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

The Artificial Intelligence and Psychology Project

Departments of Computer Science and Psychology
Carnegie Mellon University

Learning Research and Development Center
University of Pittsburgh

Approved for public release; distribution unlimited.
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. Reproduction in whole or in part is permitted for purposes of the United States Government. Approved for public release; distribution unlimited.
We describe a procedure for finding $\partial E/\partial w_j$ where $E$ is an arbitrary functional of the temporal trajectory of the states of a continuous recurrent network and $w_j$ 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.
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 $\frac{\partial E}{\partial w_j}$ where $E$ is an arbitrary functional of the temporal trajectory of the states of a continuous recurrent network and $w_j$ 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 fixedpoint of a recurrent temporally continuous generalization of backpropagation networks [3]. Such networks are governed by the coupled differential equations

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

where

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

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

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

1
Consider $E(y)$, an arbitrary functional of the trajectory taken by $y$ between $t_0$ and $t_i$. Below, we develop a technique for computing $\partial E(y)/\partial w_v$ and $\partial E(y)/\partial T_i$, thus allowing us to do gradient descent in the weights and time constants so as to minimize $E$. The computation of $\partial E/\partial w_v$ 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

$$e_i(t) = \lim_{\epsilon \to 0} \epsilon^{-1} \frac{\delta E(y)}{\delta y_i(t + \epsilon)}$$

In the usual case where $E$ is of the form $E(y) = \int_{t_0}^{t_i} f(y(t), t) \, dt$ this means that $e_i(t) = \partial f(y(t), t) / \partial y_i(t)$. 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

$$z_i(t) = \frac{\partial E(y(t + \xi))}{\partial \xi} \bigg|_{\xi = 0}$$

where $y(t + \xi)$ is the same as $y$ except that $dy_i/dt$ has a Dirac delta function of magnitude $\xi$ 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 $y$ considered in $e_i(t)$ (left) and $z_i(t)$ (right).](image)

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

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

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

---

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

---
Consider incrementing $y_i(t)$ by $\epsilon$ and letting this change propagate forward. The differential of $E(y)$ w.r.t. $\epsilon$ is thus the sum of the differentials of $E(y)$ 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 $z_i(t)$,

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

(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 $y_i(t)$ 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 $z_i(t) = z_i(t + \Delta t) - \Delta t \frac{dz_i}{dt}(t + \Delta t)$, and taking the limit as $\Delta t \to 0$ we obtain a differential equation,

$$\frac{dz_i}{dt} = \frac{1}{T_i}z_i - e_i - \sum_j \frac{1}{T_j} w_{ij}\sigma'(x_j(t))z_j.$$

(6)

Let

$$u_{ij}(t) = \frac{\partial E(y^{(i,j,\xi,0)})}{\partial \xi} \Big|_{\xi = 0}$$

(7)

where $y^{(i,j,\xi,0)}$ is the same as $y$ except that $w_{ij}$ is increased by $\xi$ from $t$ through $t_i$. Again examining Figure 2, we see that the appropriate difference equation for $u$ is

$$u_{ij}(t) = u_{ij}(t + \Delta t) + \Delta t u_{ij}(t)\sigma'(x_j(t))\frac{1}{T_j}z_j(t + \Delta t),$$
which leads to the differential equation

$$\frac{d^r q}{dt^r} = - \frac{1}{T_i} \gamma_i \sigma'(x_i) z_i$$

which we can integrate from $t_0$ to $t_1$. By substituting $v_i(t_1) = 0$ and $v_i(t_0) = \partial E/\partial w_i$

into the resulting equation we eliminate $r$ and end up with

$$\frac{\partial E}{\partial w_i} = \frac{1}{T_i} \int_{t_0}^{t_1} y_i \sigma'(x_i) z_i dt.$$  \hfill (8)

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

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

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

$$\zeta_i(t) = \frac{\partial z_i(t)}{\partial z_i(t_0)},$$  \hfill (10)

and take the partial of (6) with respect to $z_i(t_0)$, substituting in $\zeta_i$ where appropriate. This results in a differential equation for $\zeta_i$,

$$\frac{d\zeta_i}{dt} = \frac{1}{T_i} \zeta_i - \sum_k \frac{1}{T_k} w_{ik} \sigma'(x_k) \zeta_k.$$  \hfill (11)

and for boundary conditions we note that

$$\zeta_i(t_0) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise}. \end{cases}$$  \hfill (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 $y$ forward from $t_0$ to $t_1$, set the boundary conditions $z_i(t_1) = 0$, and simulate the system $z$ backwards from $t_1$ to $t_0$ while numerically integrating $z_i \sigma'(x_i) y_i$ and $z_i dy_i/dt$ thus computing $\partial E/\partial w_i$ 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 $z$ as well as the integrals being computed require that $y$ also be run backwards, necessitating either remembering the trajectory of $y$, 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 $z,(t_0)$, one can simulate $y$, $z$ and $\zeta$ forward from $t_0$ to $t_1$ and then update the guesses in order to minimize $B$ where

$$B = \frac{1}{2} \sum z_i(t_1)^2$$

by making use of the fact that

$$\frac{\partial B}{\partial z_i(t_0)} = \sum z_i(t_1) \zeta_i'(t_1).$$

For notational convenience, let $b_i = \frac{\partial B}{\partial z_i(t_0)}$. 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

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

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

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

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

\[
E = \int_a^b ((y_1 - c_1)^2 + (y_2 - c_2)^2 - r^2)^2 + (y_1' - y_2')^2 \ dt
\]

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


