Available online at www.sciencedirect.com
SCIENCE DIRECT
Neural Networks 18 (2005) 1051–1063
Neural Networks
www.elsevier.com/locate/neunet

2005 Special issue

Recursive principal components analysis

Thomas Voegtlin*

INRIA-Campus Scientifique, B.P. 239 F-54506 Vandoeuvre-Les-Nancy Cedex, France

Abstract

A recurrent linear network can be trained with Oja's constrained Hebbian learning rule. As a result, the network learns to represent the temporal context associated to its input sequence. The operation performed by the network is a generalization of Principal Components Analysis (PCA) to time-series, called Recursive PCA. The representations learned by the network are adapted to the temporal statistics of the input. Moreover, sequences stored in the network may be retrieved explicitly, in the reverse order of presentation, thus providing a straightforward neural implementation of a logical stack.

Keywords: Unsupervised learning; Recursive PCA; Time series; Recurrent neural network; Logical stack

1. Introduction

Among the most interesting neural representations of time are recurrent networks, where neural activities in the recurrent loops reflect the past values of a time-varying input. These representations are biologically plausible, robust to noise and can handle temporal deformations of the signal. A very influential example illustrating this is the Simple Recurrent Network (Elman, 1990).

The main difficulty with recurrent networks has been to find adequate and efficient training algorithms. Various supervised and unsupervised methods have been proposed. In the supervised case, a network has to predict the next value of a time series given its predecessors (Weigend, Huberman, & Rumelhart, 1990), and error gradient descent is classically used to train the network (Hochreiter & Schmidhuber, 1997; Williams & Zipser, 1989). In general, these methods encounter difficulties to learn long-term dependencies (Bengio, Simard, & Frasconi, 1994). In most unsupervised methods, sequences of inputs are stored as attractors of the dynamics of a recurrent network (Hopfield, 1982; Morita, 1996). However, attractor networks suffer from very limited capacity.

One common aspect of these methods is that they optimize the internal states of a recurrent network to a particular task. Recently, it has been argued that this approach cannot account for the general ability of neural circuitry to carry out several real-time computations in parallel (Maass, Natschlger, & Markram, 2002). This, combined with the above mentioned difficulties to train recurrent networks, has motivated so-called 'echo states', or 'liquid states' approaches. In this paradigm, the rich non-linear dynamics of a randomly connected recurrent neural network provides an internal state. A supervised feed-forward classifier network, which is the second part of the machine, uses this internal state as its input (Maass & Markram, 2003). Although these methods do provide interesting results, they remain unsatisfactory from a descriptive point of view. Indeed, the synaptic efficiencies of their recurrent connections are chosen at random; the difficulty of learning a time-dependent task has been shifted from the recurrent network to the feed-forward classifier, for which a better theory is available. At the same time, the ambition of training a recurrent neural network has been given up, which underlines our lack of understanding of these networks.

In this paper, an unsupervised learning algorithm for recurrent networks is proposed. Following (Maass et al., 2002), we do not adapt the network to a particular task, but we rather try to make its dynamics as rich as possible. However, instead of relying on random connections and complex non-linear dynamics, we use a very simple linear network, and we propose a learning rule that enhances the dynamics of the network. This learning rule is a generalization of Oja's Hebbian rule (Oja, 1989), which is

* Tel.: +33 3 83 59 30 72; fax: +33 3 83 27 83 19.
E-mail address: voegtlin@loria.fr

0893-6080/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2005.07.005

known to extract the principal components of a distribution. We demonstrate that this generalized Oja rule results in a network that efficiently stores and retrieves arbitrary sequences of inputs. In addition, stored sequences can be retrieved in reverse order, which provides an implementation of a logical stack.

The paper provides a detailed analysis of the learning model. Although the proposed network is very simple, understanding its learning dynamics does provide useful insights on recurrent networks.

2. A simple recurrent neural network

Consider a simple recurrent neural network, whose linear dynamics are described by the following equation:

$$\mathbf{y}_t = \mathbf{W}_\mathbf{x}\mathbf{x}_t + \sqrt{\alpha}\mathbf{W}_\mathbf{y}\mathbf{y}_{t-1} \tag{1}$$

Here t is a discrete time index, $\mathbf{x}_t$ is a zero-mean input vector, and $\mathbf{y}_t$ is an output vector. We assume that the time series $(\mathbf{x}_t)$ is bounded and stationary. Let n and m denote the dimensions of $\mathbf{x}$ and $\mathbf{y}$, respectively. $\mathbf{W}_\mathbf{x} \in \mathbb{R}^{n\times m}$ and $\mathbf{W}_\mathbf{y} \in \mathbb{R}^{m\times m}$ are the matrices of synaptic efficiencies, which correspond to feed-forward and recurrent connections, respectively. We assume that matrix $\mathbf{W}_\mathbf{y}$ has eigenvalues below or equal to one, and that α is a positive gain, strictly lower than one.[1] Therefore, $\mathbf{y}_t$ is bounded, and asymptotically independent of its initial value.[2] Hence, the distribution of the series $(\mathbf{y}_t)$ does not depend on the initial condition.

Oja (1989) proposed a constrained Hebbian learning rule that trains a feed-forward neural network to extract the principal subspace of a distribution.[3] Here, we generalize this learning rule to our recurrent network. Intuitively, we may consider that Oja's rule will try to maximize the variance of the output $\mathbf{y}_t$, and that $\mathbf{y}_t$ is a compressed representation of both $\mathbf{x}_t$ and $\mathbf{y}_{t-1}$. Maximizing variance can thus be seen as a way to provide a rich internal dynamics. Let us consider vector $\mathbf{z}_t = [\mathbf{x}_t; \sqrt{\alpha}\mathbf{y}_{t-1}]^\mathrm{T}$, where T denotes the transpose. Eq. (1) may be rewritten as:

$$\mathbf{y}_t = \mathbf{W}\mathbf{z}_t \tag{2}$$

where $\mathbf{W} = |\mathbf{W}_\mathbf{x};\ \mathbf{W}_\mathbf{y}|$. Applying Oja's rule to vector $\mathbf{z}_t$ yields the following equations

$$\begin{cases} \Delta w_\mathbf{x}^{ij} = \eta y_t^i \left(x_t^j - \sum_{k=1}^{m} w_\mathbf{x}^{kj} y_t^k \right) \\ \Delta w_\mathbf{y}^{ij} = \eta y_t^i \left(\sqrt{\alpha} y_{t-1}^j - \sum_{k=1}^{m} w_\mathbf{y}^{kj} y_t^k \right) \end{cases} \tag{3}$$

where η is a learning rate, and connections are updated on each time step.

In the remainder of this paper, we will study this learning model both theoretically (in Sections 3 and 4) and experimentally (in Sections 5 and 6). In Section 3, convergence of the learning dynamics is studied. In Section 4, we demonstrate that the network is able to store and retrieve sequences of inputs, and that an objective error function is associated to this capability. In Section 5, the performance of the network on a number of time series is demonstrated. In Section 6, the existence of local minima of the error function is established experimentally.

3. Recursive PCA

In this section, we propose a theoretical study of the operation performed by the network. Oja's rule is known to extract the principal subspace of a distribution. Here, we applied this learning rule to vector $\mathbf{z}$. Hence, we expect our network to perform something related to principal components analysis (PCA), but in the context of a recurrent network. The purpose of this section is to formally define what this 'something related' is.

Before proceeding with the theoretical analysis, we shall first say that numerical convergence of the weights is observed experimentally, for any stationary input time series, and for all $\alpha \in [0;\ 1]$. Convergence should be understood in a broad sense here: the question is whether the expected values of the weights will settle to a stable value, given a fixed and arbitrarily small value of the learning rate. Hence, weights are approximately constant over a small interval of time, and $\mathbf{W}$ refers to their average value over such an interval. In this broad sense, convergence of the weights through Oja's rule does imply that the rows of $\mathbf{W}$ span the principal subspace of vector $\boldsymbol{z}$. For this reason, we base our definition on the PCA of vector $\boldsymbol{z}$. We define Recursive PCA as follows:

Definition. Given an input time series $(\mathbf{x}_t)_{t\in\mathbb{Z}}$, a weight matrix $\mathbf{W}$, and a value of the gain α, the distribution of vector $\mathbf{z}$ is entirely defined by Eq. (2). Let us assume that the lines of $\mathbf{W}$ are orthonormal vectors that form a basis of the m-subspace of highest variance of $\mathbf{z}$. Hence, $\mathbf{W}\mathbf{W}^\mathrm{T}$ is the identity matrix of size m, and $\mathbf{W}^\mathrm{T}\mathbf{W}$ is a projection onto the m-subspace of highest variance of $\mathbf{z}$. We call *Recursive Principal Components Analysis* the operation of finding such a matrix $\mathbf{W}$.

[1] The reason why we consider the gain not to be part of $\mathbf{W}_\mathrm{y}$, and why we use a square root, is convenience, as will become clear later.

[2] The influence of the initial condition decays geometrically, and does not affect the results presented in this paper.

[3] The networks proposed by Sanger and Oja both involve feedback connections. However, the operation of these networks is feed-forward, and feedback is only considered for the propagation of a learning signal.

3.1. Existence of solutions

The above definition does properly describe the operation that is performed by our network; if the weights settle to stable values, then the rows of $\mathbf{W}$ will span the principal subspace of $\mathbf{z}$.

However, this definition is self-referent, because the distribution of $\mathbf{z}$ is not defined a priori. Instead, this distribution does depend on $\mathbf{W}$. Therefore, the existence of a matrix $\mathbf{W}$ satisfying the constraints is not guaranteed, at least from a theoretical point of view. In order to demonstrate the existence of such a matrix, it would be sufficient to demonstrate that weights converge through our generalization of Oja's rule, which is what we observe experimentally. This would demonstrate the existence of solutions *a posteriori*.

