Complex Systems **3** (1989) 269–293

Energy Functions in Neural Networks with Continuous Local Functions

F. Fogelman Soulié
C. Mejia
Laboratoire de Recherche en Informatique, bat. 490,
Université de Paris Sud, 91405 Orsay Cedex, France

Eric Goles
S. Martinez
Departamento de Matemáticas, Escuela de Ingeniería,
Universidad de Chile, Casilla 170, Correo 3, Santiago, Chile

Abstract. Neural networks with continuous local transition functions have been recently used for a variety of applications, especially in learning tasks and combinatorial optimization. Previous works have shown that Lyapunov — or "energy" — functions could be derived for networks of binary elements, thus allowing a rather complete characterization of their dynamics. We show here that it is possible to write down Lyapunov functions for continuous networks as well. We then use these functions to provide some results for the dynamical behavior of such networks. We discuss the link with the binary case and illustrate our results with some simulations.

1. Introduction

Work on neural networks can be traced back to the 1940s when McCulloch and Pitts [24] proposed a model of formal neurons based on binary elements. In 1982, Hopfield [19] stressed the analogy between spin glasses and formal neurons and renewed the interest for those networks. Meanwhile, many results have been reported which show that networks of continuous elements can be used in the same framework as binary networks and eventually do provide better results [5]. In particular, the recent literature on the so-called backpropagation algorithm [22] is based on such continuous elements.

In these models, the general question of the dynamics is usually not addressed. For binary networks, it is assumed that the "result" of the computation is a fixed point of the dynamics run on the network [19]: Lyapunov functions provide a tool for studying the dynamics. They allow characterization of the limit cycles (usually fixed points, see below) and the transient

times. For continuous networks, some results can be found [4,20] in the case where the dynamics is driven by a differential equation. However, when time is discrete and state continuous, which is the usual assumption in the "back-propagation" literature, no results are available. In fact, in these models, no advantage is taken of the dynamical dimension of the system: "recognition" is a one-shot process, whereby the data are passed only once through the network. However, simulation results [5–7] have shown clearly that in auto-association tasks (i.e., tasks where the input and the output are identical), performances are always better if the network is allowed to stabilize through successive iterations. This phenomenon can be intuitively understood as follows: each iteration step of the network allows the computed output to get closer to the desired output (usually a fixed point). Thus, allowing only one iteration usually prevents the network to "finish the job."

We give in this paper some theoretical results which lay the foundations for such an intuition in the case of networks with continuous elements and discrete time evolution: a Lyapunov function is exhibited. Since it decreases along time, the network, under appropriate assumptions on the connection matrix, can only be driven to fixed points (for sequential iterations) or limit cycles of period 2 at most (for parallel iterations). This allows us to provide bounds for the transient.

The results are then applied in the case of multilayer networks and simulations are shown for illustration in associative memory tasks.

In section 2, we give some definitions, then state our results in the case of parallel iterations (section 3) and sequential iterations (section 4). Simulation results are then given in section 5 to illustrate the theory in the case of multilayer networks trained by the gradient backpropagation (GBP) algorithm.

2. Definitions

2.1 Automata networks

An *automaton* will be defined here as an element which has an internal state s in some state space S, receives inputs $s_1, \ldots, s_n$, and changes state by using some transition function f. An *automata network* is an ensemble of interconnected automata, the inputs being either the internal states of the automata or signals sent from the environment along the connections of the network. The total weighted input to automaton i is A_i:

$$A_i = \sum_{k=1\ldots n} w_{ik} s_k - b_i \tag{2.1}$$

where s_k is the state of automaton k and n is the number of automata in the network.

2.2 Dynamics on automata networks

A dynamical evolution of an automata network is defined through a rule which tells how the state vector $s(t)$ changes with time. Various models

have been proposed depending on whether the time variable t is discrete or continuous.

1. In the continuous case, Grossberg and coworkers [4] propose a differential equation:

$$\frac{d}{dt}s_i = a_i(s_i)[b_i(s_i) - \sum_{k=1...n} w_{ik}d_k(s_k)] \tag{2.2}$$

a particular case of which is used by Hopfield [20] and Pineda [26,27]:

$$c_i\frac{d}{dt}s_i = -s_i/R_i + \sum_{k=1...n} w_{ik}f(s_k) + I_i \tag{2.3}$$

2. In the discrete case, the dynamics can be viewed as an iteration and various iteration modes are classically used:

In the *parallel* iteration mode, all automata change state, one at a time. The dynamics is thus defined by

$$s_i(t+1) = f[A_i(t)] \tag{2.4}$$

where

$$A_i(t) = \sum_{k=1...n} w_{ik}s_k(t) - b_i \tag{2.5}$$

In the *sequential* iteration mode, the automata change state one at a time in a prescribed order. If, for example, this order is the natural permutation of $1 \ldots n$, the dynamics can be viewed as if at time $t + m/n$, element m only changed state

$$s_m(t+1) = f[A_m(t + (m-1)/n)] \tag{2.6}$$

with

$$\begin{aligned} A_m(t + (m-1)/n) \;=\; & w_{m1}s_1(t+1) + \ldots \\ & + w_{mm-1}s_{m-1}(t+1) \\ & + \sum_{k=m...n} w_{mk}s_k(t) - b_m \end{aligned} \tag{2.7}$$

The network state at time $t + m/n$ is thus

$$s(t+m/n) = (s_1(t+1), \ldots, s_m(t+1), s_{m+1}(t), \ldots, s_n(t)) \tag{2.8}$$

More iteration modes have been studied in [14].

2.3 Lyapunov functions

In some cases, Lyapunov functions have been derived to study the asymptotic behavior of the network; typically, the trajectory would then lead to a fixed point.

2.3.1 Continuous state, continuous time

Cohen-Grossberg [4] show that, under the assumptions

$$w_{ik} = w_{ki} \quad a_i(s) \geq 0, \qquad d_j(s) \geq 0 \tag{2.9}$$

function V defined by

$$V(s) = -\sum_{i=1\ldots n} \int_0^{s_i} b_i d_i'(x)dx + 1/2 \sum_{i,k=1\ldots n} w_{ik} d_i(s_i) d_k(s_k) \tag{2.10}$$

is a Lyapunov function for the dynamics given in (2.2).

In the Hopfield's case [19], the "energy" function (2.10) becomes

$$V(s) = \sum_{i=1\ldots n} \int_0^{f(s_i)} 1/R_i f^{-1}(x)dx \; - \; 1/2 \sum_{i,k=1\ldots n} w_{ik} f(s_i) f(s_k) \; - \sum_{i=1\ldots n} I_i f(s_i) \tag{2.11}$$

2.3.2 Discrete state, discrete time

This is the case for example for perceptrons [25], adalines [29], and so on. Lyapunov functions have been proposed for different classes of transition functions.

If $S = \{0, 1\}$ and the transition function f is a threshold function, i.e.:

$$f(u) = \begin{cases} 1 & \text{if } u \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

it can be shown [9,19] that the mapping V defined by

$$V(s) = -1/2 \sum_{i=1\ldots n} s_i \sum_{k=1\ldots n} w_{ik} s_k + \sum_{i=1\ldots n} b_i s_i \tag{2.13}$$

is a Lyapunov function for the sequential iteration, if W is a symmetric matrix with nonnegative diagonal.

In the parallel iteration case

$$V[s(t)] = -\sum_{i,k=1\ldots n} w_{ik} s_i(t+1) s_k(t) + \sum_{i=1\ldots n} b_i [s_i(t) + s_i(t+1)] \tag{2.14}$$

is a Lyapunov function [14] under the only condition that W be symmetric.

Those results were extended to other iteration modes (e.g., block sequential or random [14]) or transition functions: multithreshold [8], majority [12,14], positive [13,15], cellular [17,18].

