The Problem of Learning Long-Term Dependencies in Recurrent Networks

Yoshua Bengio†, Paolo Frasconi‡, and Patrice Simard†
†AT&T Bell Laboratories
‡Dip. di Sistemi e Informatica, Universitá di Firenze

***Abstract*— We are interested in training recurrent neural networks to map input sequences to output sequences, for applications in sequence recognition or production. In this paper, we present results showing that learning long-term dependencies in such recurrent networks using gradient descent is a very difficult task. We show how this difficulty arises when robustly latching bits of information with certain attractors: the derivatives of the output at time t with respect the unit activations at time 0 tends rapidly to 0 as t increases for most input values. In such a situation, simple gradient descent techniques appear inappropriate and we suggest to consider alternative optimization methods and architectures.**

I. Introduction

A recurrent network has *cycles* in its graph that allow it to keep information about past inputs for an amount of time that is not fixed a-priori, but rather depends on its weights and on the input data. In contrast, static networks, even if they include delays (e.g., as in Time-Delay Neural Networks), have a finite impulse response and can't store a bit of information for an indefinite amount of time. The type of recurrent networks studied here can be used either for sequence recognition or production: units are not usually clamped. Instead, the dynamics of the network are non-autonomous. Hence the recurrent network transforms an input sequence (e.g., speech spectra) into an output sequence (e.g., degrees of evidence for phonemes), while taking into account contextual information in a flexible way. In this paper, we restrict our attention to discrete-time dynamical systems.

Although recurrent networks can in many instances outperform static networks [1], they appear more difficult to train optimally. Earlier experiments indicated that their parameters settle in sub-optimal solutions which take into account short-term dependencies but not long-term dependencies [2]. Similar results were obtained by Mozer [7]. It was found that back-propagation was not sufficiently powerful to discover contingencies spanning long temporal intervals. In this paper, we present experimental and theoretical results in order to further the understanding of this problem, i.e., of the conditions in which gradient descent in the output error is inappropriate to train a non-autonomous recurrent network. We begin in Section ii, by presenting a minimal problem for such recurrent networks. Being able to learn this task is a necessary condition for any recurrent network and learning algorithm applied to sequence recognition problems involving long-term context. It requires

1. a system that is able to store one or more bit of information for an arbitrary duration,
2. that is resistant to background noise,
3. and that is learnable[1].

In Section iii, theoretical results are presented showing that when trying to satisfy conditions (1) and (2) above, the magnitude of the derivative of the state of a dynamical system at time t with respect to the state at time 0 decreases exponentially as t increases. We show how this makes the back-propagation algorithm (and gradient descent in general) inefficient for learning of long term dependencies in the input/output sequence, hence failing condition (3) for sufficiently long sequences.

II. A minimal Problem for Recurrent Network

This minimal task is designed as a test that must necessarily be passed in order to satisfy the three conditions enumerated above. For this task, a dynamic system is to learn discriminating between two different sets of sequences, whose class is only determined by the values of the input on a fixed number L of time steps at the beginning of each sequence of length T. If we allow sequences of arbitrary length, then the problem can be solved only if the network is able to latch information about the initial input values. The input u_t for this system can be thought of as the weighted sum of the outputs of a subnetwork directly interfaced with the actual inputs. Hence we can regard u_t as values which can be tuned by gradient

[1]Given external targets, it can compute variations of its inputs and parameters necessary to approach those targets.

0-7803-0999-5/93/$03.00 ©1993 IEEE

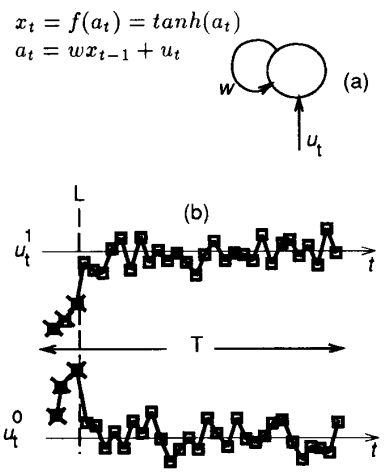

Figure 1: a) Latching neuron. b) Sample input sequences after successful learning (free parameters are marked with small crosses).

descent. The problem is thus stated as follows. The system has one input u_t and one output x_t (at each discrete time step t). The initial inputs u_t^k $t \leq L$ are learnable parameters whereas u_t is zero mean gaussian noise for $t > L$. $k = 0$ or $k = 1$ for the two classes (0 and 1) of sequences. Optimization is based on the cost function $C = (x_T^0 + \overline{x})^2 + (x_T^1 - \overline{x})^2$ with $\overline{x}$ a target close to 1 for the last time step T. Although very simple, this example can provide insights on the behavior of more complex networks. The ability of learning the free input values is a measure of the effectiveness of the gradient which would be propagated further back if the neuron were connected to the output of another subnet.

