motion for the order parameters in the case of dynamic learning. The four terms in Δ_C represent the four possibilities that cause an updating in the attacker hidden unit. For instance, the first term of Δ_C represents the case where the hidden unit in the attacker and in the first TPM have the same state, the TPMs' outputs are different (indicated by P_1) and the outputs in the hidden units of A and B are the same as their TPMs, (the probability for such an event is $\frac{1}{2}$).

The equation of motion after synchronization, i.e., when $\rho_{A,B} = -1$, $\rho_{A,C} = -\rho_{B,C}$, is derived by averaging Eqs. (27) over the joint probability distributions that is given in Eq. (16). It depends on the learning rate and the overlap $\rho_{A,C}$ and is given explicitly by

$$\frac{d\rho_{A,C}}{d\alpha} = \frac{\eta^2}{2}\left(1 - \frac{1}{\pi}\cos^{-1}\rho_{A,C} - \rho_{A,C}\right). \quad (28)$$

This equation describes the development of the overlap between the attacker and one of the two machines that are synchronized in both cases, when each machine learns the opposite of its result, Eq. (26).

As can be derived from Eq. (28), independent of the learning rate, η, there is a unique fixed point $\rho_f \sim 0.79$. The point $\rho = 1$ is not a fixed point at all. Note that this fixed point describes only the failure of the continuous attacker, the equivalent *discrete* attacker might synchronize and gain $\rho = 1$, as discussed in section V C. In Figure 4 we present analytical (solid lines) and simulation results (symbols) for the overlap between that attacker and player A, ρ_{AC}. We carried out simulations with $N = 5000$, and each result averaged 20 times. A good agreement between simulation results and analytical results is presented in Figure 4 in both cases; when the overlap is initialized zero, $\rho_{AC} = 0$ and in the inset, when the initial value of the overlap is almost 1, $\rho_{AC} = 0.98$. All results are for full synchronization between A and B, $\rho_{AB} = -1$.

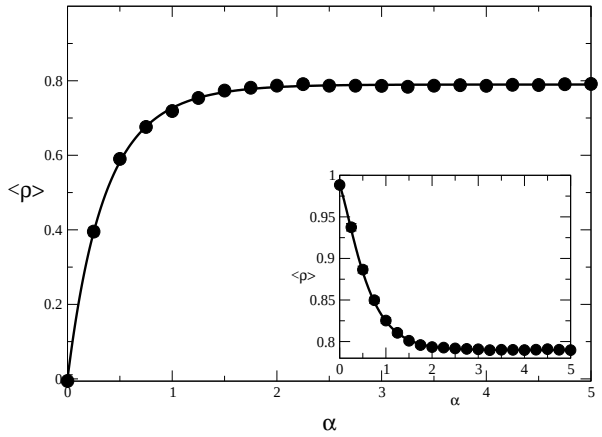

Figure 4: The analytical curve of the averaged overlap, $\langle\rho\rangle$, in a dynamic learning of TPMs as obtained from Eqs. (28) (solid line), with $\eta = 10$. The initial state is $\rho = 0$. Inset: Analytical results for the dynamic learning with the initial state $\rho = 0.98$. Symbols represent the corresponding simulations, carried out with $N = 5000$ and averaged over 20 runs.

C. Summary

In summary, we showed that an initiated pair of random TPMs that perform mutual learning results in a full synchronization state for $\eta > \eta_c$. We introduce here a specific dynamic where the parties update only in a mismatch between the outputs, the updating is in opposite directions of each other and they are normalized in each step (case A in II B). Analyzing case B, for instance, reveals that for all η, the stationary solution is a synchronized state. Using the dynamics appearing in II B but without normalizing the weight vectors does not end in a synchronization state at all. The specific algorithm we chose contains the reach phenomenon of phase transition [23]. Moreover, its synchronization abilities are closely related to the discrete synchronization studied in the following section.