2.3.3 Continuous state, discrete time

This is the case in particular for linear associative memories [21], ("Brain-state-in-the-box") BSB models [11], and multilayer networks [28].

In the case of the BSB model, the dynamics is defined [11] by

$$s_i(t+1) = \begin{cases} F & \text{if } \alpha \sum_{k=1\ldots n} w_{ik}s_k(t) + s_i(t) \geq F \\ \alpha \sum_{k=1\ldots n} w_{ik}s_k(t) + s_i(t) & \text{if } |\alpha \sum_{k=1\ldots n} w_{ik}s_k(t) + s_i(t)| \leq F \\ -F & \text{if } \alpha \sum_{k=1\ldots n} w_{ik}s_k(t) + s_i(t) \leq -F \end{cases} \tag{2.15}$$

It has been shown [11] that

$$V(s) = -1/2 \sum_{i=1\ldots n} s_i \sum_{k=1\ldots n} w_{ik}s_k \tag{2.16}$$

is a Lyapunov function for the dynamics defined by (2.15) provided that W is symmetric and either positive semi-definite or $\alpha < 2/|\lambda_{\min}|$, where $\lambda_{\min}$ is the minimum eigenvalue of W.

This is the only case to our knowledge where a Lyapunov function has been derived for a discrete time-continuous state model. Note that all the functions given so far are Lyapunov functions under the requirement that the connection matrix W be symmetric.

At present, the most pervasive neural network model is the multilayer network trained by the gradient backpropagation rule. It is used as a "one-shot" decision tool: an input is presented, one "iteration" run, and the updated state is considered as the output of the system. However, when the output is of the same dimensionality as the input (as in tasks of auto-association or "identity mapping" [31]), simulations have shown that performances could be improved by allowing various iterations [5,6]. It would then be useful to have the Lyapunov function tool to study the asymptotic behavior of this model, which we do in the following sections.

3. Parallel iterations

In the following two sections, we use an automata network of n elements. Each automaton is quasi-linear with a transition function $f : \mathcal{R} \to \mathcal{R}$, continuous, strictly increasing on an interval $S =]a, b[$ $(a < b)$, and constant outside:

$$\begin{aligned} &\forall x \leq a, \quad f(x) = f(a) \\ &\forall x \geq b, \quad f(x) = f(b) \end{aligned} \tag{3.1}$$

For example, f could be a truncated "sigmoidal" function (figure 1) similar to those classically used in multilayered networks [7].

As previously, we define the total weighted input to i by

$$A_i = \sum_{k=1\ldots n} w_{ik}s_k - b_i \tag{3.2}$$

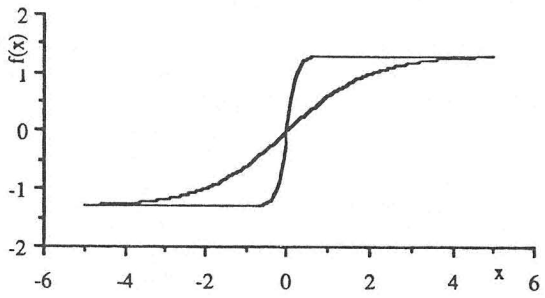

Figure 1: Sigmoid function. The figure shows a quasilinear automaton: $s_i = f(A_i)$ where f is a sigmoid function $f(x) = s[e^{kx} - 1]/[e^{kx} + 1]$. Parameter $T = 1/k$ is the "temperature."

and the parallel iteration on the network by

$$[s_i(t+1) = f[A_i(t)] \tag{3.3}$$

We then have the following theorem.

Theorem 1. *Let V be defined by*

$$V[s(t)] = \sum_{i,k=1\ldots n} w_{ik} s_i(t) s_k(t-1) - \sum_{i=1\ldots n} [\int_c^{A_i(t-1)} f(s)ds + \int_c^{A_i(t)} f(s)ds] \tag{3.4}$$

where $c \in S$ is arbitrary.

Then, if W is symmetric, V is a Lyapunov function for the parallel iteration.

Proof. We will prove that V is decreasing along any trajectory.

Let

$$\Delta_t V = V[s(t+1)] - V[s(t)] \tag{3.5}$$

We will show that $\Delta_t V \leq 0$, $\forall t \geq 0$.

$$\begin{aligned}\Delta_t V &= \sum_{i,k=1\ldots n} w_{ik} s_i(t+1) s_k(t) - \sum_{i=1\ldots n} [\int_c^{A_i(t)} f(s)ds \\ &+ \int_c^{A_i(t+1)} f(s)ds] - [\sum_{i,k=1\ldots n} w_{ik} s_i(t) s_k(t-1) \\ &- \sum_{i=1\ldots n} [\int_c^{A_i(t-1)} f(s)ds + \int_c^{A_i(t)} f(s)ds]]\end{aligned}$$

Assuming W is symmetric, we then have

$$\begin{aligned}\Delta_t V &= \sum_{i,k=1\ldots n} w_{ik} s_i(t)[s_k(t+1) - s_k(t-1)] \\ &- \sum_{i=1\ldots n} [\int_c^{A_i(t+1)} f(s)ds - \int_c^{A_i(t-1)} f(s)ds]\end{aligned}$$

From (3.3), it follows:

$$\begin{aligned}\Delta_t V &= \sum_{i,k=1\ldots n} w_{ik} f[A_i(t-1)][s_k(t+1) - s_k(t-1)] \\ &- \sum_{i=1\ldots n} [\int_{A_i(t-1)}^{A_i(t+1)} f(s)ds]\end{aligned}$$

and thus from (3.2):

$$\begin{aligned}\Delta_t V &= \sum_{i=1\ldots n} f[A_i(t-1)][A_i(t+1) - A_i(t-1)] \\ &\quad - \sum_{i=1\ldots n} [\int_{A_i(t-1)}^{A_i(t+1)} f(s)ds] \\ &= \sum_{i=1\ldots n} f[A_i(t-1)][A_i(t+1) - A_i(t-1)] \\ &\quad - [A_i(t+1) - A_i(t-1)]f(d_i) \\ \Rightarrow \Delta_t V &= \sum_{i=1\ldots n} [A_i(t+1) - A_i(t-1)][f(A_i(t-1)) - f(d_i)]\end{aligned} \tag{3.6}$$

where $d_i \in]A_i(t-1),\ A_i(t+1)[$ (from the mean value theorem).

Since f is increasing, it follows that $\Delta_t V \leq 0$. ∎

Remark 1. Equation (3.6) could also be obtained from the limit to the multithreshold case of the threshold expression [16], by making use of a morphism between the two cases.

Theorem 2. *Let W be symmetric and $s(0), \ldots, s(T-1)$ be a limit cycle of period T. Then $T \leq 2$.*

Proof. To prove theorem 2, we just have to show that $s(t) = s(t+2)$, $\forall t \geq 0$. From theorem 1, we have

$$\begin{aligned}&V[s(0)] \geq V[s(1)] \geq \ldots \geq V[s(T-1)] \geq V[s(T)] = V[s(0)] \\ \Rightarrow\ &\Delta_t V = 0, \forall t \geq 0\end{aligned}$$

Then, from (3.6) it follows:

$$\begin{aligned}[A_i(t+1) - A_i(t-1)]f[A_i(t-1)] &= \\ \int_{A_i(t-1)}^{A_i(t+1)} f(s)ds &= [A_i(t+1) - A_i(t-1)]f[d_i]\end{aligned}$$

with

$$d_i \in]A_i(t-1), A_i(t+1)[$$

Case 1. $A_i(t-1) \leq a$
Then (3.1) $\Rightarrow s_i(t) = f[A_i(t-1)] = f(a)$
If $A_i(t+1) \leq a$, then $s_i(t+2) = f(a) = s_i(t)$, which ends the proof.
If $A_i(t+1) > a$, then $A_i(t+1) - A_i(t-1) \neq 0$ and:

$$\begin{aligned}\int_{A_i(t-1)}^{A_i(t+1)} f(s)ds &= \int_{A_i(t-1)}^{a} f(s)ds + \int_{a}^{A_i(t+1)} f(s)ds \\ &= f(a)[a - A_i(t-1)] + [A_i(t-1) - a]f(d_i)\end{aligned}$$

where $d_i \in]a, A_i(t+1)[$

$$\Rightarrow \quad [A_i(t+1) - a][f(d_i) - f(a)] = 0$$

which is impossible from the strict monotonicity of f.