Unfortunately, such a proof could not be established. In the following paragraphs, we will present a different proof, which involves hypotheses that are not satisfied by our learning algorithm. In addition, the proof does not hold for the whole interval [0; 1] of values of α. However, this proof still demonstrates the existence of solutions, and it has the merit to shed light on an important distinction, between objective and subjective error functions.

3.2. Objective and subjective errors

One quantity of interest for Recursive PCA is the mean-squared reconstruction error E between vector $\mathbf{z}$ and its projection $\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{z}$:

$$E = \langle ||\mathbf{z} - \mathbf{W}^{T}\mathbf{W}\mathbf{z}||^2 \rangle \tag{4}$$

where $<.>$ denotes the mean over time. This error depends on the weights and on the distribution of vector $\mathbf{z}$. However, due to the recurrent connections, the distribution of $\mathbf{z}$ also depends on the weights. It is, therefore possible, for a fixed input time series, to write E as a function of only the weights: $E = E(\mathbf{W})$. We call *objective error* this function.

Although the objective error obviously has minima, it is wrong to assert that these minima satisfy the constraints of Recursive PCA. As we will see in Section 6, it might happen that a minimum of E is unstable through the learning rule, because it does not satisfy these constraints. In fact, the learning rule updates the weights in a way that maximizes the variance of the output, *assuming a fixed distribution of* $\mathbf{z}$. This means that the error landscape associated to our learning rule is not the objective error landscape $E(\mathbf{W})$, but a different one. To see this, consider a fixed value, $\tilde{\mathbf{W}}$, of the weight matrix. A unique distribution of $\mathbf{z}$, denoted by $\tilde{\mathbf{z}}$, results from using the set of weights $\tilde{\mathbf{W}}$ in Eq. (2). It is possible to consider the mean squared error $E_{\mathrm{subj}}(\mathbf{W}, \tilde{\mathbf{W}})$ that results from using $\tilde{\mathbf{z}}$, and another set of weights, $\mathbf{W}$:

$$E_{\mathrm{subj}}(\mathbf{W}, \tilde{\mathbf{W}}) = \langle ||\tilde{\mathbf{z}} - \mathbf{W}^{\mathrm{T}}\mathbf{W}\tilde{\mathbf{z}}||^2 \rangle \tag{5}$$

We call *subjective error* this error landscape. Subjective error is the error that is taken into account by the learning rule. Note that both objective and subjective error functions coincide for $\mathbf{W} = \tilde{\mathbf{W}}$:

$$E_{\mathrm{subj}}(\mathbf{W}, \mathbf{W}) = E(\mathbf{W}) \tag{6}$$

The distinction between objective and subjective error landscapes is due to the fact that we use a learning rule that was designed for a feed-forward network, in a recurrent architecture. Using this learning rule to update $\mathbf{W}$ results in a moving-target problem, where the minimum of the subjective error surface might change whenever weights are updated.

For any fixed value $\tilde{\mathbf{W}}$ of the weights, the classical result by Baldi and Hornik (1988) holds: up to equivalence, the landscape of the subjective error $E_{\mathrm{subj}}(\mathbf{W}, \tilde{\mathbf{W}})$ has a unique minimum in $\mathbf{W}$, and all its other critical points are saddle points. Let E^{*}_{subj} denote this minimum. The 'moving target' of the learning algorithm is any orthonormal matrix $\mathbf{W}^{*}$ that corresponds to this minimum:

$$E^{*}_{\mathrm{subj}} = \min\{E_{\mathrm{subj}}(\mathbf{W}, \tilde{\mathbf{W}})\} = \mathrm{E}_{\mathrm{subj}}(\mathbf{W}^{*}, \tilde{\mathbf{W}}) \tag{7}$$

A property of PCA is that there are an infinity of matrices $\mathbf{W}^{*}$ that satisfy this condition. Given one of them, if we replace $\tilde{\mathbf{W}}$ with $\mathbf{W}^{*}$, then the objective mean squared error takes on a new value:

$$E(\mathbf{W}^{*}) = E_{\mathrm{subj}}(\mathbf{W}^{*}, \mathbf{W}^{*}) \tag{8}$$

A crucial question is whether the error performed by the network at the current target, $E(\mathbf{W}^{*})$, will be lower than the current error, $E(\mathbf{W})$. If this was the case, we could use this result to demonstrate convergence of a Newton method, and establish the existence of solutions.

Unfortunately, this is not true. As we will see in Section 6, it sometimes happens that $E(\mathbf{W}^{*}) > E(\mathbf{W})$. For our current purpose, this negative result means that it is not possible to demonstrate the existence of solutions by Newton methods. For the same reason, a proof using infinitesimal quantities and monotonic decrease of E is not possible: the objective error does not always decrease along the path followed by the weights during learning.

3.3. Convergence of an ideal procedure

In order to demonstrate the existence of solutions, we will demonstrate the convergence of an iterative procedure that finds a solution. However, the proof involves hypotheses that are not satisfied by Oja's learning rule. Therefore, we consider an ideal learning procedure that has the properties we want. We shall do this, because this proof is not about the convergence of Oja's rule, but about the very existence of solutions. Indeed, this ideal procedure is a theoretical tool that does not correspond to a particular learning rule. The ideal procedure is defined as follows:

- Let $\mathbf{W}$ denote the weights at a given time. The ideal procedure chooses a target $\mathbf{W}^{*}$, and it updates $\mathbf{W}$ in the direction of $\mathbf{W}^{*}$, by a small amount. The assumption

that weights are updated *in the direction* of the target is part of the ideal procedure. Note that this assumption is not true for most learning rules. For example, during a gradient descent, the local direction of the gradient might differ from the direction of the attractor, even if the error surface is convex.

- Here is how the target is chosen: given $\mathbf{W}$, there exists an infinity of possible targets $\mathbf{W}^*$, that minimize $E_{subj}(\mathbf{W}^*, \mathbf{W})$. These possible targets belong to a subspace, which is characterized by $\mathbf{P}^* = \mathbf{W}^{*\mathrm{T}}\mathbf{W}^*$, where the projection $\mathbf{P}^*$ is unique and does not depend on the choice of $\mathbf{W}^*$. We assume that the ideal procedure picks the closest target, so that $\mathbf{W}$–$\mathbf{W}^*$ is orthogonal to the subspace of possible targets.

We will compare a modification of $\mathbf{P}$ to the induced change of $\mathbf{P}^*$. If $\mathbf{P}$ moves faster than $\mathbf{P}^*$, then the iterative procedure will converge, and the existence of solutions to Recursive PCA will be ensured. Let $\delta\mathbf{W}$ denote an infinitely small modification of $\mathbf{W}$ that results from the ideal procedure. Let $\delta\mathbf{P}$ denote the corresponding modification of $\mathbf{P}$, and let $\delta\mathbf{P}^*$ denote the resulting modification of the target $\mathbf{P}^*$. Let $\| \|_F$ denote the Frobenius norm (that is, the Euclidean norm of a vector that contains all the elements of a matrix). Given the preceding assumptions, we shall demonstrate the following result:

Theorem 1. *For all* $K \in [0;1[$, *there exists* $\alpha_K \in [0;1[$ *so that, if* $\alpha \leq \alpha_K$ *and if* $\delta\mathbf{P}$ *is in the direction of its target* $\mathbf{P}^*$, *then* $\|\delta\mathbf{P}^*\|_F \leq K\|\delta\mathbf{P}\|_F$

Proof. See Appendix A
□

This theorem means that there exists an interval of values of α where $\mathbf{P}$ moves faster than the target $\mathbf{P}^*$. For α in that interval, the ideal learning procedure will reach the target, in no more than $1/(1-K)$ times the number of iterations it would need to converge if the target was not moving. This demonstrates that $\mathbf{P}$ will reach a stable point through this ideal procedure, for α in that interval.

The existence of a stable point for $\mathbf{P}$ implies that the corresponding value of $\mathbf{W}$ does extract the principal components of the corresponding distribution of $\mathbf{z}$. This demonstrates the existence of solutions to the definition of Recursive PCA, in an interval of values of α.

However, as mentioned earlier, we experimentally observe numerical convergence of the learning rules (3) for any $\alpha \in [0;1]$. This suggests that solutions always exist, which is a much stronger statement. A general proof of convergence remains to be established.

4. Representing temporal context

In this section, we demonstrate how our network can be used to store and retrieve sequences of inputs, in the reverse order of presentation. We demonstrate that there exists a relation between the error function and the reconstruction error of previous events.