The attacker tries to learn the parties' weight vectors but manages to achieve only partial success. This difficulty in learning that such a naive attacker faces as indicated by the fixed point that differs from 1, also characterizes the naive attacker in the other cases presented in II B. However, the analysis is not relevant for the discrete case studied below. In the discrete case the naive attacker performance is restricted too but perfect learning is possible, see V C. The flipping attacker that makes use of the local fields (see V D) has a better performance in the discrete case. An open question which deserves further research, is how to analyze the continuous flipping attacker.

V. DISCRETE MACHINES

The study of discrete networks requires different methods of analysis than those used for the continuous case. We found that instead of examining the evolution of R and Q, we must examine $(2L+1)\times(2L+1)$ parameters, which describe the mutual learning process. By writing a Markovian process that describes the development of these parameters, one gains an insight into the learning procedure. Thus we define a $(2L+1)\times(2L+1)$ matrix, $\mathbf{F}^{\mu}$, in which the state of the machines in the time step μ is represented. The elements of $\mathbf{F}$, are f_{qr}, where $q, r = -L, ...-1, 0, 1, ...L$. The element f_{qr} represents the fraction of components in a weight vector in which the A's components are equal to q and the matching components in d unit B are equal to r. Hence, the overlap between the two units as well as their norms are defined through this matrix,

$$R = \sum_{q,r=-L}^{L} qr f_{qr}, \qquad (29)$$

$$Q^A = \sum_{q=-L}^{L} q^2 f_{qr} \quad Q^B = \sum_{r=-L}^{L} r^2 f_{qr}.$$

The matrix elements are updated, if and only if, an update of the weight vectors occurs.

A. Learning with Discrete Perceptrons

The mutual learning scenario is much simplified in the case of the perceptron, therefore we present here the full description of the analytical procedure used for this case. Updating is done in the case of a mismatch, and the aim is to arrive at a state in which the weight vectors are anti-parallel, $\rho = -1$ (we could aim at $\rho = 1$ instead, see the manifold of possible dynamics in II A, and the results would be equivalent). The dependence of the weight vector in a new step on the weight vector in the former one is given by:

$$\mathbf{w}_i^{A+} = \mathbf{w}_i^A + D(\mathbf{w}_i^A \mathbf{x}_i \sigma^B)\mathbf{x}_i \sigma^B \theta(-\sigma^A \sigma^B), \qquad (30)$$
$$\mathbf{w}_i^{B+} = \mathbf{w}_i^B + D(\mathbf{w}_i^B \mathbf{x}_i \sigma^A)\mathbf{x}_i \sigma^A \theta(-\sigma^A \sigma^B),$$

where $\sigma^{A/B}$ represents the output of TPM A/B, and $\mathbf{w}^{A/B}$ represents its weight vector.

The update of the elements of the matrix $\mathbf{F}$, is calculated directly from Eq. (30), where one must average over the input components X_{ij}. On the average, half of the updated weights in one machine are increased by 1, while the matching weights in the other machine are decreased by 1 and vice versa.

The possibility for agreement/disagreement between the parties is a function of the current overlap between them, calculated using the matrices (see Eq. (29)). This probability is implemented by choosing a random parameter, p_α between $[0, 1]$. If it is smaller than ϵ, as defined in

Eq. (20), the parties disagree, otherwise they agree. The updating of matrix elements is described as follows: for the elements with q and r which are not on the boundary, ($q \neq \pm L$ and $r \neq \pm L$) the update can be written in a simple manner,

$$f^+_{q,r} = \theta\left(p_\alpha - \epsilon\right) f_{q,r} + \theta\left(\epsilon - p_\alpha\right)\left(\frac{1}{2}f_{q+1,r-1} + \frac{1}{2}f_{q-1,r+1}\right). \tag{31}$$