Case 2. $a < A_i(t-1) < b$
If $A_i(t+1) \in S$ then:

either $A_i(t-1) = A_i(t+1) \Rightarrow s_i(t) = s_i(t+2)$, which ends the proof

or $A_i(t-1) \neq A_i(t+1) \Rightarrow f[A_i(t-1)] = f(d_i)$, which is impossible.

The cases $A_i(t+1) \leq a$ or $A_i(t+1) \geq b$ are impossible (the argument goes similarly to case 1).

Case 3. $A_i(t-1) \geq b$ The proof is similar to case 1.

Corollary 1. *In the case where $c = 0$ and $f(0) = 0$, we have*

$$\begin{aligned}V[s(t)] = &- \sum_{i,k=1\ldots n} w_{ik}s_i(t+1)s_k(t) + \sum_{i=1\ldots n} b_i[s_i(t) + s_i(t+1)] \quad (3.7) \\ &+ \sum_{i=1\ldots n} [\int_0^{s_i(t)} f^{-1}(y)dy + \int_0^{s_i(t+1)} f^{-1}(y)dy]\end{aligned}$$

Proof.

$$\begin{aligned}V[s(t)] &= \sum_{i,k=1\ldots n} w_{ik}s_i(t)s_k(t-1) - \sum_{i=1\ldots n} [\int_c^{A_i(t-1)} f(s)ds \qquad (3.8) \\ &+ \int_c^{A_i(t)} f(s)ds] \\ &= \sum_{i=1\ldots n} [\sum_{k=1\ldots n} w_{ik}s_k(t-1)]s_i(t) - \sum_{i=1\ldots n} [\int_c^{A_i(t-1)} f(s)ds \\ &+ \int_c^{A_i(t)} f(s)ds]\end{aligned}$$

Since f is continuous, strictly increasing on S, it has an inverse f^{-1} and we have, $\forall c, d \in S$:

$$\begin{aligned}[d-c][f(d) - f(c)] = \int_c^d f(x)dx - [d-c]f(c) \;&+\; \int_{f(c)}^{f(d)} f^{-1}(y)dy \\ &- \; c[f(d) - f(c)]\end{aligned}$$

$$\Rightarrow \int_c^d f(x)dx + \int_{f(c)}^{f(d)} f^{-1}(y)dy = df(d) - cf(c)$$

$$\begin{aligned}\Rightarrow V[s(t)] &= \sum_{i=1\ldots n} s_i(t)[A_i(t-1) + b_i] + \sum_{i=1\ldots n} [\int_{f(c)}^{s_i(t)} f^{-1}(y)dy \\ &+ \int_{f(c)}^{s_i(t+1)} f^{-1}(y)dy] - \sum_{i=1\ldots n} [s_i(t)A_i(t-1) \\ &+ s_i(t+1)A_i(t) - 2cf(c)]\end{aligned}$$

$$\begin{aligned}\Rightarrow V[s(t)] &= -\sum_{i,k=1\ldots n} w_{ik}s_i(t+1)s_k(t) \\ &+ \sum_{i=1\ldots n} b_i[s_i(t) + s_i(t+1)] + 2ncf(c) \\ &+ \sum_{i=1\ldots n} [\int_{f(c)}^{s_i(t)} f^{-1}(y)dy + \int_{f(c)}^{s_i(t+1)} f^{-1}(y)dy]\end{aligned}$$

Note that (3.7) is similar to the expression given by Hopfield for the sequential case (2.11): our result is thus an extension of Hopfield's. Equation (3.7) is also quite similar to the Lyapunov function found for the discrete state space case:

$$V[s(t)] = -\sum_{i,k=1\ldots n} w_{ik}s_i(t+1)s_k(t) + \sum_{i=1\ldots n} b_i[s_i(t) + s_i(t+1)] \qquad (3.9)$$

This is related to the fact that continuous state automata can be shown to behave in the average in the same way as binary automata with noise [1,7,30].

Theorem 3. *Let W be symmetric and $\Delta V, \Delta_0$ be defined by:*

$$\begin{aligned}\Delta V &= \max\{V[s(t)]/s(0) \in \mathcal{R}\} - \min\{V[s(t)]/s(0) \in \mathcal{R}\} \\ \Delta_0 &= \min\{|\Delta_1 V|/s(0) \in \mathcal{R}, s(1) \neq s(3)\}\end{aligned}$$

then, if $f(\mathcal{R})$ is finite or Δ_0 is nonzero, the transient length L of any trajectory is either 0 or 1 or bounded by

$$L \leq \Delta V/\Delta_0 \qquad (3.10)$$

Proof. Suppose $L \geq 2$ and let $s(0), s(1), \ldots, s(L), s(L+1), s(L), s(L+1), \ldots$ be a trajectory.

Then $s(1) \neq s(3)$; hence $\Delta_1 V \neq 0$.

Thus $\Delta_0 \neq 0$ (either by using that assumption or as a min of nonzero $\Delta_1 V$ on a finite set).

$$\begin{aligned}V[s(t+1)] &= V[s(t+1)] - V[s(t)] + V[s(t)] - V[s(t-1)] \\ &\quad + \ldots + V[s(1)] \\ &= \sum_{k=1,\ldots,t} (V[s(k+1)] - V[s(k)]) + V[s(1)] \\ &= \sum_{k=1,\ldots,t} \Delta_k V + V[s(1)]\end{aligned}$$

$$\forall t < L, s(t+2) \neq s(t)$$

$$\Rightarrow \quad \exists i(1,\ldots,n) : s(t+2) \neq s(t)$$

$$\Rightarrow \quad A_i(t+1) \neq A_i(t-1)$$

$$\Rightarrow \quad \Delta_t V \neq 0$$

But

$$\begin{aligned} \Delta_t V &= V[s(t+1)] - V[s(t)] \\ &= V[x(2)] - V[x(1)] \\ &= \Delta_0 \quad \text{with } x(0) = s(t-1) \end{aligned}$$

$$\Rightarrow \Delta_t V \leq -\Delta_0$$

$$\Rightarrow \quad \forall t < L, V[s(t+1)] \leq -(t+1)\Delta_0 + V[s(1)]$$

$$\Rightarrow \quad \min V \leq V[s(t+1)] \leq -(t+1)\Delta_0 + \max V$$

$$\Rightarrow \quad \min V \leq V[s(t+1)] \leq -(L-1+1)\Delta_0 + \max V$$

which ends the proof. ∎

Corollary 2. *Under the assumptions of theorem 3 and if in addition f is a multithreshold function:*

$$f(u) = \sum_{k=0\ldots p} \alpha_k \cdot 1_{]\theta_k, \theta_{k+1}[}(u)$$

where $\alpha_0 < \ldots < \alpha_{k-1} < \alpha_k < \alpha_{k+1} < \ldots < \alpha_p$ and $-\infty = \theta_0 < \theta_1 < \ldots < \theta_{k-1} < \theta_k < \theta_{k+1} < \ldots < \theta_p < \theta_{p+1} = +\infty$, then there exists $\eta > 0$ such that the transient length is bounded by

$$L \leq \| W \| [\underline{\alpha}^2 - \alpha_0 \alpha_p]/\eta \Delta\alpha \qquad (3.11)$$

where $\|W\| = \sum_{ik} |w_{ik}|, \underline{\alpha} = \max(\alpha_p, -\alpha_0), \Delta\alpha = \inf\{\alpha_{k+1} - \alpha_k, k = 0 \ldots p\}$.