We performed experiments on this task with a single recurrent neuron, as shown in fig. 1a. This recurrent neuron can latch information if $w > 1/f'(0)$ [3],[4]. Two types of trajectories are considered for this system, for the two classes ($k = 0$, $k = 1$):

$$\begin{aligned} x_t^k &= \tanh(a_t^k) \\ a_t^k &= w \tanh(a_{t-1}^k) + u_t^k \qquad t = 1 \ldots T \\ a_0^0 &= a_0^1 = 0 \end{aligned} \tag{1}$$

The recurrent weight w is also learnable. The solution for $T \gg L$ requires to adapt $w > 1$ to produce two stable attractors $\overline{x}$ and $-\overline{x}$. The free input parameters must yield the state of the neuron towards $\overline{x}$ or $-\overline{x}$ in order to robustly latch a bit of information against the input noise. Note that a linear network would not be resistant to noise in such a situation.

In fig. 1b we show two sample input sequences. A set of simulations were carried out to evaluate the effectiveness

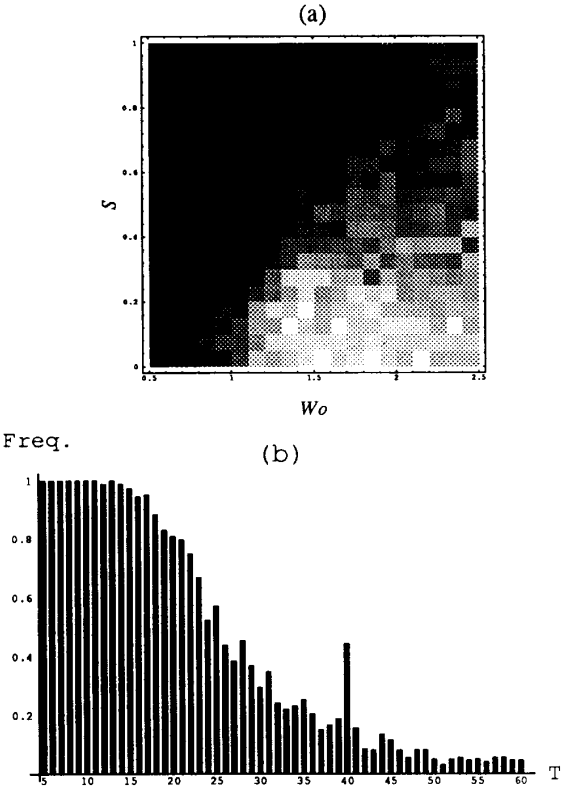

Figure 2: Experimental results for the illustrating problem. a) Density of convergence with respect to the initial weight w_0 and the noise variance s (white $\Rightarrow$ high density), with $L = 4$ and $T = 20$. b) Frequency of convergence with respect to the sequence length T, ($s = 0.2, w_0 = 1.25$).

of back-propagation on this simple task. In a first experiment we investigated the effect of the noise variance s and of different initial values w_0 for the self loop weight. A density plot of convergence is shown in fig. 2a, averaged over 18 runs for each of the selected pairs (w_0, s). It can be seen that convergence becomes very unlikely for large noise variance or small initial values of w. $L = 3$ and $T = 20$ were chosen in these experiments. In fig. 2b, we show instead the effect of varying T, keeping fixed $s = 0.2$ and $w_0 = 1.25$. In this case the task consists in learning only the input parameters u_t. When T becomes large it is extremely difficult to attain convergence. These experimental results show that even in the very simple situation where we want to robustly latch on 1 bit of information about the input, gradient descent on the output error fails for long term input/output dependency, for most initial parameter values.

