Recurrent Backprop Networks

15-486/782: Artificial Neural Networks
David S. Touretzky

Fall 2006

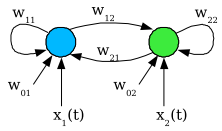
Recurrent Networks

- "Recurrent" means feeding back on itself.
- Why have recurrent nets?
  - Can process or output temporal sequences
  - More computational power? Oscillations; chaos.
  - More "brain-like"?
  - Equivalent to deep feed-forward net with constrained weights (inductive bias)
- Drawbacks:
  - Harder to analyze
  - Possibly slow training
  - Stability issues

Types of Recurrent Network

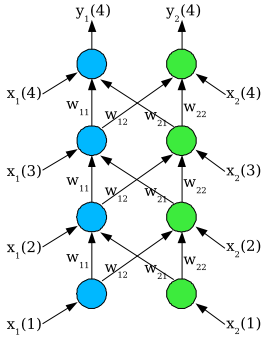
- Recurrent backprop network
  - Discrete time
    - Simple Recurrent Network (SRN): Elman net
    - Jordan net
    - Fixed-point attractor networks
  - Continuous time
- Spin-glass models (Hopfield, Boltzmann)
- Interactive-activation models (cognitive modeling)
- Competitive networks (self-organizing feature maps)

Discrete Time Recurrent Backprop



$$net_i(t) = x_i(t) + w_{0i} + \sum_j y_j(t-1)\, w_{ji}$$

$$y_i(t) = f(net_i(t))$$

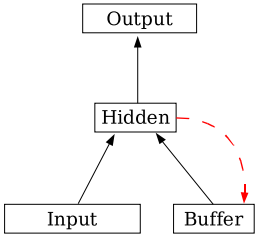


"Backpropagation through time"

Varieties of Discrete Time Recurrent Backprop

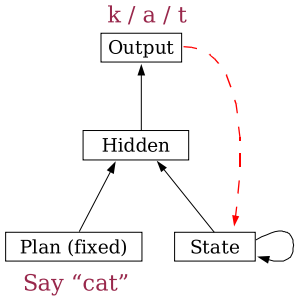
- Input vector: fixed **x**, or varying over time **x**(t)
- Desired output: look only at last step **y**(n), or at entire sequence **y**(1) ... **y**(n)
- Backprop step: through one layer, or through all layers (unroll the entire network: BPTT).
- Note: weights are always constrained to be the same across layers.

Simple Recurrent Network (Elman Net)



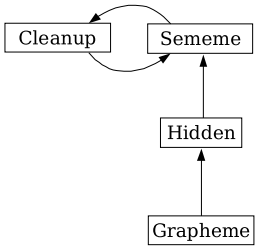
- Inputs presented sequentially.
- Outputs produced sequentially
- Network trained to predict the next output in the sequence.
- Error back-propagated for one time step.

Jordan Net: Sequence Generation (Motor Planning)



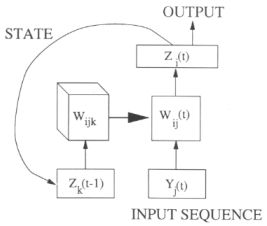
- Fixed input (the “plan”).
- Sequential output.
- Output (not hidden) vector is recirculated via the state units.
- $State_i(t+1) = Output_i(t) + \gamma\, State_i(t)$
  Exponential decay

Hinton/Shallice/Plaut Model



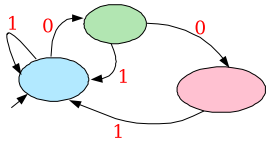
- Fixed input: written form of a word.
- Fixed output: semantic features of the word.
- Network settles to a fixed point (attractor state).
- Error back-propagated through time.

Pollack: Learning Finite State Recognizers for Binary Languages



- Input one bit at a time.
- Three hiddens z_1 ... z_3 plus one output z_4
- Multiplicative weights
- Network trained to output a "1" if the substring seen so far is in the language.
- Use small training sets from Tomita's work on FSA learning.