For elements with both indices on the boundary, the update is

$$\begin{aligned} f^+_{L,L} &= \theta\left(p_\alpha - \epsilon\right) f_{L,L}, \qquad (32)\\ f^+_{-L,-L} &= \theta\left(p_\alpha - \epsilon\right) f_{-L,-L},\\ f^+_{L,-L} &= \theta\left(p_\alpha - \epsilon\right)\left(\frac{1}{2}f_{L,-L}\right) + \theta\left(\varepsilon - p_\alpha\right) \times \\ &\quad \left(\frac{1}{2}f_{L-1,-L+1} + \frac{1}{2}f_{L-1,-L} + \frac{1}{2}f_{L,-L+1}\right),\\ f^+_{-L,L} &= \theta\left(p_\alpha - \epsilon\right) f_{-L,L} + \theta\left(\epsilon - p_\alpha\right) \times \\ &\quad \left(\frac{1}{2}f_{-L+1,L-1} + \frac{1}{2}f_{-L+1,L} + \frac{1}{2}f_{-L,L-1}\right). \end{aligned}$$

For elements with just one of the indices on the boundary ($q = \pm L$ and $r \neq \pm L$ or vice versa), the update is

$$\begin{aligned} f^+_{q,L} &= \theta\left(p_\alpha - \epsilon\right) f_{q,L} + \qquad (33)\\ &\theta\left(\epsilon - p_\alpha\right)\left(\frac{1}{2}f_{q+1,L-1} + \frac{1}{2}f_{q+1,L}\right),\\ f^+_{q,-L} &= \theta\left(p_\alpha - \epsilon\right) f_{q,-L} + \\ &\theta\left(\epsilon - p_\alpha\right)\left(\frac{1}{2}f_{q-1,-L+1} + \frac{1}{2}f_{q-1,-L}\right),\\ f^+_{L,r} &= \theta\left(p_\alpha - \epsilon\right) f_{L,r} + \\ &\theta\left(\epsilon - p_\alpha\right)\left(\frac{1}{2}f_{L-1,r+1} + \frac{1}{2}f_{L,r+1}\right),\\ f^+_{-L,r} &= \theta\left(p_\alpha - \epsilon\right) f_{-L,r} + \\ &\theta\left(\epsilon - p_\alpha\right)\left(\frac{1}{2}f_{-L+1,r-1} + \frac{1}{2}f_{-L,r-1}\right), \end{aligned}$$

The main quantity of interest is the number of steps required in order to arrive at a state of full synchronization. In simulations there is a discrete transition from an overlap which is almost anti-parallel to a completely anti-parallel state. This is due to the finite nature of the vectors, the largest value of overlap before synchronization is $-1 + O(1/N)$. In simulations with $N = 10^4$, for example, the largest value of the overlap before full synchronization is $\rho = 0.99999$, and this is the value we used in our analytical procedure, for defining full synchronization for comparison to simulations with $N = 10^4$.

Our results indicate that the order parameters are not self-averaged quantities [19]. Several runs with the same N, results in different curves for the order parameters as a function of the number of steps, see Figure 5. This

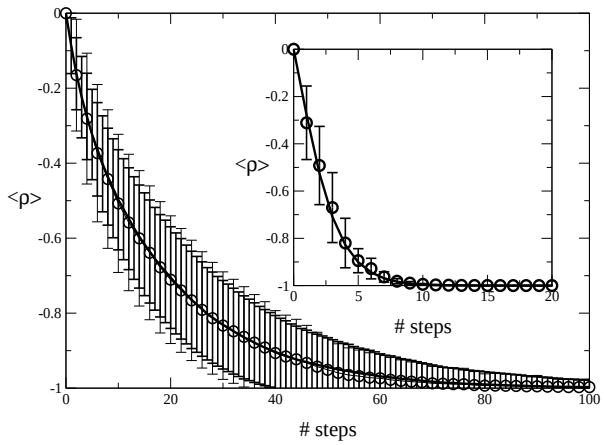

Figure 5: The averaged overlap $\langle\rho\rangle$ and its standard deviation as a function of the number of steps as found from the analytical results (solid line) and simulation results (circles) of mutual learning in TPMs. Inset: analytical results (solid line) and simulation results (circles) results for the perceptron, with $L = 1$ and $N = 10^4$.

explains the non-zero variance of ρ as a results of the fluctuations in the local fields induced by the input even in the thermodynamic limit.