Proof. We have

$$A_i(t) = \sum_k w_{ik} s_k(t) = \sum_{k:w_{ik}>0} w_{ik} s_k(t) + \sum_{k:w_{ik}<0} w_{ik} s_k(t)$$

$$\begin{aligned} \Rightarrow \quad & \alpha_0 \sum_{k:w_{ik}>0} w_{ik} + \alpha_p \sum_{k:w_{ik}<0} w_{ik} \leq A_i(t) \\ & \leq \alpha_p \sum_{k:w_{ik}>0} w_{ik} + \alpha_0 \sum_{k:w_{ik}<0} w_{ik} \end{aligned}$$

Let us denote

$$\|W\| = \sum_{ik} |w_{ik}| = \|W_+\| + \|W_-\|$$

with

$$\|W_+\| = \sum_{k:w_{ik}>0} w_{ik} \text{ and } \|W_-\| = -\sum_{k:w_{ik}<0} w_{ik}$$

We thus have:

$$\alpha_0\|W_+\| - \alpha_p\|W_-\| \leq A_i(t) \leq \alpha_0\|W_+\| - \alpha_p\|W_-\|$$

Let us denote

$$\underline{A} = \alpha_0\|W_+\| - \alpha_p\|W_-\|$$

and

$$\overline{A} = \alpha_p\|W_+\| - \alpha_o\|W_-\|$$

We have

$$\underline{A} \leq A_i(t) \leq \overline{A}$$

$$\int_0^{\overline{A}} f(s)ds = \left(\sum_{k=0}^{p} \alpha_k[\max(\theta_k, 0) - \min(\theta_{k+1}, \overline{A})]\right)^+$$

$$\int_{\underline{A}}^{0} f(s)ds = \left(\sum_{k=0}^{p} \alpha_k[\max(\theta_k, \underline{A}) - \min(\theta_{k+1}, 0)]\right)^+$$

Hence:

$$-\int_{\overline{A}}^{0} f(s)ds \leq \int_0^{A_i(t)} f(s)ds \leq \int_0^{\overline{A}} f(s)ds$$

Now, by using the same method, we can bound $\sum_{ik} w_{ik}s_is_k$:

$$\alpha_0\alpha_p \sum_{k:w_{ik}>0} w_{ik} + \underline{\alpha}^2 \sum_{k:w_{ik}<0} w_{ik} \leq \sum_{ik} w_{ik}s_is_k \leq \underline{\alpha}^2 \sum_{k:w_{ik}>0} w_{ik} + \alpha_0\alpha_p \sum_{k:w_{ik}<0} w_{ik}$$

Hence:

$$\alpha_0\alpha_p\|W_+\| - \underline{\alpha}^2\|W_-\| \leq \sum_{ik} w_{ik}s_is_k \leq \underline{\alpha}^2\|W_+\| - \alpha_0\alpha_p\|W_-\|$$

Let us denote

$$\underline{W} = \alpha_0\alpha_p\|W_+\| - \underline{\alpha}^2\|W_-\|$$

and

$$\overline{W} = \underline{\alpha}^2 \|W_+\| - \alpha_0 \alpha_p \|W_-\|$$

We have

$$\underline{W} \leq \sum_{ik} w_{ik} s_i s_k \leq \overline{W}$$

Now, the Lyapunov function V can be bounded:

$$\underline{W} - 2n \int_0^{\overline{A}} f(s)ds \leq V[s(t)] \leq \overline{W} + 2n \int_{\underline{A}}^0 f(s)ds$$

hence

$$\begin{aligned} \Delta V &\leq \overline{W} - \underline{W} + 2n \int_{\underline{A}}^{\overline{A}} f(s)ds \\ &\leq \overline{W} - \underline{W} = [\underline{\alpha}^2 - \alpha_0 \alpha_p][\|W_+\| + \|W_-\|] \\ &\leq [\underline{\alpha}^2 - \alpha_0 \alpha_p]\|W\| \end{aligned}$$

Note that, in the particular case where $\alpha_p = -\alpha_0 = a$, we have

$$\Delta V \leq 2a^2 \|W\|$$

From theorem 3, we have $L \leq \Delta V / \Delta_0$. We have just bounded ΔV; we now have to bound Δ_0, which will be done by bounding $\Delta_t V$.

$$\Delta_t V = \sum_i [A_i(t+1) - A_i(t-1)][f(A_i(t-1)) - f(d_i)]$$

with

$$d_i \in]A_i(t-1), A_i(t+1)[$$

Let

$$L_i(t) = [A_i(t+1) - A_i(t-1)][f(A_i(t-1)) - f(d_i)]$$

$L_i(t) \leq 0$, from theorem 1.

If $s(t) \neq s(t+2)$, then there exists k such that:

$$s_k(t) \neq s_k(t+2) \quad \text{i.e.} \quad f[A_k(t-1)] \neq f[A_k(t+1)\}$$

$$\Rightarrow A_k(t-1) \in]\theta_{k_1}, \theta_{k_1+1}]$$

$$\text{and } A_k(t+1) \in]\theta_{k_2}, \theta_{k_2+1}] \quad \text{with } k_1 \neq k_2$$

We suppose, in the following, that $A_k(t+1) > A_k(t-1) > 0$. The proof for the other cases is similar.

Then:

$$\begin{aligned} L_k(t) &= [A_k(t+1) - A_k(t-1)][f(A_k(t-1)) - f(d_k)] \\ &= -[A_k(t+1) - A_k(t-1)][\alpha - \alpha_{k_1}] \quad \text{where } \alpha \in [\alpha_{k_1}, \alpha_{k_2}] \\ \Rightarrow L_k(t) &\leq -[A_k(t+1) - \theta_{k_1}] \cdot \Delta\alpha \end{aligned}$$

We define

$E = \{h : \{1 \ldots n\} \to \{0 \ldots p\}\}$. E is a finite set.

$E_{jk} = \{h \in E / \sum_i w_{ji}\alpha_h(i) - \theta_k > 0\}$. E_{jk} is also a finite set and thus:

$\eta_{jk} = inf\{\sum_i w_{ji}\alpha_{h(i)} - \theta_k / h \in E_{jk}\} > 0$

and so

$$\eta = \inf \eta_{jk} > 0$$

Thus

$$A_k(t+1) - \theta_{k_1} \geq \eta$$

$$\Rightarrow \quad \Delta_t V \leq L_k(t) \leq -\eta.\Delta\alpha \Rightarrow \quad |\Delta_t V| \geq \eta.\Delta\alpha$$

$$\begin{aligned} \Rightarrow \quad L &\leq \Delta V/\Delta_0 \\ &\leq [\underline{\alpha}^2 - \alpha_0\alpha_p]\|W\|/\eta.\Delta\alpha \end{aligned}$$

which in the case where: $\alpha_p = -\alpha_0 = a$ is just

$$L \leq 2a^2\|W\|/\eta.\Delta\alpha$$

Corollary 3. *Under the assumptions of theorem 3 and if in addition f is sigmoidal truncated:*

$$f(u) = \begin{array}{ll} s[e^{ku} - 1]/[e^{ku} + 1] & \text{if } u[-a, +a] \\ -s & \text{if } u \leq -a \\ s & \text{if } u \geq a \end{array}$$

where $s = f(a)$. Then the transient length is either 0 or 1 or is bounded by

$$L \leq 2\|W\|s^2[3 + 2n]/\Delta_0 \qquad (3.12)$$

Proof. It is easy to see that, from the assumptions on f:

$$-s \leq s_i(t) \leq s$$

$$-s\|W_i\| \leq A_i(t) \leq s\|W_i\| \quad \text{with } \|W_i\| = \sum_k |w_{ik}|$$

c in (3.4) was arbitrary in S. ∎

In the following, we will take $c = -\|W\|s$.