III. Gradient Loss: Analysis

In this section, we attempt to understand better why the gradient with respect to the weights often ignores the contributions due to long term dependencies in the input/output sequences. Let us consider the non-autonomous discrete-time map with additive inputs

$$a_t = M(a_{t-1}) + u_t \tag{2}$$

and the corresponding autonomous dynamics

$$a_t = M(a_{t-1}) \tag{3}$$

where a_t and u_t are n-vectors representing respectively the system state and the external input[2] at time t. In the next subsection, we find that the problem mentionned in the previous section occurs when $\frac{\partial x_t}{\partial x_\tau} \to 0$ as $t - \tau$ increases, with $\tau < t$. Furthermore, we argue that this is a common situation, when the network is used to "robustly latch" on bits of information, i.e., when its dynamics are restricted to one of several basins of attraction while being resistant to input noise.

A. Latching in a basin of attraction

Here we will argue that in certain conditions of network dynamics that we will define as "robust latching", the derivatives of a_t with respect to a_0 tend to decrease rapidly as t increases, for most initial conditions a_0 and for inputs that do not disturb the latching condition. This situation is the essential reason for the difficulty in using gradient descent to train a dynamical system to model long-term dependencies.

Definition 1 *A set of points E is said to be* invariant *under a map M if $E = M(E)$.*

Definition 2 *A* hyperbolic attractor *is a set of points X invariant under the differentiable map M, such that $\forall a \in X$, all eigenvalues of $M'(a)$ are less than 1 in absolute value.*

An attractor X may contain a single point (fixed point attractor), a finite number of points (periodic attractor), or an infinite number of points (chaotic attractor). Note that a stable and attracting fixed point is hyperbolic for the map M, whereas a stable and attracting periodic attractor of period l for the map M is hyperbolic for the map M^l. For a recurrent net, the kind of attractor depends on the weight matrix. In particular, for a network defined by $a_t = W \tanh(a_{t-1}) + u_t$, if W is symmetric and its minimum eigenvalue is greater than -1, then the attractors are all fixed points [6]. On the other hand, if $||W|| < 1$, the system has a single fixed point attractor at the origin.

Definition 3 *The* basin of attraction *of an attractor X is the set $\beta(X)$ of points a* converging *to X under the map M, i.e., $\beta(X) = \{a : \forall \epsilon, \exists l, \exists x \in X \text{ s.t. } ||M^l(a) - x|| < \epsilon\}$.*

Definition 4 *We call $\Gamma(X)$ the* reduced attracting set *of a hyperbolic attractor X the set of points y in the basin of attraction of X, such that $\forall l \geq 1$, all the eigenvalues of $(M^l)'(y)$ are less than 1.*

Lemma 1 *For a hyperbolic attractor X, $X \subset \Gamma(X) \subset \beta(X)$.*

Definition 5 *A network is* robustly latched *at time t_0 to X, one of several hyperbolic attractors, if a_{t_0} is in the reduced attracting set of X under a map M defining the autonomous network dynamics.*

Again, for the case of non-autonomous dynamics, it remains robustly latched to X as long as the inputs u_t are such that $a_t \in \Gamma(X)$ for $t > t_0$. Let us now see why it is more robust to store a bit of information by keeping a_t in $\Gamma(X)$.

Theorem 1 *Assume x is a point of $\mathbf{R}^n$ such that there exist an open sphere $U(x)$ centered on x for which $||M'(z)|| > 1$ for all $z \in U(x)$. Then there exist $y \in U(x)$ such that $||M(x) - M(y)|| > ||x - y||$.*

Proof: By hypothesis and definition of norm, $\exists\ u$ s.t. $||u|| = 1$ and $||M'(x)u|| > 1$. The Taylor expansion of M at x for small value of λ is:

$$M(x + \lambda u) = M(x) + M'(x)\lambda u + O(||\lambda u||^2) \tag{4}$$

Since U(x) is an open set, $\exists\ \lambda$ s.t. $||O(||\lambda u||^2)|| < \lambda(||M'(x)u|| - 1)$ and $x + \lambda u \in U(x)$. Letting $y = x + \lambda u$ we can write $||M(y) - M(x) - M'(x)\lambda u|| = ||O(||\lambda u||^2)|| < \lambda||M'(x)u|| - \lambda$ or $-||M(y) - M(x) - M'(x)\lambda u|| + ||M'(x)\lambda u|| > \lambda$. This implies using the triangle inequality $||M(y) - M(x)|| > \lambda = ||x - y||$. □

This theorem implies that for a hyperbolic attractor X, if a_0 is in $\beta(X)$ but not in $\Gamma(X)$, then the size of a ball of uncertainty around a_0 will grow exponentially as t increases. Thus the system would not be resistant to input noise. In contrast, the following results show that if a_0 is in $\Gamma(X)$, it is guaranteed to remain within a certain distance of X when the input is bounded.