4.1. A logical stack

In Section 2, we briefly mentioned that $\mathbf{y}_t$ could be seen as a compressed representation of both $\mathbf{x}_t$ and $\mathbf{y}_{t-1}$. We now elaborate on this idea. At any time t, it is possible to reconstruct vector $\mathbf{z}_t$, by applying the transpose of $\mathbf{W}$ to $\mathbf{y}_t$. Let $\bar{\mathbf{z}}_t$ denote the reconstruction of $\mathbf{z}_t$:

$$\bar{\mathbf{z}}_t = \mathbf{W}^{\mathrm{T}}\mathbf{y}_t = \mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{z}_t \tag{9}$$

Vector $\bar{\mathbf{z}}_t$ can be separated into two sub-vectors that are the reconstructions of $\mathbf{x}_t$ and of $\mathbf{y}_{t-1}$, respectively:

$$\bar{\mathbf{z}}_t = \begin{bmatrix} \bar{\mathbf{x}}_t \\ \sqrt{\alpha}\bar{\mathbf{y}}_{t-1} \end{bmatrix} \tag{10}$$

In this expression, $\bar{\mathbf{x}}_t$ and $\bar{\mathbf{y}}_{t-1}$ denote the reconstructions of $\mathbf{x}_t$ and $\mathbf{y}_{t-1}$, respectively. The reconstruction of $\mathbf{y}_{t-1}$ allows us to recursively reconstruct previous events. Knowing an estimate $\bar{\mathbf{y}}_{t-k}$ of $\mathbf{y}_{t-k}$, we use $\mathbf{W}^{\mathrm{T}}$ to estimate $\mathbf{y}_{t-k-1}$. Using the convention $\bar{\mathbf{y}}_t = \mathbf{y}_t$, we may write, for all $k \geq 0$:

$$\bar{\mathbf{z}}_{t-k} = \begin{bmatrix} \bar{\mathbf{x}}_{t-k} \\ \sqrt{\alpha}\bar{\mathbf{y}}_{t-k-1} \end{bmatrix} = \mathbf{W}^T\bar{\mathbf{y}}_{t-k} \tag{11}$$

In this notation, it is implicit that all reconstructions are performed *at time t*. In fact, the reconstruction of a given vector will differ depending on when it is performed. A rigorous notation should, therefore, include an additional time index that would indicate the starting point of the reconstruction process. For simplicity, however, we do not use an additional index here. Hence, we always mean that reconstructions are performed at a given time t, after vector $\mathbf{x}_t$ has been measured and $\mathbf{y}_t$ has been computed.

Recursive PCA can be viewed as a generative model of past events, where estimates of the past events are retrieved in the reverse order of presentation. This property implements a logical stack, with 'push' and 'pop' operators. A 'push' corresponds to reading a new vector in $\mathbf{x}$, and applying $\mathbf{W}$ to $\mathbf{z}$. A 'pop' corresponds to reconstructing $\mathbf{z}$, by applying $\mathbf{W}^{\mathrm{T}}$ to $\mathbf{y}$.

4.2. Context and contextual mean-squared error

Given the above reconstruction procedure, it is possible to see the output of the network, $\mathbf{y}_t$, as a representation of all inputs measured so far. Formally, we define the *temporal context at time t*, as this ordered sequence of inputs: $\{\mathbf{x}_t; \mathbf{x}_{t-1}; \mathbf{x}_{t-2}; \ldots\}$. For convenience, we consider that this sequence is infinite. The question is now whether $\mathbf{y}_t$ is a good representation of context. For that, we shall examine whether reconstructed vectors are close to the corresponding input vectors. This will be assessed numerically in the experimental

sections. Before that, we will demonstrate that the objective error associated to Recursive PCA is a measure of the quality of a representation of context.

A good representation of temporal context should optimize its capacity, given the temporal statistics of the input series. For each integer k, it is possible to define the mean-squared error of the kth previous event:

$$e_k = \langle ||\mathbf{x}_{t-k} - \bar{\mathbf{x}}_{t-k}||^2 \rangle \tag{12}$$

To optimize capacity means to minimize the mean-squared errors e_k. Note that joint-minimizing a sum of several e_k's, associated to different values of k, is equivalent to finding the subspace of highest variance of a composite vector, that contains the corresponding $\mathbf{x}_{t-k}$'s. If the input time-series is not i.i.d., such a joint-minimization will exploit its time-dependencies.

However, it is not possible to minimize the sum of all e_k's, because this sum will be infinite. It is, therefore, necessary to make a choice, to forget some of events. We propose the following error criterion

$$E_\alpha = \sum_{k \geq 0} \alpha^k \mathrm{e}_k \tag{13}$$

where $0 < \alpha < 1$ is a *forgetting factor*. Since the time series $(\mathbf{x}_t)_{t \in \mathbb{Z}}$ is bounded, the sum E_α is finite. E_α is called the *contextual mean-squared error*.

A representation that minimizes E_α will jointly minimize all terms of the sum, and therefore it will learn the time dependencies of the input sequence. The forgetting term α implies a geometrical loss of importance of past events. Hence, a representation that minimizes E_α will progressively forgets past events.

We introduced the contextual mean-squared error because it is related to the error function of Recursive PCA. This relation is given by the following theorem:

Theorem 2. *The mean-squared reconstruction error associated to Recursive PCA is proportional to the contextual mean-squared error. More precisely*:

$$E = (1-\alpha)E_\alpha.$$

Proof. See Appendix B

□

This theorem implies that it is equivalent to minimize E and E_α. Hence, the objective error function of our network is a measure of how well past events are represented. However, it is not true that Recursive PCA always minimizes E, due to the difference between the objective and subjective errors. In the next sections, we will investigate this experimentally.

5. Computer simulations

In this section, we assess the performance of our network, trained on several time series. In each experiment, the initial value of $\mathbf{y}$ was zero, and the network was tested on a part of the time series different from the part that was used for training. The mean-squared reconstruction errors of previous events were measured, as explained in Sections 4.1 and 4.2. Reconstructions of the previous events were computed on each time step. For computational convenience, mean-squared reconstruction errors (e_k) were not computed, but estimated, using the following leaky averages:

$$\hat{e}_k(t) = (1-\gamma)\hat{e}_k(t-1) + \gamma ||\mathbf{x}_{t-k} - \bar{\mathbf{x}}_{t-k}||^2 \tag{14}$$

with $\gamma = 0.001$. The reconstruction of previous events and estimation of mean-squared errors was a pure monitoring process that used separate memory buffers and did not interfere with the network.

5.1. Binary time series

In a first experiment, the network was trained on two binary random time series. The input $\mathbf{x}$ was one-dimensional and it took on values $+1$ and -1. The output vector $\mathbf{y}$ used $m = 10$ neurons.

We compared two conditions. In condition A, the input was generated by picking $+1$ or -1 with probability 0.5 (coin toss). This i.i.d. series was used as a control condition. In condition B, the input was generated by a state machine with two states, labeled $+1$ and -1. On every time step, the probability of transition from one state to the other was equal to 0.3, while the probability of staying in a given state was 0.7. Hence, series B was not i.i.d.

After 20,000 training iterations the mean-squared errors $\hat{e}_k$ reached stable values, for $0 \leq k < 20$. These values are plotted on Fig. 1. In condition A (coin toss), the error curve has the shape of a step function. It means that exactly 10 events ($0 \leq k < 10$) are recovered with an error close to zero. For $k \geq 10$, the mean-squared error is equal to the variance of the input, which means that performance is not better than

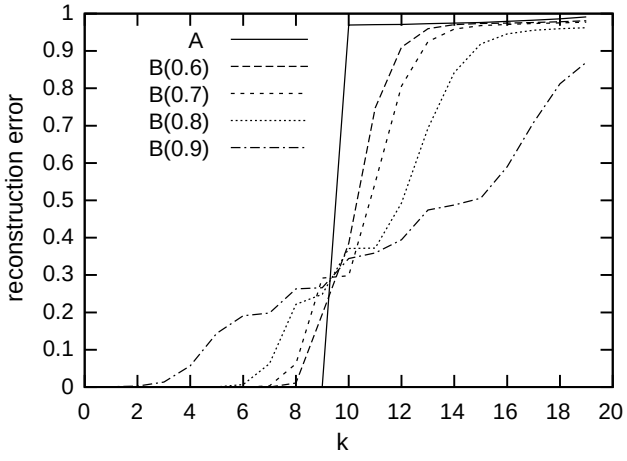

Fig. 1. Mean-squared reconstruction error, e_k, versus k, for binary random sequences. A: random input with no temporal structure. B(α): input generated by the state machine, using different values of the gain α.

chance. Hence, the network is able to unambiguously represent context of length 10. The shape of curve A was independent from the value of the gain α.

In condition B (state machine), different curves are obtained, that correspond to different values of the gain. Their general shape is a sigmoid, with a slope that depends on α. This means that temporal context can be retrieved beyond the former limit of 10 events. The counterpart is that reconstructions for $0 \leq k < 10$ are no longer perfect. This is consistent with the theory presented in Section 4.2, which predicted that temporal dependencies of the input should be exploited by the network, in order to optimize its representation of context.

When α decreases, the shape of the sigmoid $B(\alpha)$ approaches the shape of curve A. Hence, the gain seems to control a trade-off between the quality of the reconstructions and the depth of the neural stack. If there are no temporal dependencies, the representation is restricted to 10 events because no additional information can be gained from the statistics of the input. If there are temporal dependencies, higher values of the gain seem to favor the representation of older events, at the expenses of more recent events.

5.2. Mackey–Glass series

The Mackey–Glass time-series models the dynamics of white blood cell production in the human body (Mackey & Glass, 1977). This series is defined by the following differential equation, where time is continuous:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = bx(t) + \frac{ax(t-d)}{1 + x(t-d)^{10}} \tag{15}$$

For $d > 16.8$, both chaos and periodicity are present in the dynamics of the series. We used $a = 0.2$, $b = -0.1$, $d = 17$, and random initial conditions.

A linear network using $m = 30$ and $n = 1$ was trained on the Mackey–Glass series. The input was zero-centered, and sampled every time unit. The network was trained for 10^6 iterations, for different values of α. The input series at a given time step and its reconstruction, obtained after training, are shown on Fig. 2, for $\alpha = 0.99$. Mean-squared reconstruction errors of the 500 previous events were estimated. The value of $\hat{e}_k$ versus k is plotted on the bottom of Fig. 2, for $\alpha = 0.7$, 0.9 and 0.99. The variance of the input was 0.051.