If this does not lie in $S =]-a, a[$, we could take $c = a[-\|W\|s]$: for a small enough, c would be in S, but the bound in (3.11) would have to be slightly modified (see below).

Then

$$A_i(t) \geq c \quad \forall i, t$$

$$\Rightarrow \quad \int_c^u f(x)dx = (u-c)f(d) \quad \text{with } d \in]c, u[$$

$$\Rightarrow \quad -(u-c)s \leq \int_c^u f(x)dx \leq (u-c)s$$

From (3.4) it follows that

$$\begin{aligned} -\|W\|s^2 \; - \; & s \sum_{i=1\ldots n} [A_i(t-1) + A_i(t) - 2c] \\ \leq \; & V[s(t)] \leq \|W\|s^2 - s \sum_{i=1\ldots n} [A_i(t-1) + A_i(t) - 2c] \end{aligned}$$

$$\begin{aligned} \Rightarrow \quad -\|W\|s^2 \; - \; & s \sum_{i=1\ldots n} [2s\|W_i\| - 2c] \\ \leq \; & V[s(t)] \leq \|W\|s^2 + s \sum_{i=1\ldots n} [2s\|W_i\| - 2c] \end{aligned}$$

With $c = -\|W\|s$, the bound in (3.11) follows.
If $c = \alpha[-\|W\|s]$, the bound must be modified to

$$L \leq 2\|W\|s^2[3 + 2n\alpha]/\Delta_0$$

Remark 2. Corollaries 2 and 3 allow to compare the transient times for the threshold and the sigmoidal cases. Suppose that we have on one side a network made up from threshold units:

$$\begin{aligned} f(u) = 1 \quad & \text{if } u \geq 0 \\ -1 \quad & \text{otherwise} \end{aligned}$$

and on the other a network with elements using a (truncated) sigmoid function (3.1) with $b = -a = 1$, then

$$\begin{aligned} L_t &\leq 2\|W\|/\Delta 0t \quad && \text{for the threshold case,} \\ L_s &\leq 2\|W\|[3 + 2n]/\Delta 0s \quad && \text{for the sigmoid case} \end{aligned}$$

Hence, if ever $\Delta 0t$ and $\Delta 0s$ are the same, then the sigmoid case should require more iterations than the threshold case: we will verify this result in the simulations presented in section 5.

4. Sequential iterations

The dynamics of the network is now defined by

$$s_m(t+1) = f[A_m(t+(m-1)/n)] \tag{4.1}$$

with $m = 1, \ldots, n$ and

$$\begin{aligned} A_m(t+(m-1)/n) &= w_{m1}s_1(t+1) + \ldots + w_{mm-1}s_{m-1}(t+1) \\ &\quad + \sum_{k=m\ldots n} w_{mk}s_k(t) - b_m \end{aligned} \tag{4.2}$$

Theorem 4. *Let V be defined by*

$$\begin{aligned} V[s(t)] = &- 1/2 \sum_{i,k=1\ldots n} w_{ik}s_i(t)s_k(t) \\ &+ \sum_{i=1\ldots n} [\int_O^{s_i(t)} f^{-1}(s)ds + \sum_{i=1\ldots n} b_i s_i(t) \end{aligned} \tag{4.3}$$

Then, if W is symmetric, with nonnegative diagonal, V is a Lyapunov function for the sequential iteration.

Proof. We will give the proof in the case where the threshold $b_i = 0$. The extension to the general case is straightforward.

$$\begin{aligned} \Delta V &= V[s(t+1)] - V[s(t)] \\ &= \sum_i V_i \end{aligned}$$

with

$$\begin{aligned} V_i = &- [s_i(t+1) - s_i(t)] \sum_k w_{ik}s_k(t) - w_{ii}/2[s_i(t+1) - s_i(t)]^2 \\ &+ \int_0^{s_i(t+1)} f^{-1}(s)ds - \int_0^{s_i(t)} f^{-1}(s)ds \end{aligned}$$

The quadratic term $-w_{ii}/2[s_i(t+1)-s_i(t)]^2$ is clearly negative. For the other terms, let us denote

$$\begin{aligned} u &= \sum_k w_{ik}s_k(t) \\ v &= \sum_k w_{ik}s_k(t-1) \end{aligned}$$

Then

$$-[f(u) - f(v)]u + \int_0^{f(u)} f^{-1}(s)ds - \int_0^{f(v)} f^{-1}(s)ds$$

Since

$$\int_0^{f(x)} f^{-1}(s)ds = xf(x) - \int_0^x f(s)ds$$

we have

$$-[f(u)-f(v)]u + \int_0^{f(u)} f^{-1}(s)ds - \int_0^{f(v)} f^{-1}(s)ds = (u-v)f(v) + \int_0^{v} f(s)ds - \int_0^{u} f(s)ds \leq 0$$

which ends the proof. ∎

Remarks. The Lyapunov function, given in (4.3), is very similar to expression (2.11) given by Hopfield for the case of sequential iterations where time is continuous. We could also derive in a way similar to the parallel case a bound for the transient length.

5. Simulations

We will present in this section simulation results that we have run on multilayer networks trained by the gradient backpropagation (GBP) algorithm [23,28]. We assume that the network has one hidden layer only and denote by W_1 and W_2 the weight vectors from the input to the hidden layer and from the hidden to the output layer, respectively. We restrict our study to the auto-association case — or identity mapping — where the input and output vectors have the same dimensionality.

Let n be the dimension of the inputs x and p the number of hidden units. W_1 is a $p \times n$ matrix and W_2 a $n \times p$ matrix. Let us now consider the automata network with $n+p$ elements, no hidden layer, and a connection matrix W defined by

$$W = \begin{vmatrix} 0 & W_2 \\ W_1 & 0 \end{vmatrix} \tag{5.1}$$

This network is obtained from the previous one by folding the output layer on the input layer.

Then, running the multilayer network with connection weights W_1 and W_2 in the "usual" way, i.e., layer after layer, is equivalent to running the network with connection matrix W in block-sequential fashion, i.e., running in parallel all first n elements, and then run in parallel the last p elements. In that case, the output of the multilayer network when it is shown a vector x as input can be read as the next state of the network with no hidden layer after it has run one iteration on the initial state x.

It has been shown [3] that when all the elements of the multilayer network are linear, or at least the elements in the output layer are, then one solution of the GBP algorithm is the principal component analysis matrix where

$$\begin{aligned} W_1 &= U_p^t \\ W_2 &= U_p \end{aligned} \tag{5.2}$$

and U_p is a $n \times p$ matrix, the columns of which are the eigenvectors of the inputs covariance matrix XX^t associated to its p largest eigenvalues.

This link with standard data analysis techniques also stands in the case of classification tasks (heteroassociation): the linear multilayer network has then been shown to perform a discriminant analysis on the inputs [10].

In the case of (5.1,2), it is clear that W is a symmetric matrix with nonnegative diagonal. Hence, all the results in the previous sections apply to the network with no hidden layer when it is run either in parallel or in sequential. By using techniques very similar to those of the previous sections, it could be easy to extend our results to block sequential iteration, which we will not do here (such extensions were proved [14] for discrete state space). Taking this extension for granted, we can then apply all the results of the previous sections to the multilayer network as well.

