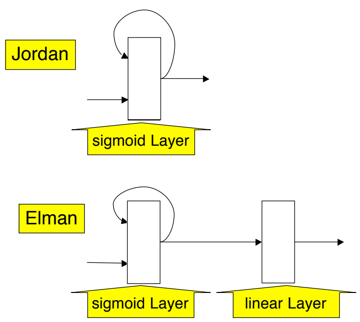
Training Recurrent Networks

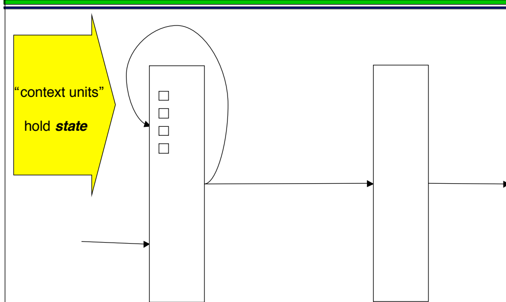
Recurrent Network

- 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



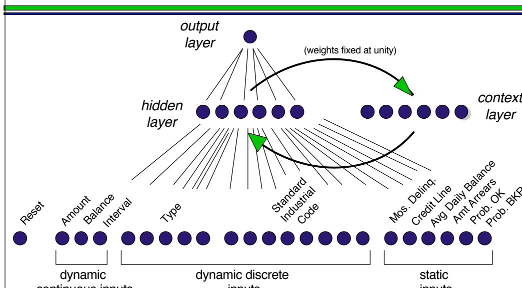
Elman Networks



Reference

- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179-211.
- Neural Networks: Automata and Formal Models of Computation, Mikel L. Forcada, http://www.dlsi.ua.es/~mlf/nnafmc/pbook/pbook.html

Elman PRNN for Bankruptcy Prediction (Oak Ridge Nat'l Lab, Grady, et al.)

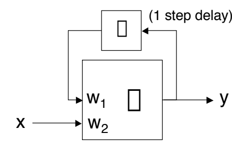


How to Train an Elman Network?

- 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

$y(k) = w_1\, y(k-1) + w_2\, x(k)$

$\partial/\partial w_2\;\; y(k) = x(k)$

Conclusion: The sensitivities will depend on all previous values of the output.

$\partial/\partial w_1\;\; y(k)$

$= \partial/\partial w_1\; w_1\, y(k-1)$

$= y(k-1) + w_1\, \partial/\partial w_1\, y(k-1)$ [derivative of a product]

$= y(k-1) + y(k-2) + w_1\, \partial/\partial w_1\, y(k-2)$ [a recurrence]

$= y(k-1) + y(k-2) + y(k-3) + \ldots$

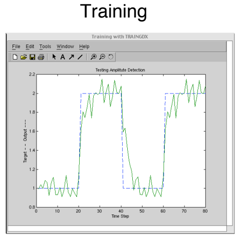
Demos of Elman Training

- Two demos:
  - Matlab appelm1
  - NAS demo 11.2

Elman Demo (appelm1)

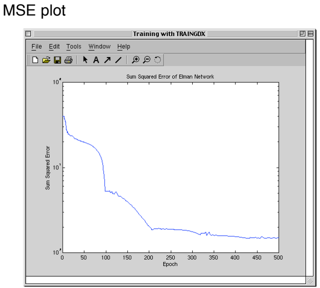
In this demo, an Elman network is trained to track the ***amplitude*** of a sine wave, by training on a signal of two different amplitudes.

Training



Elman Demo (appelm1)

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 □ is the state, g’ is the derivative of the activation function, x is the error, and u is the external input.

$$\frac{\partial x_i[t]}{\partial W_{kl}^{xx}} = g'(\Xi_i[t]) \left(\delta_{ik} x_l[t] + \sum_{j=1}^{n_X} W_{ij}^{xx} \frac{\partial x_j[t-1]}{\partial W_{kl}^{xx}} \right),$$

$$\Xi_i[t] = \sum_{j=1}^{n_X} W_{ij}^{xx} x_j[t-1] + \sum_{j=1}^{n_U} W_{ij}^{xu} u_j[t] + W_i^x$$