For $\alpha = 0.7$, $\hat{e}_k$ increases by thresholds. The interval between two thresholds corresponds to the periodicity of the time series. Eventually, the reconstruction error becomes equal to the variance, as in the previous case. For $\alpha = 0.9$, the reconstruction error is much smaller; $\hat{e}_k$ is significantly lower than the variance of $\mathbf{x}$ for values of k up to 500. Unlike in the previous experiment, $\hat{e}_k$ does not increase monotonically with k; some past events are better reconstructed than other more recent events. This suggests that some periodic properties of the series have been learned.

For $\alpha = 0.99$, the reconstruction error is further reduced, except for recent events ($0 \leq k \leq 40$). Here too, the representation of older events is made at the expenses of

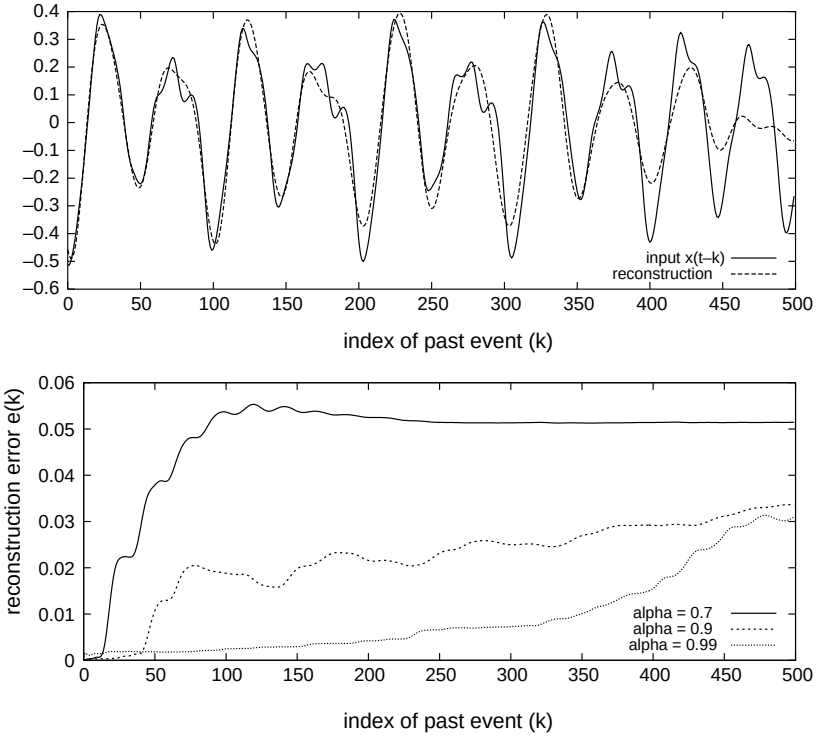

Fig. 2. Top: the (centered) Mackey–Glass chaotic series and its reconstruction for $\alpha = 0.99$. Bottom: mean-squared reconstruction errors of the Mackey–Glass series, for $k \leq 500$.

the representation of recent events. About 300 events are reconstructed with fairly good accuracy, while the representation uses only 30 neurons. Hence, the number of events that can be accurately reconstructed is significantly higher than the size of the network. This suggests that the representation of context devised by the network is highly compressed. According to theory, the network achieved this compression by learning the temporal statistics of the Mackey–Glass series.

6. Local minima

The results presented in the previous section suggest that Recursive PCA provides a good representation of context. However, we have not demonstrated that E_α is minimized. The purpose of this section is to show that this is almost the case.

A stable set of weights minimizes the 'subjective' error E_{subj}. However, there is no guarantee that this set of weights will minimize the objective error E. We will experimentally demonstrate the following results:

- The objective error landscape $E(\mathbf{W})$ is not convex, and it may have local minima.
- A minimum of the objective function $E(\mathbf{W})$ may not be a minimum of the subjective function $E_{\text{subj}}(\mathbf{W}, \tilde{\mathbf{W}})$.
- A minimum of the subjective function $E_{\text{subj}}(\mathbf{W}, \tilde{\mathbf{W}})$ may not be a minimum of the objective function $E(\mathbf{W})$.

These results mean that Recursive PCA does not minimize E_α. However, our experiments also suggest that the behaviour of the network improves when its dimension is increased. That is, if m increases, local minima tend to vanish and the error gets close to the global minimum.

6.1. Expression of E

In order to find local minima, it is useful to have an expression of E that is easy to measure. E is equal to the variance of $\mathbf{z}$ minus the variance of $\mathbf{y}$. Given, the definition of $\mathbf{z}$, E can be expressed as a function of the variances of $\mathbf{x}$ and $\mathbf{y}$ only:

$$E = \text{var}(\mathbf{x}) - (1-\alpha)\text{var}(\mathbf{y}) \tag{16}$$

where *var*() denote the variance of a random vector, that is, the trace of its covariance matrix. In this expression, *var*($\mathbf{x}$) is a constant, so it is equivalent to maximize the variance of $\mathbf{y}$, and to minimize the mean-squared error E. In order to show the existence of local minima of E, we will show that the variance of y has local maxima.

6.2. Existence of local minima of E

In Section 5.1, we used a state machine in order to generate a non-i.i.d. time series. Since this machine had only two states, the temporal structure of the input was very simple. It seems that this structure is too simple for the error function to have local minima.

However, it is possible to observe local minima by using a slightly more complex machine, with three states:

- *State 0*. The value of the input is 0.25. The probability of transition to state 1 is 0.1, and of transition to state 2 is 0.9.
- *State 1*. The value of the input is -0.4. The probability of transition to state 0 is 0.9, and of transition to state 2 is 0.1.
- *State 2*. The value of the input is 0.7. The probability of transition to state 1 is 0.25, and to stay in state 2 is 0.75.

These values were chosen so that the mean value of the input is close to zero, and so that it is possible to observe local minima. If the representation uses a single neuron ($m=1$), then only two weights are present. Although, the initial value of the weight vector is random, learning imposes that the weight matrix is orthonormal. Orthonormality is observed relatively quickly during learning, and it is maintained during the search for a minimum. Hence, we will consider that the matrix is orthonormal for visualization purposes. For $m=1$, this means that the constrained weight vector $[w_{\mathbf{x}}, w_{\mathbf{y}}]$ can be parameterized with one single scalar θ:

$$\begin{bmatrix} w_{\mathbf{x}} \\ w_{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \tag{17}$$

We use this parameterization because it facilitates visualization. The variance of $\mathbf{y}$ as a function of θ is plotted on Figs. 3 and 4, for four different values of the gain α: 0.4, 0.62, 0.76, and 0.9. For $\alpha=0.62$, 0.76, and 0.9, it is possible to see on the figures that the variance has two maxima. This demonstrates that the objective error function has local minima. For $\alpha=0.4$, there is only one maximum of the variance. This suggests that local minima occur for higher values of α.

In addition to variance, each graph displays the eigenvalues of the covariance matrix of $\mathbf{z}$, denoted by Σ. If the variance of $\mathbf{y}$ is lower than the first eigenvalue of Σ, we can deduce that the minimum of the subjective error function is not reached. Experimentally, we observe that all points where this minimum is not reached are unstable. Attractors are points where the first eigenvalue of $\boldsymbol{\Sigma}$ and the variance of $\mathbf{y}$ are equal.

- For $\alpha=0.4$, there is only one attractor. This point also maximizes objective variance.
- For $\alpha=0.9$, there are three different points where variance and the first eigenvalue of $\boldsymbol{\Sigma}$ are equal. Two of these points correspond to maxima of objective variance, and are attractors of the learning dynamics. The third point is at the limit between the two corresponding basins of attraction, and is unstable.

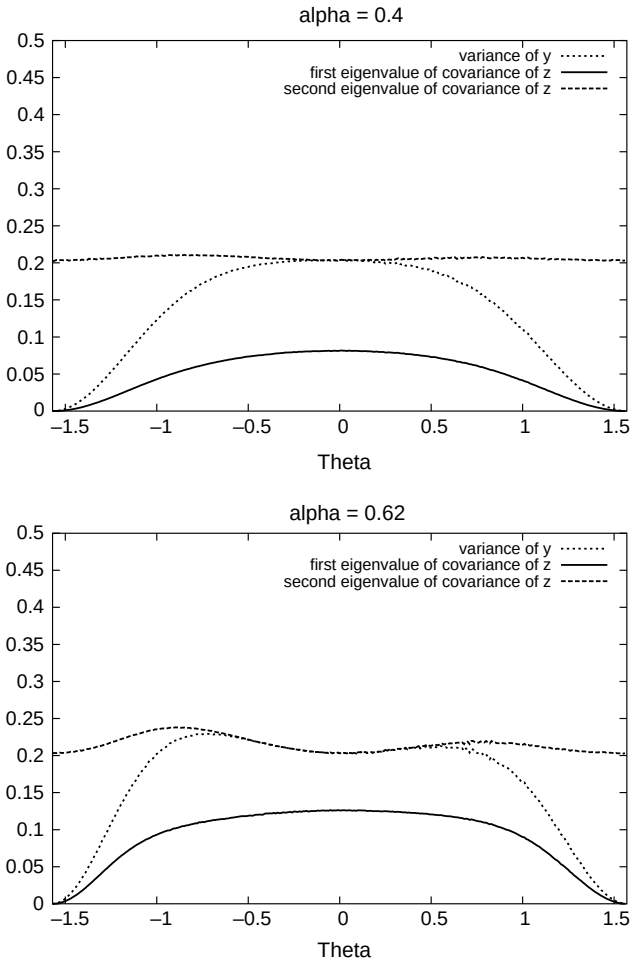

Fig. 3. Variance of **y**, and eigenvalues of the covariance of **z**, for $\alpha = 0.4$ and 0.62.