We have run simulations for an auto-association task [6,7]: random binary patterns 16 bits long are generated and a network is trained to retrieve those patterns from noisy inputs. The gradient backpropagation algorithm is used for training. The training set contains 10 noisy versions at noise level 1, 2, and 3 (i.e., with 1, 2, or 3 bits inverted) for m patterns set at random from $\{-1,+1\}^n$, with $n = 16$. We used training sets with $m = 10$ and 20.

Two architectures were tested:

1. a network with two layers: the input and the output.
2. a network with three layers: the input and output layers and one hidden layer with $p = 8$ (for $m = 10$) and 14 (for $m = 20$) units.

All the units had a sigmoidal transition function (figure 1). It is well known [5] that if the units were linear, then the network with no hidden layer would just compute the optimal auto-associative map of Kohonen [21]. However, if it is further tested with nonlinear mappings, then [5] the sigmoid mapping yields better results than the threshold mapping. By varying the "temperature" T of the sigmoid, we can test for the influence of the nonlinearity of the mapping. We used two different values of temperature T : $T = 1$ and $T = 1/8$ (figure 1). We will see that our simulation results demonstrate that the influence of the temperature is not important when the network is trained with the same nonlinear mapping.

In order to illustrate the results of the previous sections, we had to make sure that the assumption of symmetry (5.2) was valid. We thus measured the deviation from this assumption by computing

$$\delta^2(W_1, W_2) = \frac{\|W_2 - W_1^t\|^2}{\frac{\|W_1\|^2 + \|W_2\|^2}{2}} \tag{5.3}$$

Figure 2 shows that this deviation remains relatively small.

Training was always practically complete: all m "pure" patterns were correctly memorized in both architectures, even though they were never presented to the network during learning (figure 3). Learning was slightly slower for $T = 1/8$ than for $T = 1$: parameters were harder to tune, especially in the architecture with hidden layer. The network without hidden layer (shown

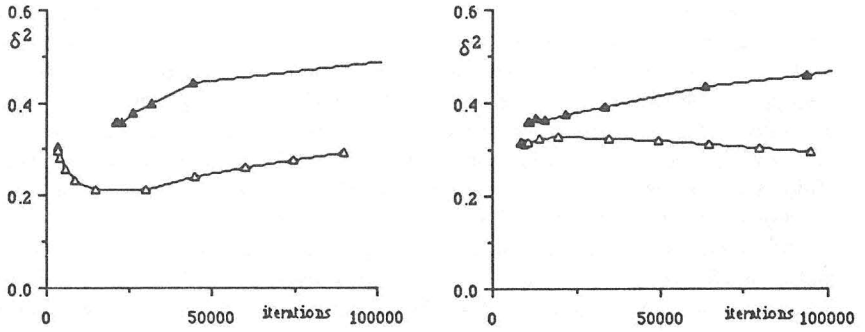

Figure 2: Deviation from symmetry for the two layer networks for $T = 1$ (left) and $T = 1/8$ (right).

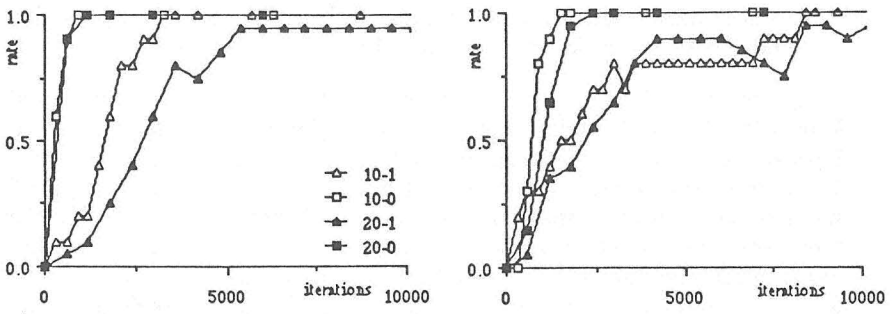

Figure 3: Learning for the two network architectures for $T = 1$ (left) and $T = 1/8$ (right). The symbol marked $m - k$ refers to the network with k hidden layer ($k = 0$ or 1) trained on m patterns. The unit on the iteration axis is the presentation of one example.

by a rectangle: 10-0 and 20-0) always learned faster than the network with hidden layer (shown by a triangle).

Generalization was then investigated: the m memorized patterns were modified by inverting d bits and the networks were then tested on these noisy patterns. We first tested for generalization as usually done for multilayer networks, by presenting the input to the network and then checking the output (one shot process). The results (figure 4) show that the network with one hidden layer did better than the other and also that memorizing

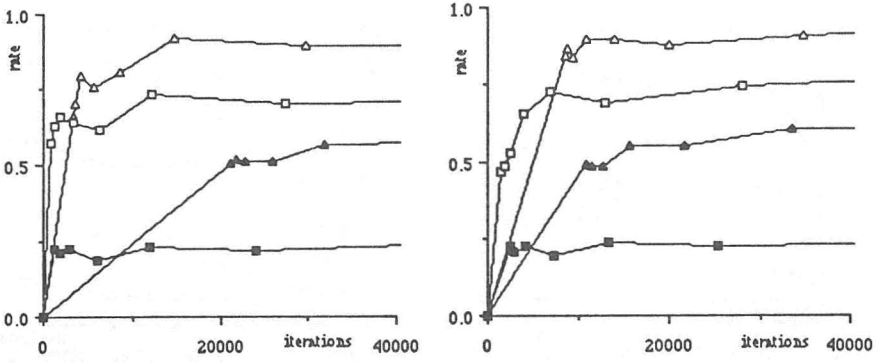

Figure 4: Generalization rates of patterns at distance 1 for the different architectures, for $T = 1$ (left) and $T = 1/8$ (right).

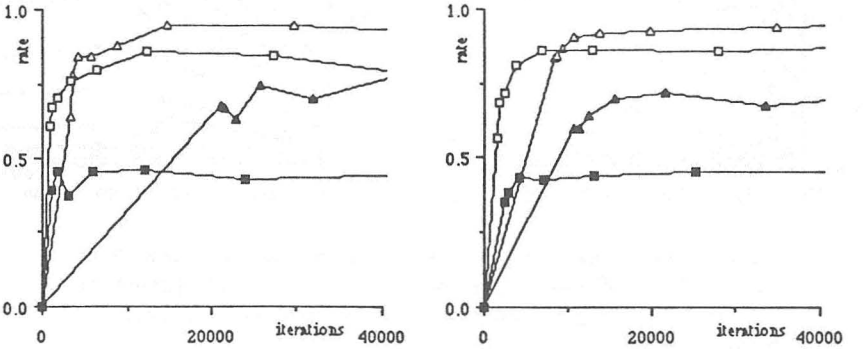

Figure 5: Iterated generalization rates of patterns at distance 1 for the different architectures, for $T = 1$ (left) and $T = 1/8$ (right).

20 patterns was probably beyond the capacity of the networks. (However, performances degraded "gracefully" even in this case of overloading.)

We then allowed for iteration: the input was presented to the network, the output read and then refed into the network until "correct" retrieval, i.e., corresponding bit signs identical in computed and desired outputs. The results show that the performances were always increased, especially in those cases where the performances were relatively poor, i.e., for the network with no hidden layer, for the network with a hidden layer, overloaded with 20 patterns (figure 5), or the retrieval of very noisy patterns (d large: figure 6).

