LEARNING STATE SPACE TRAJECTORIES IN RECURRENT NEURAL NETWORKS: A PRELIMINARY REPORT

Technical Report AIP - 54

Barak A. Pearlmutter

Department of Computer Science
Carnegie Mellon University
Pittsburgh, Pa. 15213

24 July 1988

This research was supported by the Computer Sciences Division, Office of Naval Research and DARPA under Contract Number N00014-86-K-0678. The author is a Fannie and John Hertz Foundationation Fellow.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	3. DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; Distribution unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AIP -- 54	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Computer Sciences Division Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code) Department of Psychology Pittsburgh, Pennsylvania 15213	7b. ADDRESS (City, State, and ZIP Code) 800 N. Quincy Street Arlington, Virginia 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Same as Monitoring Organization 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N00014-86-K-0678
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS p4000ub201/7-4-86

PROGRAM ELEMENT NO	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO
N/A	N/A	N/A	N/A

11. TITLE (Include Security Classification) LEARNING STATE SPACE TRAJECTORIES IN RECURRENT NEURAL NETWORKS: A PRELIMINARY REPORT

12. PERSONAL AUTHOR(S) Barak A. Pearlmutter

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Technical	FROM 86Sept15 TO 91Sept14	1988 July 24	6

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	connectionism, learning algorithm, trajectories following, minimizing functionals

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

We describe a procedure for finding $\partial E/\partial w_{ij}$ were E is an arbitrary functional of th temporal trajectory of the states of a continuous recurrent network and w_{ij} are the weights of that network. An embellishment of this procedure involving only computations that go forward in time is also described. Computing these quantities allows one to perform gradient descent in the weights to minimize E, so our procedure forms the kernel of a new connectionist learning algorithm.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION	
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Alan L. Meyrowitz	22b. TELEPHONE (Include Area Code) (202) 696-4302	22c. OFFICE SYMBOL N00014

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

Unclassified

Learning State Space Trajectories in Recurrent Neural Networks: A Preliminary Report

Barak A. Pearlmutter
Carnegie Mellon University
Pittsburgh, PA 15213

July 24, 1988

Abstract

We describe a procedure for finding $\partial E/\partial w_{ij}$ where E is an arbitrary functional of the temporal trajectory of the states of a continuous recurrent network and w_{ij} are the weights of that network. An embellishment of this procedure involving only computations that go forward in time is also described. Computing these quantities allows one to perform gradient descent in the weights to minimize E, so our procedure forms the kernel of a new connectionist learning algorithm.

1 Introduction

Pineda [2] has shown how to train the fixpoints of a recurrent temporally continuous generalization of backpropagation networks [3]. Such networks are governed by the coupled differential equations

$$T_i \frac{dy_i}{dt} = -y_i + \sigma(x_i) + I_i \tag{1}$$

where

$$x_i = \sum_j w_{ji} y_j$$

is the total input to unit i, y_i is the state of unit i, T_i is the time constant of unit i, σ is an arbitrary differentiable function[1], w_{ij} are the weights, and the boundary conditions $\mathbf{y}(t_0)$ and driving functions $\mathbf{I}$ are the input to the system. See figure 2 for a graphical representation of this equation.

[1] Typically $\sigma(\xi) = (1 + e^{-\xi})^{-1}$, in which case $\sigma'(\xi) = \sigma(\xi)(1 - \sigma(\xi))$.

Consider $E(\mathbf{y})$, an arbitrary functional of the trajectory taken by $\mathbf{y}$ between t_0 and t_1.[2] Below, we develop a technique for computing $\partial E(\mathbf{y})/\partial w_{ij}$ and $\partial E(\mathbf{y})/\partial T_i$, thus allowing us to do gradient descent in the weights and time constants so as to minimize E. The computation of $\partial E/\partial w_{ij}$ seems to require a phase in which the network is run backwards in time, but a trick for avoiding this is also developed.

2 The Equations

Let us define

$$e_i(t) = \lim_{\epsilon \to 0} \epsilon^{-1} \frac{\delta E(\mathbf{y})}{\delta y_i[t..t+\epsilon]}. \tag{2}$$

In the usual case where E is of the form $E(\mathbf{y}) = \int_{t_0}^{t_1} f(\mathbf{y}(t), t)\, dt$ this means that $e_i(t) = \partial f(\mathbf{y}(t), t) / \partial y_i(t)$. Intuitively, $e_i(t)$ measures how much a small change to y_i at time t effects E if everything else is left unchanged. We also define

$$z_i(t) = \frac{\partial E(\hat{\mathbf{y}}^{(t,i,\xi)})}{\partial \xi} \text{ at } \xi = 0 \tag{3}$$

where $\hat{\mathbf{y}}^{(t,i,\xi)}$ is the same as $\mathbf{y}$ except that $d\hat{y}_i/dt$ has a Dirac delta function of magnitude ξ added to it at time t. Intuitively, $z_i(t)$ measures how much a small change to y_i at time t effects E when the change to y_i is propagated forward through time and influences the remainder of the trajectory.