In the inset of Figure 5 we present the averaged numerical results derived from the analytical equations, (31, 32, 33) of synchronization in the perceptron (solid line) with $L = 1$, $W_i = \pm 1, 0,$. The analytical results are averaged over 500 samples and the non-zero standard deviations are not presented in order to simply the presentation. Simulation results with $L = 1$ ($W_i = \pm 1, 0$) and $N = 10^4$, averaged over 500 samples are presented by the circles; error bars are standard deviations. Note that even though the matrix elements were initiated with the same values in each run, there is still a non-zero standard deviation due to fluctuations in the local fields as a function of the particular set of random inputs even in the thermodynamic limit.

For the perceptron, synchronization is much easier and faster to achieve than for the TPM. Take for example the case where $L = 1$. If for three consecutive steps, both the other party's output and x_i were positive, an attacker can surely know that $W_i = 1$, while this is not so in the TPM case, as the attacker cannot know for sure whether the unit was updated or not. Therefore, the TPM is much more suitable for building a cryptosystem than the perceptron.

B. Synchronization in TPMs

Mutual learning in discrete TPMs is described by mutual learning discrete noisy perceptrons. As the TPM

consists of three hidden units (each evolving differently), we now have three different angles, θ_i where $i = 1, 2, 3$, for each hidden unit. Since the dynamics are not self-averaged, we use probabilities similar to those introduced in Eq. (21). The definitions of these probabilities are extended to include all three hidden units, and each one is characterized by its own angle, P_1^i, P_2^i. The probability of $P_1(\sigma^A \neq \sigma^B | \tau_i^A \neq \tau_i^B)$, is given by

$$P_1^i = \epsilon_j \epsilon_k + (1-\epsilon_j)(1-\epsilon_k). \tag{34}$$

Similarly, the probability that there is a mismatch between the two TPMs given that there is agreement between the ith pair of hidden units, for instance, is given by

$$P_2^i = \epsilon_j (1-\epsilon_k) + \epsilon_k (1-\epsilon_j). \tag{35}$$

Here, as well as in the continuous case, we chose a sequence of random parameters to represent the particular choice of random inputs.

We follow each hidden unit separately and therefore we have three matrices, $\mathbf{F}^i$. We initialize the weights randomly, therefore the matrices in the initial state have the values of $1/(2L+1)^2$ in each entry. In each step, two sets of random parameters are chosen and are used to set a specific realization of the internal presentation for the parties. The first set is used to define agreement or disagreement between each pair of hidden units, as done in the perceptron case V A.

All in all, due to inversion symmetry, when $K = 3$ there are four possible results for the internal presentations, $+++$, $+--$, $-+-$ or $--+$ and accordingly 4×4 possible states, for which the parties' output does not match, and an update is performed. We then use the second set of random parameters for defining the specific internal presentation in one of the TPMs, and therefore immediately in the other, according to their agreement/disagreement.

The case when the three hidden units disagree is exemplified below. There is a possibility that all hidden units are updated, (case (b) in II A), or only one of them; (case (b) describes two of the hidden units and case (d) describes the third). In two of the eight such internal presentations all the three hidden units are updated whereas in the other six, only one of them is updated, so that we must choose which one. All of these possibilities are equally probable, independent of θ_i. Therefore, we take all the possible internal scenarios into account and, for instance, when after using the auxiliary random numbers, all three hidden units disagree, we choose at random p_α and accordingly update,

$$\begin{aligned} f_{q,r}^{i+} &= \theta(\frac{1}{4} - p_\alpha)(\frac{1}{2} f_{q+1,r-1}^i + \frac{1}{2} f_{q-1,r+1}^i) \\ &+ \theta(\frac{i+1}{4} - p_\alpha)\theta(p_\alpha - \frac{i}{4})(\frac{1}{2} f_{q+1,r-1}^i + \frac{1}{2} f_{q-1,r+1}^i). \end{aligned} \tag{36}$$

The first term corresponds to the case where all three hidden units are updated (with probability $\frac{1}{4}$). The second term corresponds to the case where only one hidden unit is updated. Eq. (36) is valid only for q and r which are not on the boundary.