Definition 6 *A map M is* contracting *on a set D if $\exists \alpha \in [0, 1)$ such that $||M(x) - M(y)|| \leq \alpha||x - y||\ \ \forall x, y \in D$.*

[2] A dynamic system with non-additive inputs, e.g., $a_t = N(a_{t-1}, u_{t-1})$, can be transformed into one with additive inputs by introducing n additional state variables and corresponding inputs. The $2n$-vector state is $x_t = (a_t, y_t)$ where $a_t = N(a_{t-1}, y_{t-1})$ (i.e., with 0 inputs) and $y_t = u_t$ (i.e., with the 0 map).

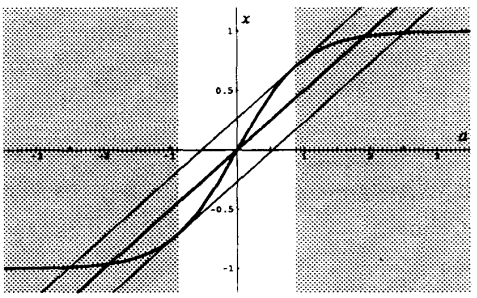

Figure 3: Reduced attracting set (shaded area) in the monodimensional case, for $w = 2$. The heavy line represents the equation $a = wx$. The two light lines represent the equations $a = wx \pm u^*$, where $u^* = 0.532$ is the maximum absolute value of the input to guarantee information latching (cf. [3],[4]).

Theorem 2 *Let M be a differentiable mapping on a convex set D. If $\forall x \in D$, $|M'(x)| < 1$, then M is contracting on D.*

Proof: See [8].

Theorem 3 *Suppose a network is robustly latched to X, starting in state a_0, and the inputs u_t are such that for all $t > 0$, $\|u_t\| < b$ and $\forall y \in D_t$, $|M'(y)| < \lambda < 1$, where D_t is a ball of radius b around a_t. Then for $t > 0$, a_t must be in a ball of radius of less than $\frac{b}{1-\lambda}$. Furthermore, for $t \to \infty$, this ball intersects X.*

Proof: Let us denote by ρ_t the radius of the "uncertainty" ball $\Phi_t = \{a : \|\tilde{a}_t - a\| < \rho_t\}$ in which we are sure to find a_t, where $\tilde{a}_t$ gives the trajectory of the autonomous system. By Lagrange's mean value theorem and convexity of Φ_t, $\exists z \in \Phi_t$ s.t. $\|M(x) - M(y)\| \leq |M'(z)|\|x - y\|$, but $|M'(z)| < \lambda$ by hypothesis. Then by the contraction theorem (2), we have $\rho_0 = 0$, $\rho_1 = b$, $\rho_2 = \lambda b + b$, $\rho_t = \lambda \rho_{t-1} + b$. Hence for $t > 0$, $\rho_t = b \sum_{i=0}^{t-1} \lambda^i = \frac{b(1-\lambda^t)}{1-\lambda}$, thus $\lim_{t\to\infty} \rho_t = \frac{b}{1-\lambda}$. To show how Φ_t intersects with X when $t \to \infty$, consider a point c_t in Φ_t obtained from a point c_{t-1} in Φ_{t-1} by setting $c_t = M(c_{t-1})$. c_t must converge to X, hence the conclusion. □

The above results justifies the term "robust" in our definition of robustly latched network: as long as u_t is such that a_t remains in the reduced attracting set of a hyperbolic attractor X, a_t is guaranteed to remain within a distance $\frac{2b}{1-\lambda}$ of some point in X. On the other hand, outside $\Gamma(X)$ but in $\beta(X)$, M is not contracting, it is expanding, i.e., the size of a ball of uncertainty grows exponentially with time.