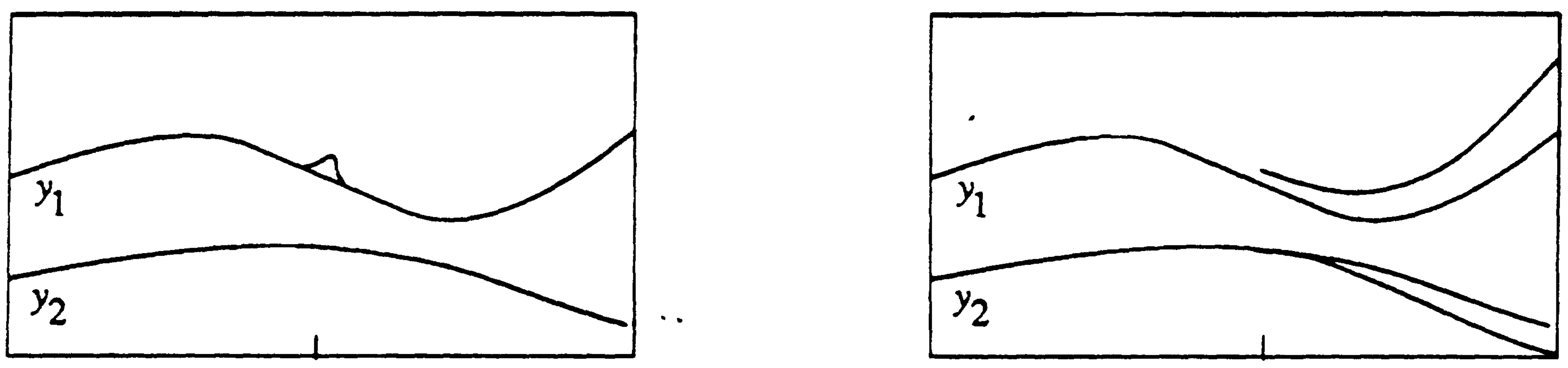

Figure 1: The infinitesimal changes to $\mathbf{y}$ considered in $e_1(t)$ (left) and $z_1(t)$ (right).

We can approximate (1) with the difference equation

$$y_i(t + \Delta t) \approx y_i(t) + \Delta t \frac{dy_i}{dt}(t)$$

or

$$y_i(t + \Delta t) \approx \left(1 - \frac{\Delta t}{T_i}\right) y_i(t) + \frac{\Delta t}{T_i} \sigma(x_i(t)) + \frac{\Delta t}{T_i} I_i(t) \tag{4}$$

which is exact in the limit as $\Delta t \to 0$.

[2]For instance, $E = \int_{t_0}^{t_1} (y_0(t) - f(t))^2 dt$ measures the deviation of y_0 from the funtion f, and minimizing this E would teach the network to have y_0 imitate f.

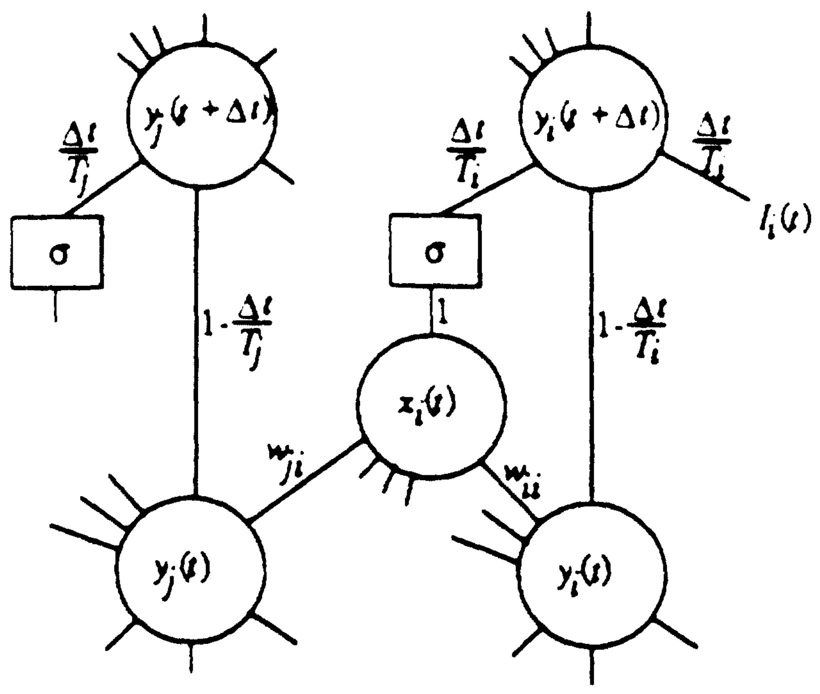

Figure 2: A lattice representation of (4).