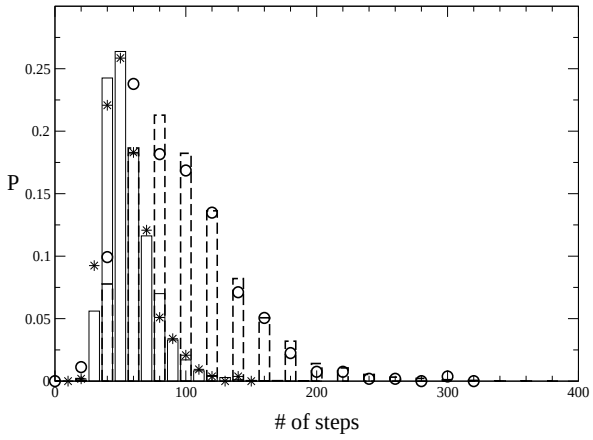

Figure 6: The synchronization time (dashed line) and the dynamic learning time (solid line) distribution, of analytical results for TPMs, with $L = 1$. Symbols stand for the simulations results, with $N = 10000$.

In the case of the perceptron when an update occurs, both sides perform the update, in opposite directions. In the case of the TPMs, two matching units do not always perform an update together; in many cases one of the parties updates unit i, while the other updates unit j, $i \neq j$, as described in case (c) in II A. In such a case, Eq. (36) is not sufficient, and we should add a description of the matrices' update when only one party is updated. Let us say the party represented by the matrix rows is updated. Then we have

$$f_{q,r}^{i+} = \frac{1}{2} f_{q+1,r}^i + \frac{1}{2} f_{q-1,r}^i, \tag{37}$$

and if the party represented by the matrix columns is updated, we have

$$f_{q,r}^{i+} = \frac{1}{2} f_{q,r+1}^i + \frac{1}{2} f_{q,r-1}^i, \tag{38}$$

where we limit the description only to q, r which are not on the boundary. An example is the case when the internal presentation of party A is $-++$ and that of B is $--+$. Then party A updates unit 1, Eq. (37) with $i = 1$, while party B updates unit 3, Eq (38) with $i = 3$.

In Figure 6 we present the distribution of time steps for synchronization according to simulations with $N = 10,000$, ($\star$), and according to the analytical results (solid line) in the case of $L = 1$, taken from 500 different runs. The evolution of the average overlap in this case is given in Figure 5. A solid line represents the analytical results and circles stand for simulation results. Both standard deviations are indicated by the error bars. There is good agreement between the analytical and simulation results.

	t_{synch}	t_{naive}	t_{flipp}
$L=1$	25 ± 14	36 ± 18	32 ± 19
$L=2$	79 ± 38	239 ± 145	108 ± 58
$L=3$	166 ± 67	3320 ± 3039	221 ± 106
$L=4$	298 ± 113	$176810 \pm 179,446$	380 ± 159

Table I: Average synchronization and dynamic learning times, for the naive attacker and the flipping attacker, for different values of L.

An attacker does not have to achieve full synchronization in order to decipher the secret code. For finite N, even a state close enough to synchronization is sufficient to break the code, thus making the system insecure. Moreover, the analysis and the simulations are faster when the aim is to arrive at a partial overlap state. We therefore considered an attacker who achieves $\langle\rho\rangle = 0.9$, a successful attacker, and synchronization and learning times given in Figure 7 and in Table I are for achieving $\langle\rho\rangle = 0.9$.

C. The Naive Attacker

The aim of an attacker is to synchronize with one of the parties and reveal the secret key (the weights of the parties), hence her natural strategy is to imitate one of them, party A for instance, by using the same learning rule. The attacker, eavesdropping on the public channel connecting the parties, knows the input vector $\mathbf{x}_i$ and the output $O^{A/B}$. When $O^A \neq O^B$, the parties update their weights, and so does the attacker. In the case where the attacker's internal presentation is the same as A's, they update the same units, an attractive step occurs, and the attacker gets closer to her goal. Yet when the internal presentations of the attacker and the party differ, she updates some wrong units, a repulsive step occurs, and this delays her. The 2^{K-1}-fold degeneracy in the output is the main reason for the attacker's failure. The dependence of the attacker's weight vector in a new step on the weight vector in the former one is given by

$$\mathbf{w}_i^{C+} = \mathbf{w}_i^C + D(\mathbf{w}_i^C \mathbf{x}_i \sigma^B)\mathbf{x}_i \sigma^B \theta(-\sigma^A \sigma^B). \qquad (39)$$

