Training Recurrent Networks

A recurrent network is one in which there is feedback from a neuron's output to its input.

Various models exist:
- Jordan Network (feedback from net output to input)
- Elman Network ("partially recurrent": feedback from internal state output to input)
- Hopfield Network (a special class, studied later)

Jordan vs. Elman Networks

Elman Networks

Reference


Elman PRNN for Bankruptcy Prediction
(Oak Ridge Nat'l Lab, Grady, et al.)
How to Train an Elman Network?

One way:
- Initialize the state values to nominal.
- Repeat
  - Simulate one step of the network.
  - Compute the actual output.
  - Backpropagate the error.
  - Adjust the weights.
  - Compute the next state.
- Until the error is sufficiently low.

Training Feedback Weights

Elman Schematic

\[ y(k) = w_1 y(k-1) + w_2 x(k) \]

Training Feedback Weights

\[
\begin{align*}
\frac{\partial}{\partial w_2} y(k) &= x(k) \\
\frac{\partial}{\partial w_1} y(k) &= \frac{\partial}{\partial w_1} w_1 y(k-1) \\
&= w_1 \frac{\partial}{\partial w_1} y(k-1) \\
&= y(k-1) + w_1 \frac{\partial}{\partial w_1} y(k-1) \\
&= y(k-1) + y(k-2) + w_1 \frac{\partial}{\partial w_1} y(k-2) \\
&= y(k-1) + y(k-2) + y(k-3) + \ldots
\end{align*}
\]

Demos of Elman Training

Two demos:
- Matlab appelm1
- NAS demo 11.2

Elman Demo (appelm1)

In this demo, an Elman network is trained to track the amplitude of a sine wave, by training on a signal of two different amplitudes.

Training MSE plot
Other Possible Ways to Train an Elman Network

- BPPT (Backpropagation Through Time, seen last time) would be another way: unroll the network some large number of levels, backpropagate, average the weight changes over the unrolled stages to get a single set of weight changes.

- (See NAS demo 11.3.)

Real-Time Recurrent Learning (RTRL)

See http://www.dlsi.ua.es/~mlf/nnafmc/pbook/node29.html

- another approach to training recurrent networks, due to Williams and Zipser (1989, 1995)
- No "unrolling" is necessary
- "dual" of BPPT? (NAS)
- Later shown to be a special case of EKF (Extended Kalman Filter)
- Gradient derivatives (as well as state) at time t are computed in terms of their values at time t-1:
- Here \( t \) is the state, \( g' \) is the derivative of the activation function, \( x \) is the error, and \( u \) is the external input.

\[
\frac{\partial x(t)}{\partial W_{ij}} = g'(E(t)) \left( \delta_{W_{ij}[t]} + \sum_{j=1}^{n} W_{ij} \frac{\partial x(t-1)}{\partial W_{ij}} \right)
\]

\[
E(t) = \sum_{j=1}^{n} W_{ij}^E x(t-1) + \sum_{j=1}^{n} W_{ij}^E u(t) + W_{ij}^E
\]