Consider incrementing $y_i(t)$ by ϵ and letting this change propagate forward. The differential of $E(\mathbf{y})$ w.r.t. ϵ is thus the sum of the differentials of $E(\mathbf{y})$ w.r.t. the other values that $y_i(t)$ influences, weighted by the magnitude of its influence. By examining all the outgoing lines from node $y_i(t)$ in figure 2 we are led to a difference equation for $z_i(t)$,

$$z_i(t) \approx \left(1 - \frac{\Delta t}{T_i}\right) z_i(t+\Delta t) + \Delta t\, e_i(t) + \sum_j \frac{\Delta t}{T_j} w_{ij} \sigma'(x_j(t)) z_j(t+\Delta t), \tag{5}$$

where the $(1 - \Delta t/T_i)z_i(t)$ term is due to the linear influence $y_i(t)$ has upon $y_i(t+\Delta t)$, the $\sum_j$ term is due to the effect that changing $y_i(t)$ has upon the other $y_j(t+\Delta t)$ through their nonlinear coupling, and the $\Delta t\, e_i(t)$ term is due to the effect that changing y_i between times t and $t+\Delta t$ has directly upon E. By rewriting (5) as

$$z_i(t) \approx z_i(t+\Delta t) - \Delta t \left(\frac{1}{T_i} z_i(t+\Delta t) - e_i(t) - \sum_j \frac{1}{T_j} w_{ij} \sigma'(x_j(t)) z_j(t+\Delta t) \right),$$

assuming this to be of the form $z_i(t) = z_i(t+\Delta t) - \Delta t\, dz_i/dt\,(t+\Delta t)$, and taking the limit as $\Delta t \to 0$ we obtain a differential equation,

$$\frac{dz_i}{dt} = \frac{1}{T_i} z_i - e_i - \sum_j \frac{1}{T_j} w_{ij} \sigma'(x_j) z_j. \tag{6}$$

Let

$$v_{ij}(t) = \frac{\partial E(\tilde{\mathbf{y}}^{(i,j,\xi,t)})}{\partial \xi} \text{ at } \xi = 0 \tag{7}$$

where $\tilde{\mathbf{y}}^{(i,j,\xi,t)}$ is the same as $\mathbf{y}$ except that w_{ij} is increased by ξ from t through t_1. Again examining figure 2, we see that the appropriate difference equation for v is

$$v_{ij}(t) = v_{ij}(t+\Delta t) + \Delta t\, y_i(t) \sigma'(x_j(t)) \frac{1}{T_j} z_j(t+\Delta t)$$

which leads to the differential equation

$$\frac{dv_{ij}}{dt} = -\frac{1}{T_j} y_i \sigma'(x_j) z_j$$

which we can integrate from t_0 to t_1. By substituting $v_{ij}(t_1) = 0$ and $v_{ij}(t_0) = \partial E / \partial w_{ij}$ into the resulting equation we eliminate v and end up with

$$\frac{\partial E}{\partial w_{ij}} = \frac{1}{T_j} \int_{t_0}^{t_1} y_i \sigma'(x_j) z_j dt. \tag{8}$$

If we substitute $\rho_i = T_i^{-1}$ into (4), find $\partial E / \partial \rho_i$ by proceeding analogously, and substitute T_i back in we get

$$\frac{\partial E}{\partial T_i} = -T_i^{-1} \int_{t_0}^{t_1} z_i \frac{dy_i}{dt} dt. \tag{9}$$

We will find a way to compute $\partial z_i(t_1) / \partial z_j(t_0)$ useful. Let us define

$$\zeta_{ij}(t) = \frac{\partial z_i(t)}{\partial z_j(t_0)} \tag{10}$$

and take the partial of (6) with respect to $z_j(t_0)$, substituting in ζ_{ij} where appropriate. This results in a differential equation for ζ_{ij},

$$\frac{d\zeta_{ij}}{dt} = \frac{1}{T_i}\zeta_{ij} - \sum_k \frac{1}{T_k} w_{ik} \sigma'(x_k) \zeta_{kj}, \tag{11}$$

and for boundary conditions we note that

$$\zeta_{ij}(t_0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

One can also derive (6), (8) and (9) using the calculus of variations and Lagrange multipliers (Dr. William Skaggs, personal communication).