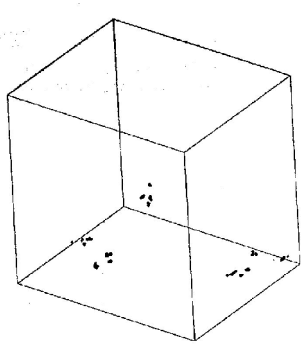
"String Contains No 000"



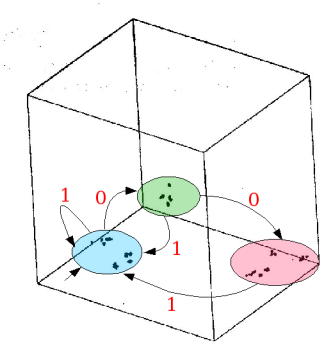
- Accept set: 0, 1, 10, 01, 00, 010, 11100, 100100, 001111110100
- Reject set: 000, 11000, 0001, 00000000, 11111000011, 1101010000010111, 1010010001, 0000, 00000

"String Contains No 000": Hidden Unit State Space

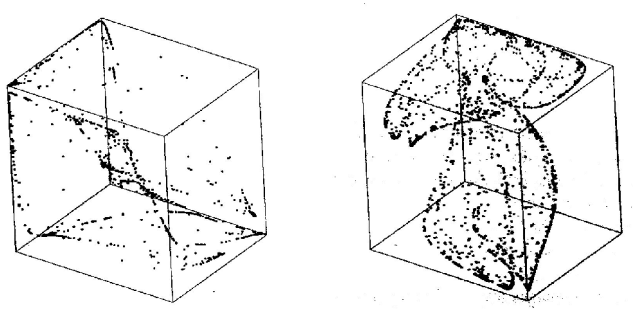
Test on all sequences up to length 10 (2048 total).



"String Contains No 000": Hidden Unit State Space



Learning Fails on Some Problems (Training Set Too Small)



Learning Context-Free Grammars with an Elman Net

- Obtaining verb agreement requires state:
  boy who boys chase CHASES boy
  boys who boys chase CHASE boy
- Context-free grammars permit arbitrarily deep embeddings, so a "stack" or "tape" is required.:
  "**boy**$_1$ who chased$_1$ boys$_2$ **sees**$_1$ **girl**"
- A recursive (context-free) grammar cannot be recognized by a finite state machine.
- But how well can an Elman net do?

The Grammar

S = NP1 VP1 | NP2 VP2

NP1 = N1 | N1 who RC1
NP2 = N2 | N2 who RC2

N1 = boy | girl | dog | ... | PN
N2 = boys | girls | dogs | ...
PN = John | Mary | ...
N = N1 | N2

RC1 = VP1 | N1 VN1 | N2 VN2
RC2 = VP2 | N1 VN1 | N2 VN2

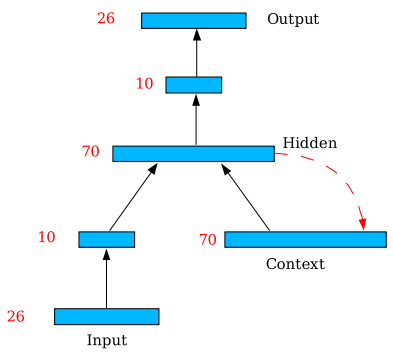
Grammar (continued)

VP1 = VN1 | VD1 (N) | VR1 N
VP2 = VN2 | VD2 (N) | VR2 N

VN1 = walks | lives | ...	no direct object
VD1 = sees | ...	optional direct obj.
VR1 = chases | ...	required direct obj.

VN2 = walk | live | ...
VD2 = see | ...
VR2 = chase | ...

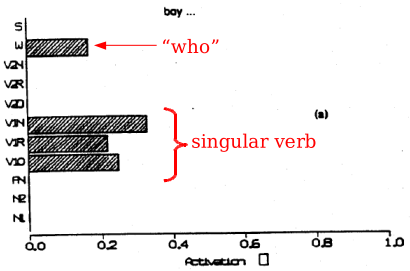
Architecture



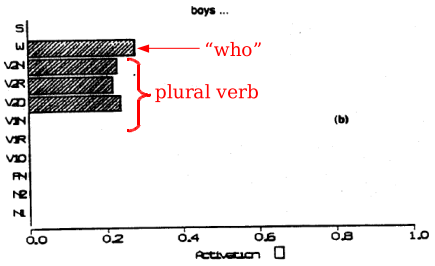
Training Regimen

- Feed the network one word at a time.
- Require it to predict the next word.
- Use backprop to adjust the weights.
- Trained network will mimic the probability distribution of words that could possibly follow from the previous input.