The analysis is similar to the synchronization process, given by Eq. (36). We now create 9 matrices, each representing the state of two matching hidden units among two parties, and the attacker and each party. We must set the parties' internal presentation, as well as the attacker's. We decide which one of the $8 \times 8 \times 8$ internal presentations occurs in each step, following the correlation between the parties and the attacker, and update the matrices accordingly, as described in V B.

Although the attacker may synchronize before the parties, the average learning time is around twice the synchronization time for $L = 1$, and is around 200 times the synchronization time for $L = 3$. It seems that the reason for the naive attacker's weakness is that too many repulsive steps occur; therefore, when trying to improve her abilities, we need to increase the probability for an attractive step, and decrease the probability for a repulsive one. It has been shown [24] that a small absolute local-field value indicates a high probability for an error. In the next section we present an advanced attacker which makes use of this knowledge.

D. The Flipping Attacker

The flipping attacker's strategy, recently introduced in [9], adds a different move to the strategy of the naive attacker when disagreement occurs between the outputs of the attacker and party A. In this case, the attacker is certain that either one or three of her hidden units are in disagreement with A's units, and therefore a repulsive step will occur. Since disagreement of three units is less likely than disagreement of one unit, the attacker's strategy treats all cases as a one unit disagreement. The flipping attacker tries to prevent the repulsive step by using a "flipping" approach; she negates the sign of one of her units, before performing the update. If the correct unit was chosen, then the "new" internal presentation matches that of the party, and the same units will be updated by both, thus performing an attractive step. To raise her chances of flipping the right unit, the attacker chooses the one whose absolute local-field value is the lowest of the three : $\hat{\tau}_i = -\tau_i$ for i that minimizes $|h_i|$.

The learning rules are the same as those given by Eq. (12) for the mutual synchronization, but the attacker's learning is different,

$$\begin{aligned} \mathbf{w}_i^{C+} &= \mathbf{w}_i^C + D(\mathbf{w}_i^C \mathbf{x}_i \sigma^B)\mathbf{x}_i \sigma^B \theta(-\sigma^A \sigma^B) \times \\ &\quad [\theta(\sigma^C \sigma^A)\theta(\sigma^C \tau_i^C) + \theta(-\sigma^C \sigma^A)\theta(\sigma^A \hat{\tau}_i^C)] \end{aligned} \qquad (40)$$

where $\hat{\tau}_i = -\tau_i$ if $|h_i| < |h_j|, \forall j \neq i$ and $\hat{\tau}_i = \tau_i$ otherwise.

The analysis used here is the same as for the naive attacker. Here too, we follow the development of 9 matrices which are updated at every time step, as described in V B. However, in cases where the attacker's output disagrees with the A's output, we compute the probability for every unit to be the one with the lowest absolute local field value. For instance, when $h_i^C > 0, \forall i$, the probability for h_1 being the smallest is given by:

$$\begin{aligned} &P(h_1^C < h_2^C, h_1^C < h_3^C) = \\ &\int_0^\infty P(h_1^C | h_1^A, h_1^B, \{\rho, Q\}) dh_1^C \\ &\int_{h_1^C}^\infty P(h_2^C | h_2^A, h_2^B, \{\rho, Q\}) dh_2^C \\ &\int_{h_1^C}^\infty P(h_3^C | h_3^A, h_3^B, \{\rho, Q\}) dh_3^C \end{aligned} \qquad (41)$$