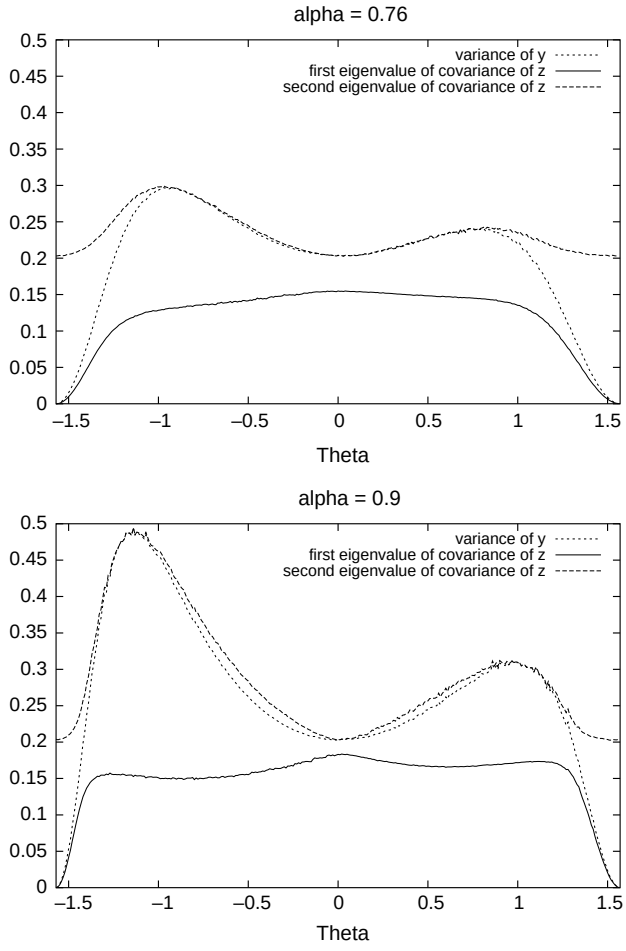

Fig. 4. Variance of **y**, and eigenvalues of the covariance of **z**, for $\alpha = 0.76$ and 0.9.

- The situations for $\alpha = 0.62$ and 0.76 are intermediate, and a bit more complex. For $\alpha = 0.62$, there is only one point of contact between the two curves, although this is difficult to see on the figure. Neither the local maximum of variance, at $\theta = 0.6$, nor its global maximum, at $\theta = -0.9$, are attractors of the learning dynamics. There is only one attractor, which is the point where variance is equal to the first eigenvalue of Σ, and it is located between the two maxima, at $\theta = -0.53$. Although, this point is not a maximum of objective variance, it does maximize subjective variance.
- For $\alpha = 0.76$ the variance curve and the curve of the first eigenvalue have two points of contact, and both are attractors. Although these points are maxima of subjective variance, they do not maximize objective variance.

Fig. 5 shows more details for $\alpha = 0.62$. The objective and subjective variance curves are plotted, as well as the attractor. The sinus-like curve is the subjective variance, i.e. the variance that **y** would have if the distribution of **z** was unchanged, equal to the distribution it has for $\theta = 0.6$ (local maximum). It shows that subjective variance is maximal for a value $\theta^* \neq 0.6$. As a consequence, the local maximum of the objective variance is unstable. Although the subjective variance at the global maximum $\theta = -0.9$ is

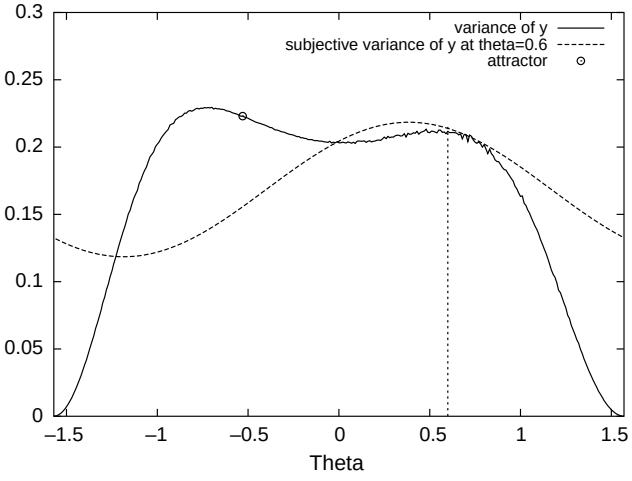

Fig. 5. Objective and subjective variance of **y**, for $\alpha = 0.62$. The sinusoidal curve is the subjective variance for $\theta = 0.6$. The unique attractor is indicated by a circle. This attractor is not a minimum of E.

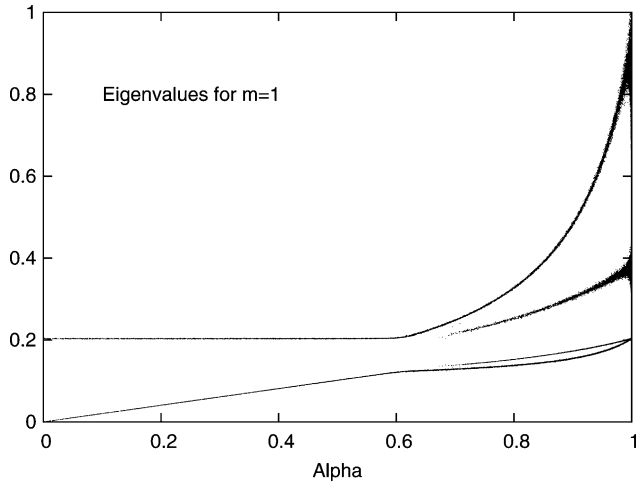

Fig. 6. Eigenvalues of the covariance matrix Σ of $\mathbf{z}$. The mean-squared error E is equal to the second eigenvalue. Bifurcations denote the existence of multiple attractors.

not plotted, it would display a similar situation. Hence, the maxima of the objective variance are unstable. The only stable point is the attractor at $\theta = -0.53$, where subjective variance is maximized.

6.3. Effect of the gain

We have seen that the shape of the error function depends on the value of the gain, α. For smaller values of α, there are no local minima, which is consistent with the limit case of standard PCA. Local minima appear when α is increased. Therefore, the attractor of E should display a bifurcation when α is increased.

In order to see this, the two eigenvalues of the covariance matrix Σ are displayed on Fig. 6, for $\alpha \in [0;1[$. For each point of the curve, a network was trained with random initial weights, and eigenvalues were measured once a stable value was reached. The figure shows that a bifurcation occurs between $\alpha = 0.6$ and 0.7. For $0 \leq \alpha \leq 0.6$, there is only one attractor, while there are two attractors for $0.7 \leq \alpha < 1$. This corresponds to what was shown in Fig. 3. In the interval $0.6 \leq \alpha < 0.7$, there is only one attractor, although the curves have a different shape than between 0 and 0.6. This interval corresponds to the region where the objective error function has local minima, but where these minima are unstable, as shown in Fig. 5.

6.4. Influence of the network's size

The previous bifurcation diagram is an efficient method to visualize local minima, because it scales well to higher dimensions. Fig. 7 shows the eigenvalues of Σ for four networks of higher dimension ($m = 2,3,5$ and 7). No bifurcation is observed for $m = 2$. Bifurcations occur again for $m = 3,5$ and 7. However, if one considers an eigenvalue of a given order, bifurcations tend to disappear when m is increased. This is balanced by the apparition of new bifurcations for higher-order eigenvalues.

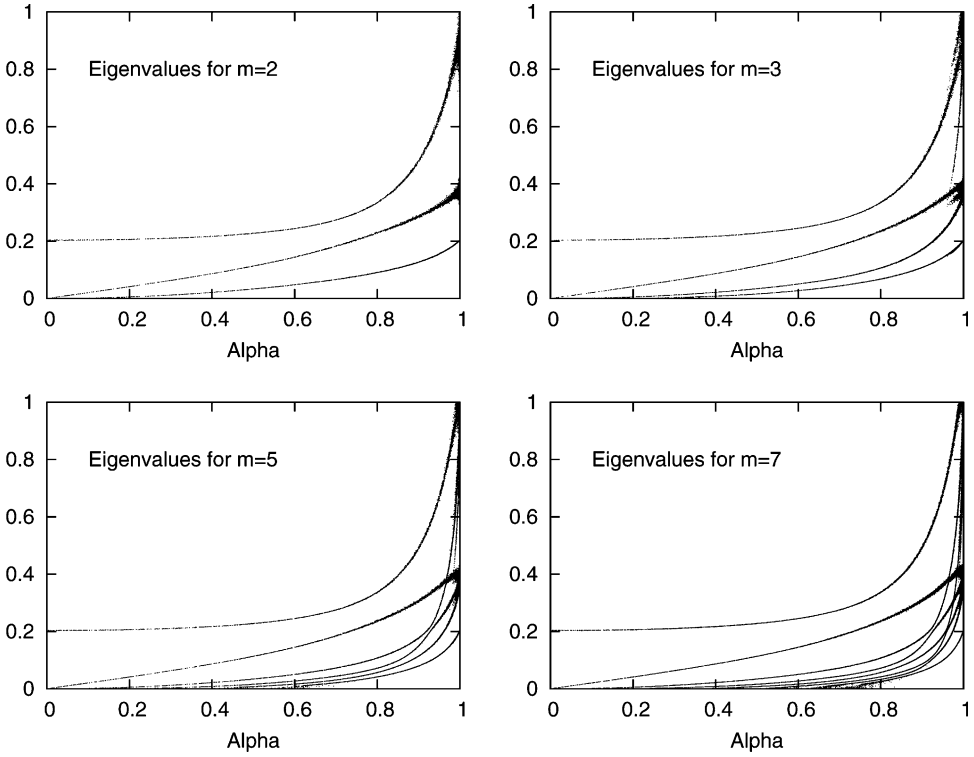

Fig. 7. Eigenvalues of the covariance matrix of $\mathbf{z}$, for $m = 2,3,5$ and 7.