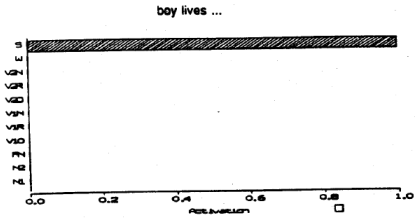
Number Agreement: boy



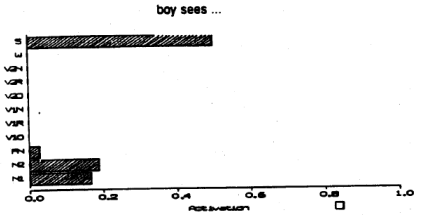
Number Agreement: boys



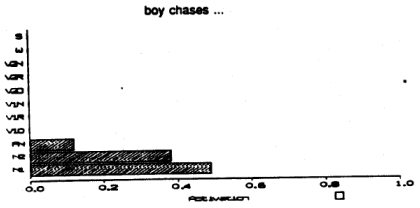
No Direct Object



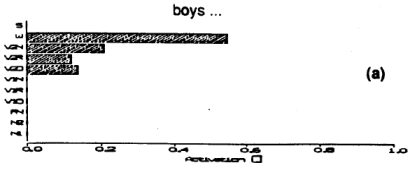
Optional Direct Object



Required Direct Object



Sample Sentence

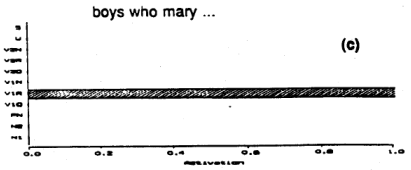


Plural noun predicts “who” or plural verb...

Sample Sentence (cont.)

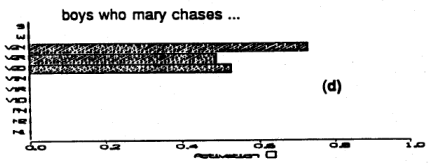
"Who" predicts a relative clause...

Sample Sentence (cont.)



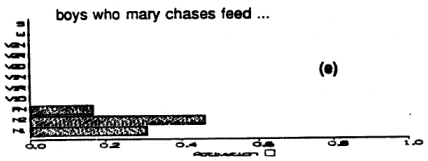
Singular verb requiring a direct object...

Sample Sentence (cont.)



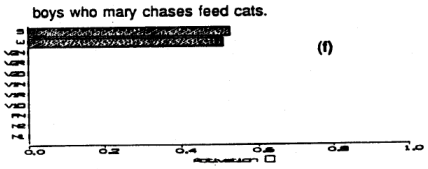
Plural verb...

Sample Sentence (cont.)



Noun phrase...

Sample Sentence (cont.)

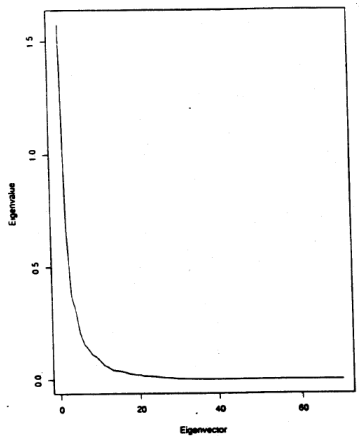


End of sentence, or relative clause...

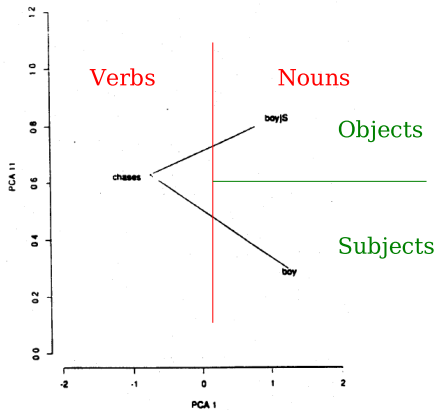
Analyzing Elman's Network

- Can't visualize state space directly: 70 hidden units!
- Instead, Elman performed a principal components analysis on the hidden unit activation patterns.
- Plot hidden unit states in a 2D subspace defined by a selected pair of principle components
- Each run of the network on a sentence generates a "trajectory" through hidden unit state space.

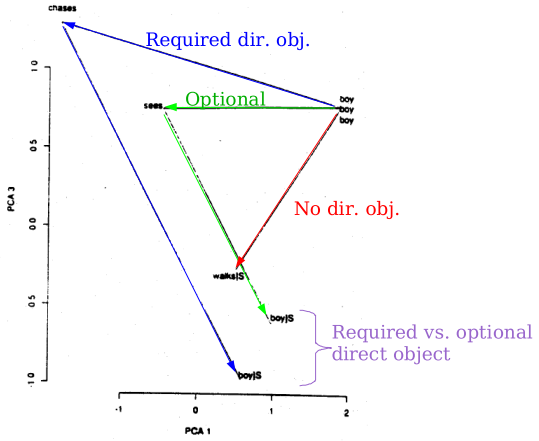
Only A Few Important Eigenvectors



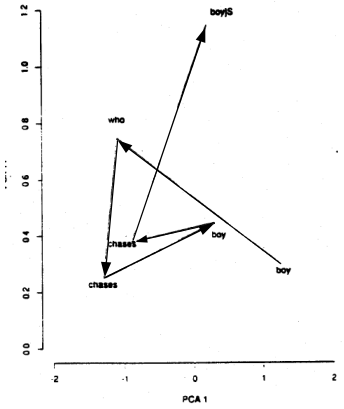
What Do The Eigvenvectors Encode?