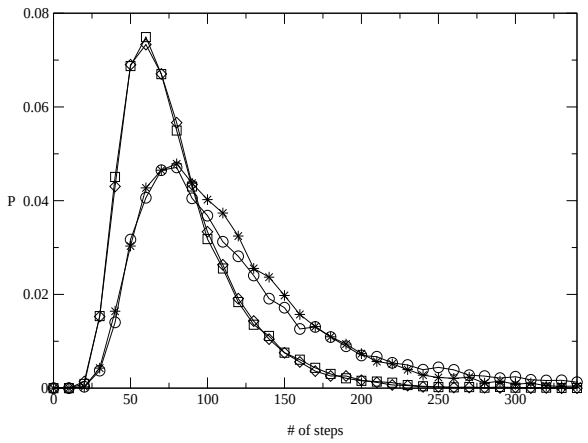

Figure 7: The synchronization time and learning time distribution for the flipping attacker, obtained by simulations with $N = 10^3$ (diamonds/stars for synchronization/learning) and analytical calculations (squares/circles for synchronization/learning) with $L = 3$, averaged over 10^4 runs.

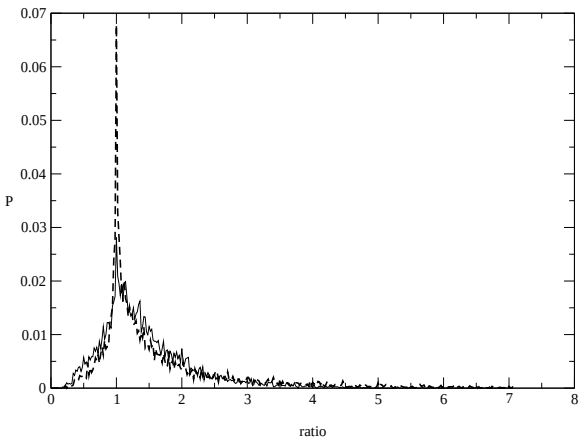

Figure 8: The distribution of the ratio $R = t_{learn}/t_{synch}$, obtained by simulations (dashed line) with $N = 10^3$, and analytical (solid line) results, with $L = 3$, averaged over 10^4 runs. .

where the conditional probabilities are given by Eq. (19).

The generalization to other cases in which h_i^C is not necessarily positive, is straightforward. We choose at random two specific local fields for the two parties h_i^A and h_i^B, from their joint probability distribution which is derived from the correlation matrix, making use of the overlap between the parties' units. We then proceed to calculate the probability of each unit of the attacker to be the one with the lowest absolute local field value, given by Eq. (41). Once we have P_i, $i = 1, 2, 3$ (P_i is the probability that unit i has the lowest local field value), we use an auxiliary random number p_α, to choose the unit to be flipped:

$$\hat{\tau}_i = \tau_i \left[1 - 2\theta \left(p_\alpha - \sum_{j=1}^{i-1} P_j \right) \theta \left(\sum_{j=1}^{i} P_j - p_\alpha \right) \right] \quad (42)$$

where $P_0 = 0$.

Simulations and analytical calculations with $L = 3$, $N = 10^3$ averaged over 10^4 runs, indicate that the flipping attacker is successful. In figure 7 we plotted the synchronization time and learning time distribution for the flipping attack, obtained by simulations (circles for synchronization and squares for learning) and analytical calculations (squares for synchronization and triangles for learning). The flipping attacker's ability can be measured by the ratio of the attacker learning time and the parties' synchronization time, $R = t_{learn}/t_{synch}$. Figure 8 shows the distribution of this ratio for simulations (dashed line) and analytical (solid line) results. The probability of the flipping attacker to finish learning before synchronization is achieved by the parties is 28%, as presented in Figure 8.