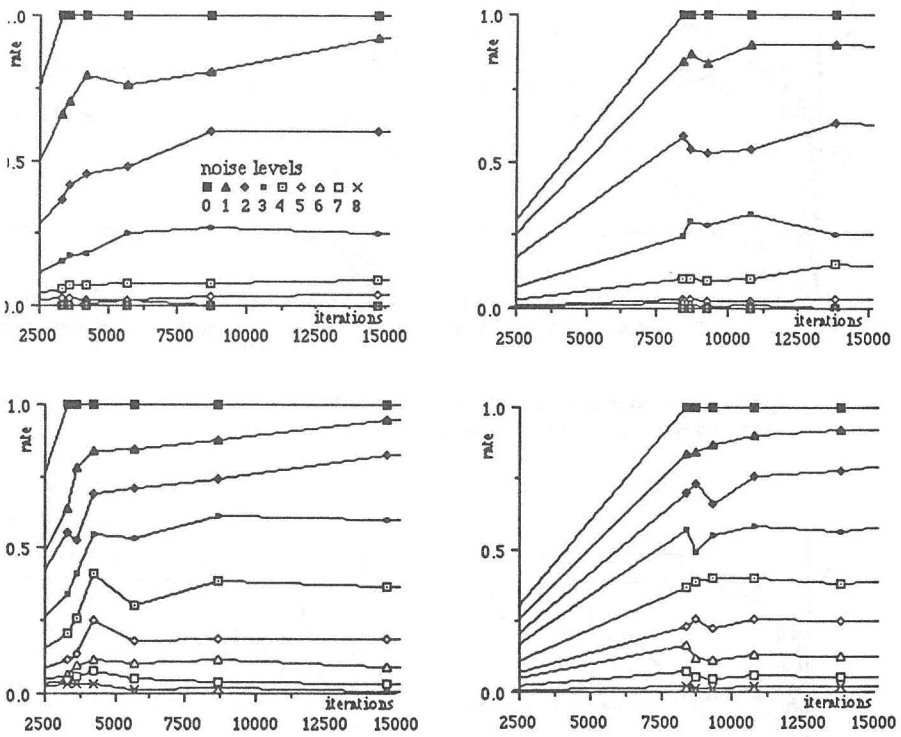

Figure 6: Generalization rates for the one hidden layer network at different levels of noise, for $T = 1$ (left) and $T = 1/8$ (right), in the case of one-step retrieval (top) and iterated retrieval (bottom). Architecture 10-1.

In the iterated generalization experiment, we also estimated the average transient length necessary for convergence: this is plotted in figure 7, which shows that the transient length is remarkably small and gradually increases with the difficulty of the task. It usually takes just one step to retrieve a pattern at distance 1 and three steps for a pattern at distance 5.

Those results thus show that allowing for iterative generalization helps improve the generalization performances, thus supporting the intuition that each "pass" through the network produces some reduction of the noise-to-signal ratio at the output. The theory we have developed also indicates that only fixed points could be reached by iteration. We thus tested whether this was true for our networks and also what the fixed points obtained were.

Figure 8 shows that all "pure" patterns lead to fixed points. These were

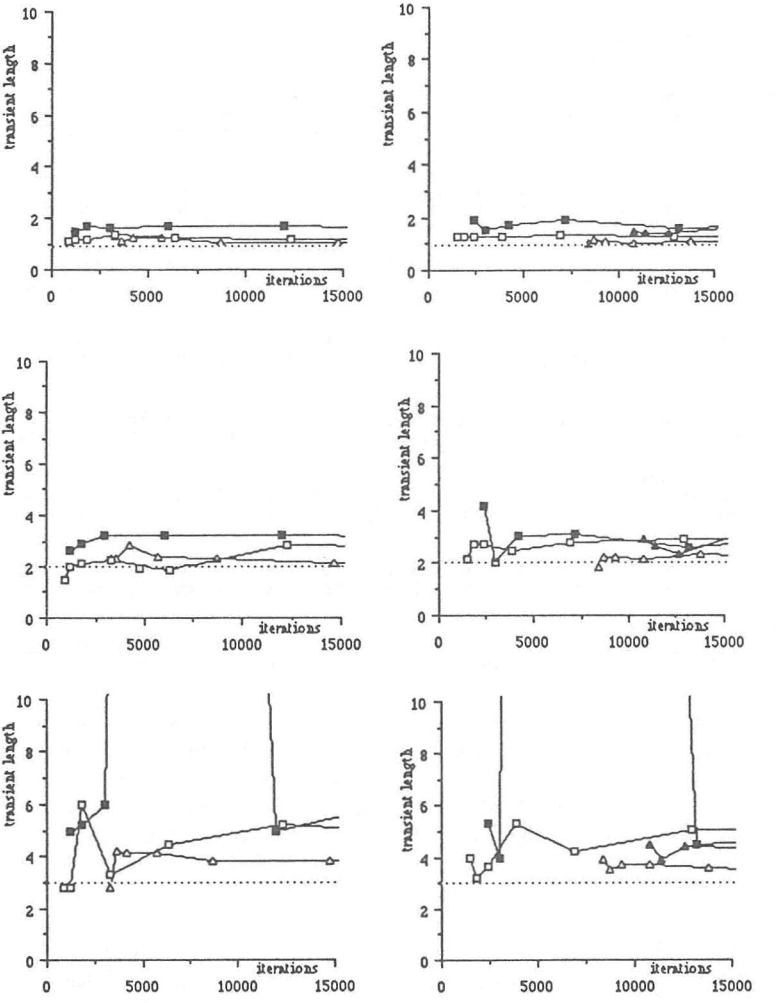

Figure 7: Transient length for the different architectures and different levels of noise: 1 (6.2%) top, 3 (18.7%) middle, and 5 (31.3%) bottom, for $T = 1$ (left) and $T = 1/8$ (right).

N°	error															
1	0.09714															
	-1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	-1.00	1.00	1.00	1.00	1.00	-1.00	1.00	1.00	1.00
	-0.59	0.99	0.95	0.28	-0.45	-0.99	1.00	-0.73	0.95	0.48	1.00	0.93	-0.82	0.95	0.64	0.80
2	0.01593															
	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00	-1.00	1.00	1.00	-1.00
	0.96	-0.66	-0.94	0.92	1.00	-0.89	1.00	-0.83	1.00	-0.82	1.00	0.93	-0.81	0.88	0.98	-1.00
3	0.01438															
	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.79	0.97	0.99	0.71	0.95	0.96	1.00	-0.99	0.85	-0.74	1.00	-0.99	-0.99	-0.96	-0.98	-0.97
4	0.01400															
	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00	1.00	1.00	1.00
	-0.98	1.00	1.00	0.99	-1.00	0.93	0.99	0.87	-0.57	-0.91	0.99	0.99	-0.92	0.97	0.98	1.00
5	0.00198															
	-1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	-1.00	-1.00
	-0.92	-1.00	-1.00	-0.88	1.00	0.95	0.98	-1.00	1.00	0.98	0.98	-0.98	0.95	0.96	-0.96	-1.00
6	0.01378															
	1.00	1.00	-1.00	1.00	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	1.00	1.00	1.00	-1.00
	0.82	0.76	-0.90	0.92	0.80	-0.98	0.99	0.83	0.98	-0.93	0.99	0.99	0.88	1.00	0.92	-0.86
7	0.00450															
	1.00	-1.00	-1.00	1.00	-1.00	1.00	-1.00	-1.00	-1.00	1.00	-1.00	1.00	1.00	-1.00	1.00	-1.00
	0.99	-0.99	-0.94	1.00	-0.85	0.88	-0.98	-0.98	-0.99	0.92	-0.97	0.98	0.86	-0.96	0.98	-1.00
8	0.00685															
	1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	-1.00	-1.00	1.00	1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.97	-0.92	0.96	0.97	0.91	-0.83	0.98	-1.00	-0.77	0.98	0.98	-0.97	-0.97	-1.00	-0.94	-1.00
9	0.01106															
	1.00	1.00	1.00	-1.00	-1.00	-1.00	1.00	1.00	1.00	-1.00	1.00	1.00	1.00	1.00	-1.00	1.00
	0.97	1.00	1.00	-0.82	-0.92	-0.90	0.99	0.99	0.85	-0.98	0.99	0.96	0.83	0.76	-0.88	0.95
10	0.04605															
	1.00	1.00	1.00	-1.00	1.00	-1.00	1.00	-1.00	1.00	1.00	1.00	1.00	-1.00	-1.00	1.00	-1.00
	0.73	0.80	0.88	-0.44	0.92	-0.98	1.00	-0.99	1.00	0.80	1.00	0.79	-0.99	-0.59	0.88	-0.86

Figure 8: Fixed points obtained when starting from pure patterns ($T = 1$, one hidden layer). The figure shows the ten memorized patterns (first line) with the corresponding fixed point (second line) and the error, i.e., the distance from the fixed point to the memorized pattern. The sigmoid was allowed to vary between -1 and 1.

slightly different in value, but not in sign. Gradient backpropagation thus does not ensure that the memorized patterns be fixed points of the dynamics, but it enforces fixed points, close to these patterns and lying in the same region of the space. This "restructuration" of the basins of attractions must certainly help for better generalization, i.e., noise reduction.