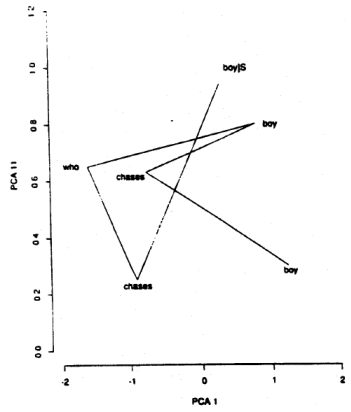
What Do the Eigenvectors Encode?



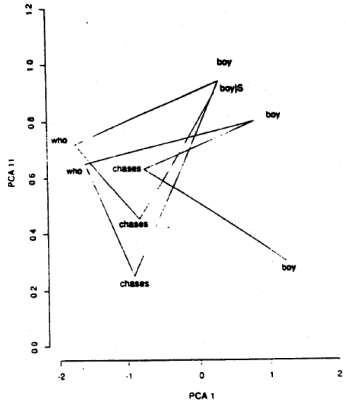
Nested Structure



Another Nested Structure



Generalization Test: Deeper Nesting

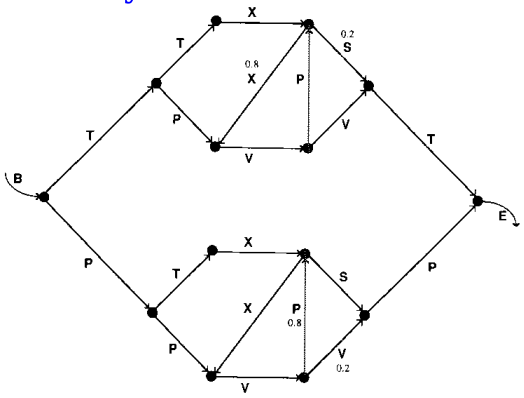


Elman's Conclusions

- Some notion of recursive structure can be learned by a dynamical state machine using backprop.
- The representation of a word is not independent of context. Words generate context-dependent representations that encode the syntactic environment in which the word was seen.
- This may have important psychological implications for human language learning.

Cleeremans et al.: Sensitivity to Probability Distribution

This grammar is only learnable when the two identical embedded subgraphs have different transition probabilities.



Real-Time Recurrent Learning (RTRL)

- Alternative to BPTT that does not require duplicating the nodes.
- Permits on-line learning.
- Let $E_k(t)$ be error for unit k (if desired output defined at time t).

$$\Delta w_{pq}(t) = -\eta \sum_k E_k(t) \frac{\partial V_k(t)}{\partial w_{pq}}$$

$$\frac{\partial V_i(t)}{\partial w_{pq}} = f'(net_i(t-1)) \left[\delta_{ip} V_q(t-1) + \sum_j w_{ij} \frac{\partial V_j(t-1)}{\partial w_{pq}} \right]$$

$$\frac{\partial V_i(0)}{\partial w_{pq}} = 0$$

RTRL

- For N fully-connected units, this requires N^3 derivatives.
- Updating each weight takes time proportional to N for each derivative, so a complete weight update takes N^4 operations. Slow!
- But if learning rate is small, updates can take place in real time at each step; don't have to store old states or pick a fixed size for the “unrolled” network.

Teacher Forcing

- Trick for speeding up training of recurrent backprop nets.
- Suppose at time t we want y_i to have value d_i.
  - Calculate error term as $d_i(t) - y_i(t)$
  - Then set $y_i(t)$ to $d_i(t)$
  - Now go on to calculate outputs for time t+1
- Allows the network to learn throughout the sequence from the beginning.

Continuous Time Recurrent Backprop

- Pineda, Almeida, Rowher and Forrest: continuous recurrent backprop with fixedpoints.
- State update equation:

$$\tau \frac{dV_i}{dt} = -V_i + g\left(\xi_i + \sum_j w_{ij} V_j\right)$$

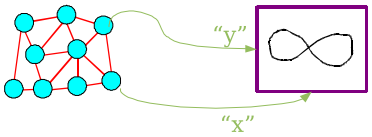
- Fixed points where $dV_i/dt = 0$:

$$V_i = g\left(\xi_i + \sum_j w_{ij} V_j\right)$$

- Result: an analog associative memory.