3 Utilization

The most straightforward way to use (6), (8) and (9) is to simulate the system $\mathbf{y}$ forward from t_0 to t_1, set the boundary conditions $z_i(t_1) = 0$, and simulate the system $\mathbf{z}$ backwards from t_1 to t_0 while numerically integrating $z_j \sigma'(x_j) y_i$ and $z_i \, dy_i/dt$ thus computing $\partial E / \partial w_{ij}$ and $\partial E / \partial T_i$. Aside from the practical problems of simulating the system backwards in an actual learning application, the backwards simulation of $\mathbf{z}$ as well as the integrals being computed require that $\mathbf{y}$ also be run backwards, necessitating either remembering the trajectory of $\mathbf{y}$, which can require prohibitive amounts of storage, or the backwards simulation of **y** itself, which is typically ill conditioned.

However, running the system backwards can be avoided. Given guesses for the correct values of $z_i(t_0)$, one can simulate **y**, **z** and ζ forward from t_0 to t_1 and then update the guesses in order to minimize B where

$$B = \frac{1}{2}\sum_i z_i(t_1)^2 \tag{13}$$

by making use of the fact that

$$\frac{\partial B}{\partial z_j(t_0)} = \sum_i z_i(t_1)\zeta_{ij}(t_1). \tag{14}$$

For notational convenience, let $b_i = \partial B/\partial z_i(t_0)$. We can use a Newton-Raphson method with the appropriate modification for the fact that B has a minimum of zero, resulting in the simple update rule

$$z_i(t_0) \leftarrow z_i(t_0) - 2\frac{B}{||\mathbf{b}||^2}b_i. \tag{15}$$

During our simulation we can accumulate the appropriate integrals, so if our guesses for $z_i(t_0)$ were nearly correct we will have computed nearly correct values for $\partial E/\partial w_{ij}$ and $\partial E/\partial T_i$. If the w_{ij} change slowly the correct values for $z_i(t_0)$ will change slowly, so tolerable accuracy can be obtained by using the $\partial E/\partial w_{ij}$ computed from the slightly incorrect values for $z_i(t_0)$ while simultaneously updating the $z_i(t_0)$ for future use, eliminating the need for an inner loop which iterates to find the correct values for the $z_i(t_0)$. This argument assumes that the quadratic convergence of the Newton-Raphson method dominates the linear divergence of the changes to the w_{ij}, which can be guaranteed by choosing suitably low learning parameters.

4 Future Work

We are planning on performing the following experiments in the immediate future:

- Learn a simple xor problem, with the functional requiring the output to be correct after 2 time units.
- Follow a square trajectory in state space, where the desired trajectories of two visible units are specified explicitly using a functional of the form

$$E = \frac{1}{2}\sum_i \int_{t_0}^{t_1} s_i(y_i - d_i)^2 dt \tag{16}$$

  where d_i is the desired trajectory for y_i and s_i is the importance of y_i attaining d_i at time t. For this functional, the instantaneous error takes on the particularly simple form $e_i = s_i(y_i - d_i)$. Note that following a square trajectory requires the use of hidden units.

- Teach two visible units to follow a circular trajectory in state space, but rather than specifying the trajectory explicitly, require that the trajectory be on the circle with center (c_1, c_2) and radius r and that the velocity be v using a functional like

$$E = \int_{t_0}^{t_1} ((y_1 - c_1)^2 + (y_2 - c_2)^2 - r^2)^2 + (y_1'^2 + y_2'^2 - v^2)^2 \, dt. \tag{17}$$

Assuming that these simulations are successful, we are planning on using this procedure in the domain of control as part of the author's thesis work on learning to control robot manipulators using connectionist networks [1].

5 Acknowledgments

This research was sponsored in part by National Science Foundation grant EET-8716324, and by the Office of Naval Research under contract number N00014-86-K-0678. Barak Pearlmutter is a Fannie and John Hertz Foundation fellow.

References

[1] Barak Pearlmutter. Manipulator control using a connectionist network. May 1988. Unpublished thesis proposal.

[2] Fernando Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 19(59):2229–2232, 1987.

[3] David E. Rumelhart, Geoffrey E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition*, Bradford Books, Cambridge, MA, 1986.

[4] Patrice Y. Simard, Mary B. Ottaway, and Dana H. Ballard. *Analysis of Recurrent Backpropagation.* Technical Report 253, Department of Computer Science, University of Rochester, June 1987.