Theorem 4 *If the input u_t is such that a network remains robustly latched on attractor X after time 0, then $\frac{\partial a_t}{\partial a_0} \to 0$ as $t \to \infty$.*

Proof: By hypothesis and definitions 4 and 2, $\left\|\frac{\partial a_\tau}{\partial a_{\tau-1}}\right\| = |M'(a_{\tau-1})| < 1$ for $\tau > 0$, hence $\frac{\partial a_t}{\partial a_0} \to 0$ as $t \to \infty$ □.

The above results show that when storing one or more bit of information in a way that is resistant to noise, the gradient with respect to past events rapidly becomes very small in comparison to the gradient with respect to recent events. In the next section we discuss how that makes gradient descent on parameter space (e.g., the weights of a network) inefficient.

B. Effect on the Weight Gradient

According to the above results, in state-space regions with strong attractors (with basins of attraction), the behavior of the network resembles that of a multi-valued analog switch (or latch). The input can push the state into one of several basins of attraction. Once in a basin of attraction, it stays there if or until a sequence of inputs pushes it out. If the network is robustly latched on one of these attractors, the derivatives of the cost function at time t with respect to the state at an earlier time rapidly tend to 0 as t increases.

Let us consider the effects of these conditions on the derivatives of a cost C_t at time t with respect to parameters of a dynamical system, say a neural network with weights W:

$$\frac{\partial C_t}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} = \sum_{\tau \leq t} \frac{\partial C_t}{\partial a_t} \frac{\partial a_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W} \qquad (5)$$

Suppose we are in the condition in which the network has robustly latched. Hence for a term with $\tau \ll t$, $\left\|\frac{\partial C_t}{\partial a_\tau} \frac{\partial a_\tau}{\partial W}\right\| \to 0$. This term tends to become very small in comparison to terms for which τ is close to t. This means that even though there might exist a change in W that would allow a_τ to jump to another basin of attraction, this effect is not taken into account. This is because the effect of a **small** change in W would be felt mostly on the near past (τ close to t). For example, if between the time the system is latched (say time 0) and the time the stored bit is used (say time t) the inputs are noisy, the gradient contribution for time 0 tends to be insignificant in comparison to the contribution for intervening times. Worse, if the intervening events are not noise but represent other input events that can influence later outputs, the system can learn these short-term dependencies but will probably fail to learn the long-term dependencies.

C. Generalization to a Projection of the State

The results in the above subsections can be generalized to the case when a projection Pa_t of the state a_t converges to an attractor under a map M. Let P and R be orthogonal

projection matrices such that

$$\begin{aligned} a_t &= P^+ z_t \; + \; R^+ y_t \\ z_t &= P a_t; \quad y_t = R a_t \\ P R^+ &= 0; \quad R P^+ = 0 \end{aligned} \tag{6}$$

where A^+ denotes the right pseudo-inverse of A, i.e. $AA^+ = I$. Suppose M is such that P can be chosen so that z_t converges to an attractor Z with the dynamics $z_t = M_P(z_{t-1}) = P_t M(P^+ z_t + R^+ y_t)$ for any y_t. Then we can specialize all the previous definitions, lemmas and theorems to the subspace z_t. When we conclude with these results that $\frac{\partial z_t}{\partial z_0} \to 0$, we can infer that $\frac{\partial a_t}{\partial a_0} \to R^+ \frac{\partial y_t}{\partial y_0} R$, i.e., that the derivatives of a_t with respect to a_0 depend only on the projection of a on the subspace Ra. Hence the influence of changes in the projection of a on the subspace Pa is ignored in the computation of the gradient with respect to W, even though non-infinitesimal changes in Pa could yield very different results (jumping into a different basin of attraction).

D. Alternative Approaches

This section helped us understand better why training a recurrent network to learn *long range* input/output dependencies is a hard problem. Gradient-based methods appear inadequate for this kind of problem. We need to consider alternative systems and optimization methods that give acceptable results even when the criterion function is not smooth and has long plateaus. Global search methods such as simulated annealing can be applied to such problems, but they are generally very slow. One way to help in the training of recurrent networks is to set their connectivity and initial weights (and even constraints on the weights) using prior knowledge. For example, this is accomplished in [4] and [5] using prior rules and sequentiality constraints. Another approach is to look for algorithms that can yield desired changes in parameters even when the map is non-differentiable. Such an approach is explored in the next section.