The simulations presented in this section thus show that our theoretical results apply to the case of multilayer networks trained through gradient backpropagation. It thus means that iterative dynamics leads to improved generalization performances and one should expect for fixed points (usually close to the memorized patterns, at least in the binary case tested here).

6. Conclusion

We have presented in this paper theoretical results which show that a Lyapunov function can be used to describe the dynamics of neural networks with continuous state and discrete time. This Lyapunov function allows to show that the dynamics on such networks can only lead to fixed points or limit cycles of length two (in the parallel iteration case), provided the connection matrix be symmetric. The transient has been bounded by making use of the Lyapunov function.

The theoretical results have been tested on multilayer networks trained by the gradient backpropagation algorithm. Departure from the symmetry condition was sufficiently low to allow for good concordance between theory and simulation results: when using such networks, one should allow for re-iteration in order to improve performances in a significant fashion.

This feature has not been used yet in the literature; it remains to be tested on real-size applications.

Acknowledgments

This work has been partially supported by PRC-GRECO Intelligence Artificielle (F.F.S.), Fondo Nacional de Ciencias FNC-88-Chile (E.G.), and FNC-88-Chile S.M.), DIB Universidad de Chile (E.G. and S.M.), and Service de Coopération Technique, Ambassade de France (F.F.S., E.G., and S.M.).

References

[1] B. Angéniol, unpublished manuscript (1987).

[2] A.G. Barto, "Game-theoretic cooperativity in networks of self-interested units," In *Neural networks for computing*, Snowbird 1986, J.S. Denker, ed., American Institute of Physics, Conf. Proc. No. 151 (1986) 41–46.

[3] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biological Cybernetics*, **59** (1988).

[4] M.A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. on Systems, Man and Cybernetics*, **13** (1983) 815–826.

[5] F. Fogelman Soulié, P. Gallinari, Y. Le Cun, and S. Thiria, "Automata networks and artificial intelligence," In *Automata Networks in Computer Science, Theory and Applications*, F. Fogelman Soulié, Y. Robert, M. Tchuente, eds. (Manchester Univ. Press, Princeton Univ. Press, 1987) 133–186.

[6] F. Fogelman Soulié, P. Gallinari, Y. Le Cun, and S. Thiria, "Evaluation of network architectures on test learning tasks," *IEEE 1st Intern. Conf. on Neural Networks*, San Diego 1987, Vol. II (1987) 653–660.

[7] F. Fogelman Soulié, P. Gallinari, Y. Le Cun, and S. Thiria, "Network learning," In *Machine Learning*, Vol. 3, Y. Kodratoff, R. Michalski, eds. (Morgan Kaufmann, to appear).

[8] F. Fogelman Soulié and E. Goles, "Knowledge representation by automata networks," In *Computers and Computing* P. Chenin, C. di Crescenzo, F. Robert, eds. (Masson-Wiley, 1986) 175–180.

[9] F. Fogelman Soulié,E. Goles, and G. Weisbuch, "Transient length in sequential iterations of threshold functions," *Disc. Appl. Math.*, **6** (1983) 95–98.

[10] P. Gallinari, S. Thiria, and F. Fogelman Soulié, "Multilayer perceptrons and data analysis," *IEEE 2nd annual International Conference on Neural Networks*, San Diego, 1988, Vol. I (1988) 391–401.

[11] R.M. Golden, "The "Brain-state-in-a-box" neural model is a gradient descent algorithm," *J. Math. Psychol.*, **30(1)** (1986) 73–80.

[12] E. Goles, *Comportement Dynamique de Réseaux d'Automates*, Thèse d'Etat, Grenoble (1985).

[13] E. Goles, "Dynamics of positive automata networks," *Theor. Comp. Sci.*, **41** (1985) 19–32.

[14] E. Goles, E.Chacc, F. Fogelman Soulié, and D. Pellegrin, "Decreasing energy functions as a tool for studying threshold functions," *Disc. Appl. Math.*, **12** (1985) 261–277.

[15] E. Goles and S. Martinez, "Properties on positive functions and the dynamics of associated automata networks," *Disc. Appl. Math.*, to appear.

[16] E. Goles and S. Martinez, "A short proof on the cyclic behaviour of multithreshold symmetric automata," *Information and Control*, **51(2)** (1981) 95–97.

[17] E. Goles and A.M. Odlyzko, "Decreasing energy functions and lengths of transients for some cellular automata," *Complex Systems*, submitted.

[18] E. Goles and G.Y. Vichniac, "Lyapunov functions for parallel neural networks," In *Neural Networks for Computing*, Snowbird 1986, J.S. Denker, ed., Am. Inst. of Physics, Conf. Proc., **151** (1986) 165–181.

[19] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, **79** (1982) 2554–2558.

[20] J.J. Hopfield and D.W. Tank, "Collective computation with continuous variables," In *Disordered Systems and Biological Organization*, NATO workshop Les Houches 1985, E. Bienenstock, F. Fogelman Soulié, and G. Weisbuch, eds., NATO ASI Series in Systems and Computer Science, **F20** (Springer Verlag, 1986) 155–170.

[21] T. Kohonen, "Self-organization and associative memory," *Springer series in Information sciences*, Vol. 8 (Springer Verlag, 1984).

[22] Y. Le Cun, "Learning process in an assymetric threshold network," In *Disordered Systems and Biological Organization*, E. Bienenstock, F. Fogelman Soulié, and G. Weisbuch, Eds., NATO ASI Series in Computer and Systems Sciences, **F20** (Springer Verlag, 1986) 233–240.

[23] Y. Le Cun, *Modèles Connexionnistes de l'Apprentissage*, Thèse, Paris (1987).

[24] W.S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, **5** (1943) 115–133.

[25] M. Minsky and S. Papert, *Perceptrons*, expanded edition (MIT Press, 1988).

[26] F.J. Pineda, "Generalization of backpropagation to recurrent neural networks," *Phys. Rev. Lett.*, **59(19)** (1987) 2229–2232.

[27] F.J. Pineda, "Dynamics and architecture in neural computation," *Journal of Complexity*, to appear.

[28] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning internal representations by error propagation," In *Parallel and Distributed Processing: Explorations in the Micro-structure of Cognition*, D.E. Rumelhart and J.L. McClelland, eds., Vol. 1 (MIT Press, 1986) 318–362.

[29] B. Widrow and M.E. Hoff, "Adaptive switching circuits," *IRE Wescon Conv. Record*, Part 4 (1960) 96–104.

[30] R.J. Williams, *Reinforcement Learning in Connectionist Networks: A Mathematical Analysis*, Institute for Cognitive Science, UCSD, ICS Report 8605 (1986).

[31] D. Zipser, "Programming neural nets to do spatial computations," In *Advances in Cognitive Science*, N.E. Sharkey, ed., Vol. 2, Chichester, Ellis Horwood, to appear.