E. Discussion

In the previous section we introduced macro-dynamical updating equations that imitate the simulation results of discrete mutual and dynamic learning. All numeric runs of the macro-dynamical equations are in good agreement with simulations. The TPMs that perform mutual learning synchronize in a finite number of steps that scales with $\ln N$. The macro-dynamical updating equations describe the system in the limit of $N \to \infty$, and they result in an exponential decay of the order parameter ρ to -1, where receiving the exact value of -1 depends on computer accuracy. However, defining the synchronization by any finite and close to -1 value, results in a synchronization state that is achieved in a finite number of steps even in the thermodynamic limit. The good fit in that limit between analytical results and simulations results is indicated in Figures 6, 7 and 8. We presented here analytical results in the case of continuous as well as discrete weight vectors. Recently, [11] the scaling between N and L was discussed, based on large scale simulations with different L and N values. It may be interesting to develop the numerical equations in the limit of infinite L and to find the appropriate interplay between these two quantities.

We conclude by presenting the potential of the TPMs to serve as a public key cryptosystem. This is based upon the following features: the synchronization state may serve as the key in a certain encryption and decryption rule. This key evolves in public without the need of prior communication; one needs only to perform a finite number of steps of exchanging inputs and outputs in order to converge to a synchronized state. The analytical derivation shows that even for infinite large sys-

tems, $N \to \infty$, there will be finite distribution of synchronization times (where synchronization time is defined by $\rho = -1+\epsilon$ where small ϵ is a coefficient) and the synchronization time itself will be finite. The flipping attacker succeeds in revealing the secret for small L values, as L enlarges the task becomes harder for her [11]. It is yet to be determined whether it is possible to make better use of the information in the channel, and to device a strategy that performs perfect learning on the average in the same number of steps typical for synchronization even for large L.

Acknowledgments

I.K. acknowledges partial support of the Israel Academy of Science. This paper is part of the Ph.D. Thesis of M.R.

[1] J. Hertz, A. Krogh and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Redwood City, CA: Addison-Wesley, 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) and W. Kinzel, R. Metzler and I. Kanter, J. Phys. A. **33** L141 (2000).
[4] W. Kinzel, Contribution to Networks, ed. by H.G. Schuster and S. Bornholdt, to be published by Wiley VCH (2002).
[5] E. Barkai, D. Hansel and I. Kanter, Phys. Rev. Lett. **65**, 2312 (1990).
[6] M. Opper, Phys. Rev. E., **51**, 3613 (1995).
[7] R. Simonetti and N.J. Caticha, J. Phys. A **29**, 4859 (1996).
[8] I. Kanter, W. Kinzel and E. Kanter, Europhys, Lett., **57**, 141 (2002).
[9] A. Shamir, A. Mityagin and A. Kilmov, Ramp Session, Eurocrypt Amsterdam 2002.
[10] M. Rosen-Zvi, I. Kanter and W. Kinzel, cond-mat/0202350
[11] R. Mislovaty, Y. Perchenok, I. Kanter and W. Kinzel, condmat/0206213
[12] D. R. Stinson, *Cryptography: Theory and Practice* (CRC Press 1995).
[13] C. Van den Broeck and M. Bouten, Europhys. Lett. **22**, 223 (1993).
[14] J. Schietse, M. Bouten and C. Van den Broeck, Europhys. Lett. **32**, 279 (1995).
[15] W. Kinzel and R. Urbanczik, J. Phys. A **31**, L27-30 (1998).
[16] M. Rosen-Zvi, J. Phys. A. **33**, 7277 (2000).
[17] M. Rosen-Zvi and I. Kanter, Phys. Rev. E, **64**, 046109 (2001).
[18] L. P. Kadanoff, *Statistical Physics: Statistics, Dynamics and Renormalization* (World Scientific Publishing, Singapore 2000).
[19] G. Reents and R. Urbanczik, Phys. Rev. Lett., **80**, 5445 (1998).
[20] E. Barkai, D. Hansel and H. Sompolinski, Phys. Rev. A., **45**, 4146 (1992).
[21] A. Engel, H. M. Kohler, F.Tschepke, H. Volllmayr and A. Zippleius, Phys. Rev. A., **45**, 7590 (1992).
[22] M. Copelli, O. Kinouchi and N. Caticha, Phys. Rev. E, **53**, 6341 (1996).
[23] W. Kinzel, Phil. Mag. B **77**, 1455 (1998).
[24] L. Ein-Dor and I. Kanter, Phys. Rev. E, **60**, 799 (1999).