Pearlmutter's Time-Dependent Recurrent BP

- Temporally continuous, like Pineda
- Instead of fixedpoint attractors, can learn arbitrary trajectories through state space
- Can learn limit cycles
- Unlike RTRL, does not do online learning
- Only N^2 space and N^3 time



Continuous-Time BP (cont.)

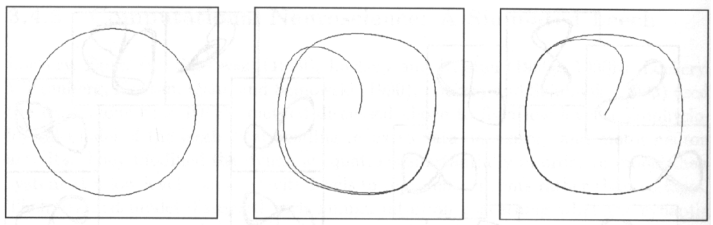


Figure 3.10: Desired states d_1 and d_2 plotted against each other (left); actual states y_1 and y_2 plotted against each other at epoch 1500 (center) and 12000 (right).

Continuous Time BP (cont.)

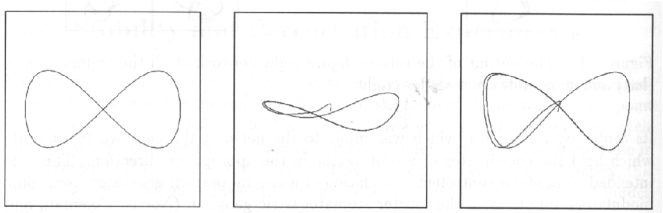


Figure 3.11: Desired states d_1 and d_2 plotted against each other (left); actual states y_1 and y_2 plotted against each other at epoch 3182 (center) and 20000 (right).

Noise Resistance

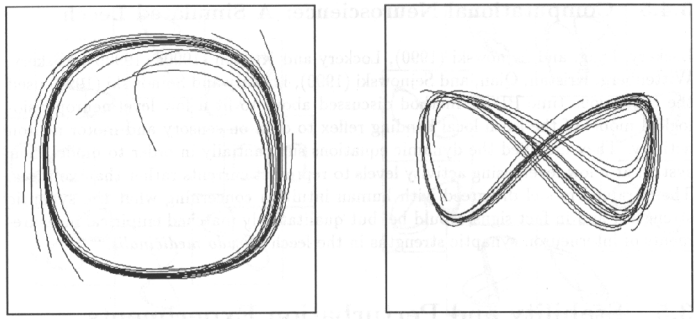
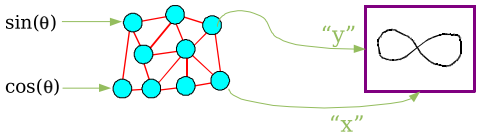


Figure 3.13: The output states y_1 and y_2 plotted against each other for a 1000 time unit run, with all the units in the network perturbed by a random amount about every 40 units of time. The perturbations in the circle network (left) were uniform in ± 0.1, and in the figure eight network (right) in ± 0.05.

Learning a Family of Sequences



Sequences Parameterized by θ

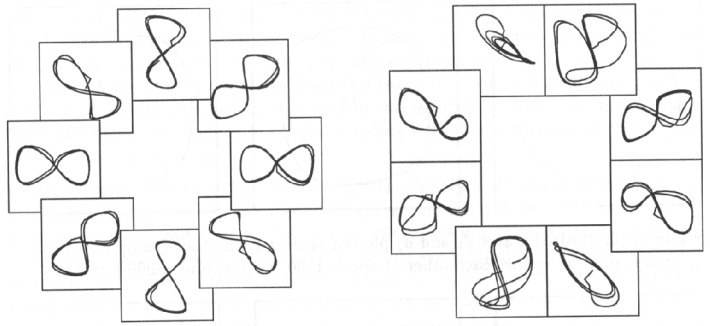


Figure 3.12: The output of the rotated figure eight network at all the trained angles (left) and some untrained angles (right).

Summary

- Continuous real-time sequence learning is the most "brain like" architecture, but...
  - It's very slow to train
  - Learning is unstable
  - Many parameters to adjust
- Discrete recurrent networks (BPTT or RTRL) are easier to use, and have been applied to a variety of cognitive modeling or knowledge representationt tasks.