The general picture is that bifurcations tend to disappear when the network's size is increased. This suggests that local minima tend to vanish. Because the mean-squared error can be expressed as a sum of eigenvalues, bifurcations that occur for higher-order eigenvalues will have a smaller effect on the error. Therefore, the overall effect of increasing the network's size should be a reduction of the influence of local minima. This suggests that local minima should not impair the performance of this type of network in large-scale applications.[4]

7. Conclusion

Using Oja's learning rule in a recurrent linear network, we have demonstrated that the network could store and retrieve arbitrary sequences. Our experiments demonstrate that an explicit recovery of sequences is possible, in reverse order. A trade-off between the representation of recent and older events is controlled by the gain α.

Our experiments also demonstrate that the network learns a representation that is adapted to the statistics of the input time series. If the input is i.i.d., then the network simply learns to store m previous vectors. If statistical dependencies are present in the input, then the network exploits them, in order to increase the number of events that it can accurately represent.

We made the assumption that the input series is stationary, in order to study learning in stable conditions. However, since our model is learning, this assumption may be removed, as long as the network has enough time to adapt to the changing dynamics of a non-stationary series. Although the learning rule does not always find the global minimum of the error function, we have demonstrated the following results:

- The mean-squared error associated to Recursive PCA is proportional to an exponentially weighted sum of reconstruction errors, called the contextual mean-squared error.
- Learning is a moving-target problem, where the error function minimized by the algorithm (subjective error) differs from the objective error function. Although, the subjective function is convex, the objective function may have local minima.
- Local minima of the objective error function tend to vanish when the dimension of the network is increased.

One could argue that using Oja's rule was a bad choice, and that we should have tried to find a method that actually minimizes the objective error function. Although it is perhaps possible to derive such a method, it does not seem to be simple; in any case, we believe that such a procedure would be computationally intensive, and that its complexity would be difficult to concile with a simple recurrent neural network, using a local Hebbian learning rule. Indeed, our goal was to study the learning dynamics of Oja's learning model, rather than to find a method that minimizes E at any cost.

A major emergent property of our algorithm is that it allows one to explicitly retrieve input sequences, in the reverse order of presentation. This provides a straightforward implementation of a logical stack. This result has implications, both from a theoretical and from a practical point of view. On the theory side, implementing a stack with neurons has been a research topic of its own, because it demonstrates the equivalence between artificial neural networks and other models of computation (Koiran, Cosnard, & Garzon, 1994; Moore, 1998; Pollack, 1987; Siegelmann & Sontag, 1995). In this context, our method is original, and the size of its memory does not depend on the level of precision of computations, as in most other models, but on the size of the network.

On the application side, a stack capability has been needed in several models of natural language processing (Miikkulainen, 1996; Sun, Giles, & Chen, 1997), and for the representation of complex structured objects (Pollack, 1990). Concerning this latter application, we have used our modified Oja rule in Pollack's Recursive Auto-Associative Memory network, in replacement of the standard error back-propagation procedure. We have shown that this learning model outperforms the original one on a language task (Voegtlin & Dominey, 2005).

Acknowledgements

The author would like to thank Prof. Yoshua Bengio and Dr Bruno Scherrer for their helpful comments. This work was supported by the French ministry of foreign affairs, and by the Alexander von Humboldt Foundation.

Appendix A. Proof of theorem 1

A.1. Notations

We will use the following notations. Let Σ_{xx}, Σ_{yy}, Σ_{xx}, denote the covariance matrices of vectors $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$, respectively. We may also denote the covariance of $\mathbf{z}$ simply by Σ:

$$\Sigma = \Sigma_{zz} = \langle \mathbf{z}\mathbf{z}^{\mathrm{T}} \rangle \tag{A1}$$

For this proof, we make the hypothesis that Σ is invertible, and that the norm of its inverse is bounded. This assumption reflects the fact that a learning procedure that maximizes output variance should not result, in general,

[4] In a previous application (Voegtlin, 2000), local minima were not observed numerically, probably because the dimension of the network was too high.

in a singular covariance matrix. This is in fact an assumption on the initial values of the weights, and on the complexity of the input time series.

Let $\Sigma^1_{\mathbf{xy}}$ denote the covariance matrix of vectors $\mathbf{x}_t$ and $\mathbf{y}_{t-1}$, and $\Sigma^1_{\mathbf{yx}}$ the transpose of $\Sigma^1_{\mathbf{xy}}$. More generally, let $\Sigma^i_{\mathbf{xy}}$ denote the covariance of $\mathbf{x}_t$ and $\mathbf{y}_{t-i}$, for all $i>0$:

$$\Sigma^i_{\mathbf{xy}} = \langle \mathbf{x}_t \mathbf{y}^{\mathrm{T}}_{t-i} \rangle \tag{A2}$$

and let $\Sigma^i_{\mathbf{yx}} = (\Sigma^i_{\mathbf{xy}})^{\mathrm{T}}$. Similarly, let $\Sigma^i_{\mathbf{xx}}$ denote the covariance of $\mathbf{x}_t$ and $\mathbf{x}_{t-i}$, for all $i>0$:

$$\Sigma^i_{\mathbf{xx}} = \langle \mathbf{x}_t \mathbf{x}^{\mathrm{T}}_{t-i} \rangle \tag{A3}$$

A.2. Speed of the projection matrix

In order to characterize the target, we will use the following property, which is a sufficient condition of PCA (Baldi & Hornik, 1988):

$$\mathbf{P}^* \Sigma = \Sigma \mathbf{P}^* = \mathbf{P}^* \Sigma \mathbf{P}^* \tag{A4}$$

Let $\delta\Sigma$ denote the small modification of Σ that results from a modification $\delta\mathbf{W}$ of the weights. Differentiating $\mathbf{P}^*\Sigma = \mathbf{P}^*\Sigma\mathbf{P}^*$ yields:

$$(\mathbf{P}^* + \delta\mathbf{P}^*)(\Sigma + \delta\Sigma) = (\mathbf{P}^* + \delta\mathbf{P}^*)(\Sigma + \delta\Sigma)(\mathbf{P}^* + \delta\mathbf{P}^*) \tag{A5}$$

We shall keeping only first-order terms

$$\delta\mathbf{P}^* \Sigma + \mathbf{P}^* \delta\Sigma = \mathbf{P}^* \Sigma \delta\mathbf{P}^* + \mathbf{P}^* \delta\Sigma \mathbf{P}^* + \delta\mathbf{P}^* \Sigma \mathbf{P}^* \tag{A6}$$

which is identical to

$$\delta\mathbf{P}^* \Sigma - \delta\mathbf{P}^* \Sigma \mathbf{P}^* - \mathbf{P}^* \Sigma \delta\mathbf{P}^* = \mathbf{P}^* \delta\Sigma (\mathbf{P}^* - \mathbf{I}) \tag{A7}$$

where $\mathbf{I}$ is the identity matrix of size m. Since $\mathbf{P}^*$ and $\mathbf{P}^* + \delta\mathbf{P}^*$ must be symmetric, $\delta\mathbf{P}^*$ is symmetric too. Therefore, $\delta\mathbf{P}^*\Sigma\mathbf{P}^* = \mathbf{P}^*\Sigma\delta\mathbf{P}^*$. We may thus write

$$\delta\mathbf{P}^* \Sigma - 2\delta\mathbf{P}^* \Sigma \mathbf{P}^* = \mathbf{P}^* \delta\Sigma (\mathbf{P}^* - \mathbf{I}) \tag{A8}$$

which yields

$$\delta\mathbf{P}^* \Sigma (\mathbf{I} - 2\mathbf{P}^*) = \mathbf{P}^* \delta\Sigma (\mathbf{P}^* - \mathbf{I}) \tag{A9}$$

Matrix $(\mathbf{I}-2\mathbf{P}^*)$ is idempotent, and $(\mathbf{I}-2\mathbf{P}^*)(\mathbf{P}^*-\mathbf{I}) = (\mathbf{P}^*-\mathbf{I})$. Therefore:

$$\delta\mathbf{P}^* = \mathbf{P}^* \delta\Sigma (\mathbf{P}^* - \mathbf{I}) \Sigma^{-1} \tag{A10}$$

Eq. (A10) describes how the target moves with the distribution. Since both $\mathbf{P}^*$ and $\mathbf{P}^* - \mathbf{I}$ are projectors, multiplying a matrix by these projectors does not increase, the Euclidean norm of its column vectors. It is, therefore, possible to give the following bound for the modification of the target:

$$||\delta\mathbf{P}^*||_{\mathrm{F}} \leq ||\delta\Sigma\Sigma^{-1}||_{\mathrm{F}} \tag{A11}$$

where $\|\ \|_{\mathrm{F}}$ denotes the Frobenius norm.

The above inequality relates a modification of the distribution $\delta\boldsymbol{\Sigma}$ and the modification of the target $\delta\mathbf{P}^*$ it induces. In order to compare the speed of $\mathbf{P}$ and $\mathbf{P}^*$, it is now sufficient to bound the modification of the distribution, $\delta\boldsymbol{\Sigma}$, that results from a small modification of the weights $\delta\mathbf{P}$.