IV. A Trainable Flip-Flop

The goal of this section is to propose a new kind of unit designed to meet the 3 conditions listed in the introduction, at least for the minimal problem presented in Section II. The idea is to have a unit whose function is not necessarily differentiable but for which condition (3) (learnability) is still satisfied. Since no gradient exists for such a unit, the variations that should be submitted to the input to induce the desired variation of the output should be computed discretely. In the simplest case, the unit has one internal state and one input. Its time behavior is described by:

$$x_{t+1} = f(x_t, u_t) \tag{7}$$

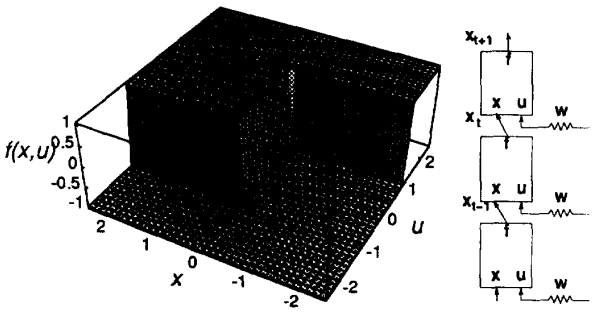

Figure 4: Left: Map $f(x, u)$ for the trainable flip-flop. Right: unfolding in time of the trainable flip-flop with its inputs x_t and u_t and its output $x_{t+1} = f(x_t, u_t)$.

A simple function f which exhibit noise resistance and can store one bit of information is shown in figure 4 and defined as follows:

$$f(x,u) = \begin{cases} 1 & \text{if } |u| < 1 \text{ and } x \geq 0 \\ & \text{or if } u \geq 1 \\ -1 & \text{otherwise} \end{cases} \tag{8}$$

This unit satisfies criterion (1) and (2) of the introduction. For learning (criterion (3)), we have to compute the variation of Δf with respect to Δx and Δu. The variation of f with respect to the variation of x can be easily infered from Eq. 8 or from figure 4 :

$$\Delta f_x(\Delta x, x, u) = \begin{cases} 2 & \text{if } |u| < 1 \ , \ x < 0 \ , \ x + \Delta x \geq 0 \\ -2 & \text{if } |u| < 1 \ , \ x \geq 0 \ , \ x + \Delta x < 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

From which we can infer a "backpropagation" rule in time:

$$\Delta x(\Delta f) = \begin{cases} k_{x,1} - x & if \ \ |u| < 1 \ , \ x < 0 \ , \ \Delta f = 2 \\ -k_{x,2} - x & if \ \ |u| < 1 \ , \ x \geq 0 \ , \ \Delta f = -2 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Where $k_{x,1}$ and $k_{x,2}$ are positive constants. Assuming the additional constraint $x \in \{-1, +1\}$, this system simplifies to:

$$\Delta x(\Delta f, u) = \begin{cases} \Delta f & \text{if } |u| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Note that this is a recursive rule for backpropagating *targets* in time, $\Delta x_t = \Delta x_{t+1}$ if $|u_t| < 1$ and 0 otherwise. In other words, the Δf signal is passed down to earlier times for as long as the flip-flop was in the final state. Similarily, the variation of f with respect to the variation of u can be easily infered from Eq. 8 or from figure 4 :

$$\Delta f_u(\Delta u, x, u) =$$

$$\begin{cases} 2 & \text{if } x<0\ ,\ u<1\ ,\ u+\Delta u>1 \\ & \text{or } x>0\ ,\ u<-1\ ,\ u+\Delta u>-1 \\ -2 & \text{if } x<0\ ,\ u>1\ ,\ u+\Delta u<1 \\ & \text{or if } x>0\ ,\ u>-1\ ,\ u+\Delta u<-1 \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

From this we can compute the variations Δu

$$\Delta u(\Delta f, x, u) = \begin{cases} 1-u+k_1 & \text{when } x<0\ ,\ u<1\ ,\ \Delta f=2 \\ -1-u+k_2 & \text{when } x>0\ ,\ u<-1\ ,\ \Delta f=2 \\ 1-u-k_3 & \text{when } x<0\ ,\ u>1\ ,\ \Delta f=-2 \\ -1-u-k_4 & \text{when } x>0\ ,\ u>-1\ ,\ \Delta f=-2 \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