A.3. Speed of the distribution

The covariance matrix of $\mathbf{z}$ reads:

$$\Sigma = \Sigma_{\mathbf{zz}} = \begin{bmatrix} \Sigma_{\mathbf{xx}} & \sqrt{\alpha}\Sigma^1_{\mathbf{xy}} \\ \sqrt{\alpha}\Sigma^1_{\mathbf{yx}} & \alpha\Sigma_{\mathbf{yy}} \end{bmatrix} \tag{A12}$$

Considering a small modification of the distribution of $\mathbf{z}$, it is possible to differentiate the covariance matrix:

$$\delta\Sigma = \sqrt{\alpha} \begin{bmatrix} 0 & \delta\Sigma^1_{\mathbf{xy}} \\ \delta\Sigma^1_{\mathbf{yx}} & \sqrt{\alpha}\delta\Sigma_{\mathbf{yy}} \end{bmatrix} \tag{A13}$$

Developing the Frobenius norm yields:

$$||\delta\Sigma||^2_{\mathrm{F}} = 2\alpha||\delta\Sigma^1_{\mathbf{xy}}||^2_{\mathrm{F}} + \alpha^2||\delta\Sigma_{\mathbf{yy}}||^2_{\mathrm{F}} \tag{A14}$$

Since $\mathbf{y}$ is a projection of $\mathbf{z}$, the covariance of $\mathbf{y}$ is smaller than the covariance of $\mathbf{z}$:

$$||\delta\Sigma_{\mathbf{yy}}||_{\mathrm{F}} \leq ||\delta\Sigma||_{\mathrm{F}} \tag{A15}$$

Replacing $\|\delta\Sigma_{\mathbf{yy}}\|_{\mathrm{F}}$ with its value in (31) yields:

$$||\delta\Sigma||_{\mathrm{F}} \leq \sqrt{\frac{2\alpha}{1-\alpha^2}}||\delta\Sigma^1_{\mathbf{xy}}||_{\mathrm{F}} \tag{A16}$$

In order to bound the right term, it is necessary to express the covariance $\Sigma^1_{\mathbf{xy}}$. Given the dynamics of the network (1), it is possible to express the covariance $\Sigma^1_{\mathbf{xy}}$ as:

$$\Sigma^1_{\mathbf{xy}} = \Sigma^1_{\mathbf{xx}} \mathbf{W}^{\mathrm{T}}_{\mathbf{x}} + \sqrt{\alpha}\Sigma^2_{\mathbf{xy}} \mathbf{W}^{\mathrm{T}}_{\mathbf{y}} \tag{A17}$$

Eq. (34) can be developed recursively, by applying the equality to all previous events. This yields the following sum:

$$\Sigma^1_{\mathbf{xy}} = \sum_{i=0}^{\infty} \alpha^{i/2} \Sigma^{i+1}_{\mathbf{xx}} \mathbf{W}^{\mathrm{T}}_{\mathbf{x}} (\mathbf{W}^{\mathrm{T}}_{\mathbf{y}})^i \tag{A18}$$

In this expression, all the $\Sigma^i_{\mathbf{xx}}$ are constant and do not depend on the weights. Hence, it is possible to differentiate this expression with respect to a small change of the weights:

$$\delta\Sigma^1_{\mathbf{x}y} = \sum_{i=0}^{\infty} \alpha^{i/2} \Sigma^{i+1}_{\mathbf{xx}} (\delta\mathbf{W}^{\mathrm{T}}_{\mathbf{x}} (\mathbf{W}^{\mathrm{T}}_{\mathbf{y}})^i + i\mathbf{W}^{\mathrm{T}}_{\mathbf{x}} \delta\mathbf{W}^{\mathrm{T}}_{\mathbf{y}} (\mathbf{W}^{\mathrm{T}}_{\mathbf{y}})^{i-1}) \tag{A19}$$

Since $\mathbf{W}$ is a projection, the Euclidean norm of $(\mathbf{W}^{\mathrm{T}}_{\mathbf{y}})^{i-1}$ is strictly lower than one. Therefore:

$$||\delta\Sigma^1_{\mathbf{x}y}||_{\mathrm{F}} \leq \sum_{i=0}^{\infty} \alpha^{i/2} \|\Sigma^{i+1}_{\mathbf{xx}} (\delta\mathbf{W}^{\mathrm{T}}_{\mathbf{x}} \mathbf{W}^{\mathrm{T}}_{\mathbf{y}} + i\mathbf{W}^{\mathrm{T}}_{\mathbf{x}} \delta\mathbf{W}^{\mathrm{T}}_{\mathbf{y}})\|_{\mathrm{F}} \tag{A20}$$

We want to compare the norms of $\delta\mathbf{P}^*$ and $\delta\mathbf{P}$. We shall differentiate $\mathbf{P} = \mathbf{W}^{\mathrm{T}}\mathbf{W}$:

$$\delta\mathbf{P} = \delta\mathbf{W}^{\mathrm{T}}\mathbf{W} + \mathbf{W}^{\mathrm{T}}\delta\mathbf{W} \tag{A21}$$

We assumed that, using our ideal procedure, $\delta\mathbf{W}$ is orthogonal to the space of matrices that correspond to $\mathbf{P}^*$. Therefore, changes to $\mathbf{W}$ that leave $\mathbf{P}$ unchanged should not be considered. Such changes are characterized by $\delta\mathbf{P}=0$, i.e. $\delta\mathbf{W}^{\mathrm{T}}\mathbf{W} = -\mathbf{W}^{\mathrm{T}}\delta\mathbf{W}$. This set of constraints defines a subspace of $\mathbb{R}^{n\times(n+m)}$, namely the eigenvector subspace of the linear application $\mathbf{A}\rightarrow\mathbf{W}\mathbf{A}^{\mathrm{T}}\mathbf{W}$, associated to eigenvalue -1. For $\mathbf{W}\mathbf{W}^{\mathrm{T}}=\mathbf{I}$, this application has only two eigenvalues, $+1$ and -1. Therefore, a change $\delta\mathbf{W}$ orthogonal to the -1 eigenvector subspace, must belong to the $+1$ eigenvector subspace, i.e. $\delta\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathbf{W}^{\mathrm{T}}\delta\mathbf{W}$. Therefore

$$\delta\mathbf{P} = 2\delta\mathbf{W}^{\mathrm{T}}\mathbf{W} \tag{A22}$$

which can be developed:

$$\delta\mathbf{P} = 2\begin{bmatrix} \delta\mathbf{W}_{\mathbf{x}}^{\mathrm{T}}\mathbf{W}_{\mathbf{x}} & \delta\mathbf{W}_{\mathbf{x}}^{\mathrm{T}}\mathbf{W}_{\mathbf{y}} \\ \delta\mathbf{W}_{\mathbf{y}}^{\mathrm{T}}\mathbf{W}_{\mathbf{x}} & \delta\mathbf{W}_{\mathbf{y}}^{\mathrm{T}}\mathbf{W}_{\mathbf{y}} \end{bmatrix} \tag{A23}$$

Considering the Frobenius norm of $\delta\mathbf{P}$, we obtain the following lower bounds:

$$||\delta\mathbf{P}||_{\mathrm{F}} \geq \sqrt{2}||\delta\mathbf{W}_{\mathbf{x}}^{\mathrm{T}}\mathbf{W}_{\mathbf{y}}||_{\mathrm{F}} \tag{A24}$$

$$||\delta\mathbf{P}||_{\mathrm{F}} \geq \sqrt{2}||\delta\mathbf{W}_{\mathbf{y}}^{\mathrm{T}}\mathbf{W}_{\mathbf{x}}||_{\mathrm{F}} \tag{A25}$$

Using these bounds and the fact that the Frobenius norm is sub-multiplicative we obtain from (A26):

$$||\delta\mathbf{\Sigma}_{\mathbf{xy}}^{1}||_{\mathrm{F}} \leq \sqrt{2}||\delta\mathbf{P}||_{\mathrm{F}} \sum_{i=0}^{\infty} \alpha^{i/2}(1+i)||\mathbf{\Sigma}_{\mathbf{xx}}^{i+1}||_{\mathrm{F}} \tag{A26}$$

Because $\mathbf{x}$ is stationary and bounded, the following upper bound is finite:

$$C_1 = \sup_{i\geq 0} ||\mathbf{\Sigma}_{\mathbf{xx}}^{i+1}|| \tag{A27}$$

We have made the assumption that $\mathbf{\Sigma}$ is invertible, and never gets too close to a singular matrix, so that there exists another bound C_2 such that:

$$||\mathbf{\Sigma}^{-1}|| \leq C_2 \tag{A28}$$

Combining inequalities (A11), (A16) and (A20) yields

$$||\delta\mathbf{P}^*||_{\mathrm{F}} \leq \sqrt{\frac{2\alpha}{1-\alpha^2}}\sqrt{2}C_1C_2\sum_{i=0}^{\infty}(\sqrt{\alpha})^i(1+i)||\delta\mathbf{P}||_{\mathrm{F}} \tag{A29}$$

which can be further simplified as:

$$\frac{||\delta\mathbf{P}^*||_{\mathrm{F}}}{||\delta\mathbf{P}||_{\mathrm{F}}} \leq 2C_1C_2\sqrt{\frac{\alpha}{1-\alpha^2}}\frac{1}{(1-\sqrt{\alpha})^2} \tag{A30}$$

The right term is a monotonically increasing function of α that takes the value zero in zero. Therefore, there exists an interval where is strictly lower than one, which completes the proof.