Where each k_i is a positive constant. When used in a network, the rule becomes recursive, that is, Δu_t is computed from $\Delta x_{t+1}, x_t, u_t$. The learning rates for each Δu_t are weighted by the probability of contribution. Lets assume for the sake of illustration that the flip-flop has ended in state +1 at time T, that the target state was -1 and that $|u_t| < 1$ for $T-n < t \leq T$ and $u_{T-n} > 1$. There are two ways of udating u_t to get the correct answer. Either $u_t < -1$ for some t in $T-n < t \leq T$ or $u_{T-n} < 1$. The learning rate should therefore be weighted by $0.5/n$ for $T-n < t \leq T$ and by 0.5 for $t = T-n$.

In order to adapt the threshold to the external level of noise and signal, the input to the unit is scaled by a weight w that can also be learnt using the target on u. Experiments were performed with this system on the minimal task described in Section II. Sequence lengths were varied from 10 to 100 by increment of 10 and zero-mean uniform noise with levels from 0.2 to 1.0 by increment of 0.2. We found the algorithm to converge in more than 99% of these cases (within a maximum of 100 epochs), independently of the noise level and the sequence length, in average within 4.6 epochs. This is in contrast to the recurrent sigmoid trained with backpropagation which was sensitive to the noise level and for which the probability of convergence decreased rapidly when increasing sequence length (as shown in Figure 2).

V. Conclusion

Recurrent networks are very powerful in their ability to represent context, often outperforming static networks [1]. However, we have presented theoretical and experimental evidence indicating that in many circumstances, simple gradient-based optimization of an error criterion may be inadequate to train them. There remains theoretical questions to be considered, such as whether the problem with simple gradient descent discussed in this paper would be observed with chaotic attractors that are not hyperbolic. A better understanding of the problem underlying the loss of information in the gradient may help us designing more efficient algorithms and/or architectures for recurrent neural networks. In fact, based on the specifications required to solve a minimal problem for such algorithms, we have introduced a new type of unit with its learning algorithm, that could learn to store one bit of information for an indefinite duration while being resistant to input noise. In any case, the results of this paper strongly suggest that different training algorithms should be considered for recurrent networks. Solutions to the challenge presented here to learning long-term dependencies with dynamical systems such as recurrent networks may have implications for many types of applications for learning systems, e.g., in language related problems, for which long-term dependencies and the use of context are essential in order to take correct decisions.

References

[1] Y. Bengio, "Artificial Neural Networks and their Application to Sequence Recognition," Ph.D. Thesis, McGill University, (Computer Science), 1991, Montreal, Qc., Canada.

[2] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, "Global Optimization of a Neural Network - Hidden Markov Model Hybrid," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, 1992, pp. 252–259.

[3] P. Frasconi, M. Gori, and G. Soda, "Local Feedback Multilayered Networks", *Neural Computation* **3**, 1992, pp. 120–130.

[4] P. Frasconi, M. Gori, M. Maggini, and G. Soda, "Unified Integration of Explicit Rules and Learning by Example in Recurrent Networks," *IEEE Trans. on Knowledge and Data Engineering*, in press.

[5] C.L. Giles and C.W. Omlin, "Inserting Rules into Recurrent Neural Networks", *Neural Networks for Signal Processing II, Procedings of the 1992 IEEE workshop*, (eds. Kung, Fallside, Sorenson and Kamm), IEEE Press, pp. 13-22.

[6] C.M. Marcus, F.R. Waugh, and R.M. Westervelt, "Nonlinear Dynamics and Stability of Analog Neural Networks", *Physica D* **51** (special issue), 1991, pp. 234-247.

[7] Mozer M.C., "Induction of multiscale temporal structure", Advances in Neural Information Processing Systems 4, (eds. Moody, Hanson, Lippman), Morgan Kauffmann, 1992, pp. 275-282.

[8] Ortega J.M. and Rheinboldt W.C. *Iterative Solution of Non-linear Equations in Several Variables and Systems of Equations*, Academic Press, New York, 1960.