Appendix B. Proof of Theorem 2

The proof is based on the following lemma:

Lemma. *for all* $k>0$,

$$<||\mathbf{z}_{t-k}-\bar{\mathbf{z}}_{t-k}||^2> - <||\mathbf{y}_{t-k}-\bar{\mathbf{y}}_{t-k}||^2> = E$$

Proof of the lemma. $\mathbf{W}$ is of rank m and $\mathbf{W}\mathbf{W}^{\mathrm{T}}$ is equal to the identity matrix of size m. This allows us to develop and simplify the expression of the reconstruction error of $\mathbf{z}$, $A_t = ||\mathbf{z}_t - \bar{\mathbf{z}}_t||^2$:

$$A_t = ||\mathbf{z}_t - \mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{z}_t||^2$$

$$A_t = ||\mathbf{z}_t||^2 - 2\mathbf{z}_t^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{z}_t + \mathbf{z}_t^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{z}_t$$

$$A_t = ||\mathbf{z}_t||^2 - ||\mathbf{W}\mathbf{z}_t||^2$$

□

Matrix $\mathbf{W}$ can be decomposed using $\mathbf{W}_{\mathbf{x}}$ and $\mathbf{W}_{\mathbf{y}}$:

$$\mathbf{y}_t = \mathbf{W}_{\mathbf{x}}\mathbf{x}_t + \sqrt{\alpha}\mathbf{W}_{\mathbf{y}}\mathbf{y}_{t-1} \tag{B1}$$

$$\mathbf{W} = [\mathbf{W}_{\mathbf{x}}; \mathbf{W}_{\mathbf{y}}] \tag{B2}$$

We may express the reconstructions using the transposes $\mathbf{W}_{\mathbf{x}}^{\mathrm{T}}$, $\mathbf{W}_{\mathbf{y}}^{\mathrm{T}}$ of $\mathbf{W}_{\mathbf{x}}$ and $\mathbf{W}_{\mathbf{y}}$, for $k\geq 0$. Remember that we use the convention $\bar{\mathbf{y}}_t = \mathbf{y}_t$:

$$\bar{\mathbf{x}}_{t-k} = \mathbf{W}_{\mathbf{x}}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k} \tag{B3}$$

$$\bar{\mathbf{y}}_{t-k-1} = \mathbf{W}_{\mathbf{y}}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k} \tag{B4}$$

Now, let us develop the left term of the lemma, which we denote by B_t, for all $k>0$:

$$B_t = ||\mathbf{z}_{t-k} - \bar{\mathbf{z}}_{t-k}||^2 - ||\mathbf{y}_{t-k} - \bar{\mathbf{y}}_{t-k}||^2$$

$$B_t = ||\mathbf{x}_{t-k} - \bar{\mathbf{x}}_{t-k}||^2 + \alpha||\mathbf{y}_{t-k-1} - \bar{\mathbf{y}}_{t-k-1}||^2 - ||\mathbf{y}_{t-k} - \bar{\mathbf{y}}_{t-k}||^2$$

We develop the squared norms into dot products:

$$B_t = ||\mathbf{x}_{t-k}||^2 - 2\mathbf{x}_{t-k}^{\mathrm{T}}\bar{\mathbf{x}}_{t-k} + ||\bar{\mathbf{x}}_{t-k}||^2 + \alpha||\mathbf{y}_{t-k-1}||^2 - 2\alpha\mathbf{y}_{t-k-1}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k-1} + \alpha||\bar{\mathbf{y}}_{t-k-1}||^2 - ||\mathbf{y}_{t-k}||^2 + 2\mathbf{y}_{t-k}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k} - ||\bar{\mathbf{y}}_{t-k}||^2$$

Several terms can be replaced using (B3) and (B4):

$$B_t = ||\mathbf{x}_{t-k}||^2 - 2\mathbf{x}_{t-k}^{\mathrm{T}}\mathbf{W}_{\mathbf{x}}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k} + ||\mathbf{W}_{\mathbf{x}}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k}||^2 + \alpha||\mathbf{y}_{t-k-1}||^2 - 2\sqrt{\alpha}\mathbf{y}_{t-k-1}^{\mathrm{T}}\mathbf{W}_{\mathbf{y}}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k} + ||\mathbf{W}_{\mathbf{y}}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k}||^2 - ||\mathbf{y}_{t-k}||^2 + 2\mathbf{y}_{t-k}^{\mathrm{T}}\bar{\mathbf{y}}_{t-k} - ||\bar{\mathbf{y}}_{t-k}||^2$$

In this expression, $\bar{\mathbf{y}}_{t-k}$ can be factored:

$$B_t = ||\mathbf{x}_{t-k}||^2 - 2(\mathbf{W}_\mathbf{x}\mathbf{x}_{t-k} + \sqrt{\alpha}\mathbf{W}_\mathbf{y}\mathbf{y}_{t-k-1})^\mathrm{T}\bar{\mathbf{y}}_{t-k} + \bar{\mathbf{y}}_{t-k}^\mathrm{T}(\mathbf{W}_\mathbf{x}\mathbf{W}_\mathbf{x}^\mathrm{T} + \mathbf{W}_\mathbf{y}\mathbf{W}_\mathbf{y}^\mathrm{T})\bar{\mathbf{y}}_{t-k} + \alpha||\mathbf{y}_{t-k-1}||^2 - ||\mathbf{y}_{t-k}||^2 + 2\mathbf{y}_{t-k}^\mathrm{T}\bar{\mathbf{y}}_{t-k} - ||\bar{\mathbf{y}}_{t-k}||^2$$

The sum $\mathbf{W}_\mathbf{x}\mathbf{W}_\mathbf{x}^\mathrm{T} + \mathbf{W}_\mathbf{y}\mathbf{W}_\mathbf{y}^\mathrm{T}$ is equal to the identity matrix, and $\mathbf{y}_{t-k}$ can be identified in the second term. After simplification, we obtain:

$$B_t = ||\mathbf{x}_{t-k}||^2 + \alpha||\mathbf{y}_{t-k-1}||^2 - ||\mathbf{y}_{t-k}||^2$$

$$B_t = ||\mathbf{z}_{t-k}||^2 - ||\mathbf{W}\mathbf{z}_{t-k}||^2$$

The last expression does not contain any reconstruction. Therefore, the statistical expectation $<B_t>$ of B_t does not depend on the index k. For $k=0$ this expression is equal to $<A_t> = E$.

Proof of the theorem.

Let us develop $E = <||\mathbf{z}_t - \bar{\mathbf{z}}_t||^2>$:

$$E = <||\mathbf{x}_t - \bar{\mathbf{x}}_t||^2 + \alpha||\mathbf{y}_{t-1} - \bar{\mathbf{y}}_{t-1}||^2>$$

Using the lemma for $k=1$, we can replace the second term:

$$E = <||\mathbf{x}_t - \bar{\mathbf{x}}_t||^2> + \alpha <||\mathbf{z}_{t-1} - \bar{\mathbf{z}}_{t-1}||^2> - \alpha E$$

If we develop $\mathbf{z}_{t-1}$, we obtain the two first terms of E_α:

$$E = <||\mathbf{x}_t - \bar{\mathbf{x}}_t||^2> + < \alpha||\mathbf{x}_{t-1} - \bar{\mathbf{x}}_{t-1}||^2> + < \alpha^2||\mathbf{y}_{t-2} - \bar{\mathbf{y}}_{t-2}||^2> - \alpha E$$

This operation can be performed recursively, applying the lemma for each value of $k>0$. Eventually, this yields

$$E = <\sum_{k=0}^{\infty} \alpha^k||\mathbf{x}_{t-k} - \bar{\mathbf{x}}_{t-k}||^2> - \sum_{k=1}^{\infty} \alpha^k E$$

which can be simplified as:

$$E = (1-\alpha)E_\alpha$$

□

References

Baldi, P., & Hornik, K. (1988). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, *2*(1), 53–58.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA (Biophysics)*, *79*, 2554–2558.

Koiran, P., Cosnard, M., & Garzon, M. (1994). Computability with lowdimensional dynamical systems. *Theoretical Computer Science*, *132*(12), 113–128.

Maass, W., & Markram, H. (2003). Temporal integration in recurrent microcircuits. In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks* 2nd ed. (pp. 1159–1163). Cambridge, MA: MIT Press.

Maass, W., Natschlger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560.

Mackey, M. C., & Glass, L. (1977). Oscillations and chaos in physiological control systems. *Science*, *197*, 287–289.

Miikkulainen, R. (1996). Subsymbolic case-role analysis of sentences with embedded clauses. *Cognitive Science*, *20*, 47–73.

Moore, C. (1998). Dynamical recognizers: real-time language recognition by analog computers. *Theoretical Computer Science*, *201*(1–2), 99–136.

Morita, M. (1996). Associative memory with nonmonotone dynamics. *Neural Networks*, *6*, 115–126.

Oja, E. (1989). Neural networks, principal components and subspaces. *International Journal of Neural Systems*, *1*(1), 61–68.

Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, *46*, 77–105.

Pollack, J. B. (1987). *On Connectionist Models of Natural Language Processing*. PhD thesis, Univ. Illinois, Urbana.

Siegelmann, H. T., & Sontag, E. D. (1995). On the computational power of neural nets. *Journal of Computer and System Sciences*, *50*(1), 132–150.

Sun, G.-Z., Giles, C. L., & Chen, H. H. (1997). *The neural network pushdown automaton: Architecture, dynamics and training Summer school on neural networks* pp. 296–345.

Voegtlin, T. (2000). Learning principal components in a contextual space. In M. Verleysen (Ed.), *Proceedings of ESANN'2000* (pp. 359–364). Bruxelles: D Facto.

Voegtlin, T., Dominey, P. F. (2005). Linear recursive distributed representations. Neural Networks (in press).

Weigend, A. S., Huberman, B. A., & Rumelhart, D. E. (1990). Predicting the future: a connectionist approach. *International Journal of Neural Systems*, *1*(3), 193–209.

Williams, R., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *1*(1), 270–280.