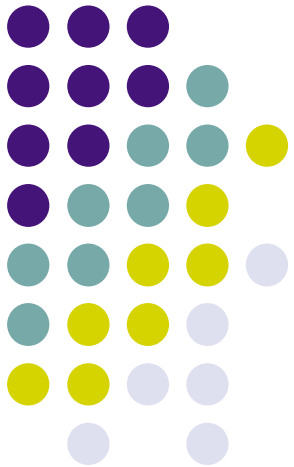
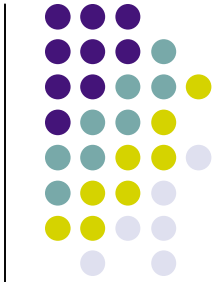
Vanishing Gradient In Recurrent Neural Models

Garret Heckel



Based on the paper *Recurrent Neural Net Learning and Vanishing Gradient* by Sepp Hochreiter

The Goal



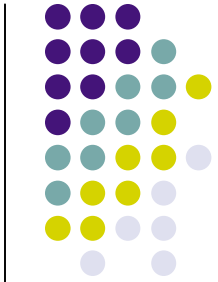
- Find temporal dependencies in data with a recurrent neural network.

- When an error value is found, apply it to inputs seen an indefinite number of epochs ago.

The Problem

- When using gradient based training rules, the “error signal” that is applied to previous inputs tends to vanish.
- Previous inputs have a negligible effect on the gradient.
- Because of this, long term dependencies in the data are often overlooked.

Gradient Descent

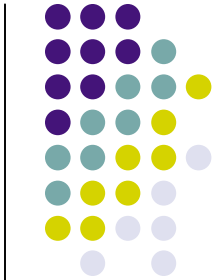


- Propagating an error back q time steps from time t scales the error by

  f’(t-q) * sum(f’(t-q+1) * sum(…f’(t-1)))

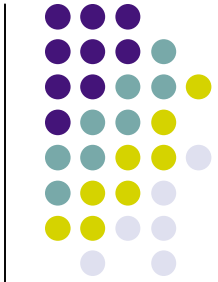
- This amounts to n^(q-1) terms.

Gradient Descent, cont.



- These n^(q-1) terms may have different signs.
- Increasing n does not necessarily increase the error signal, but does increase the expected value of the magnitude of the error signal.

Gradient Descent, cont



- With a logsig activation function, $f'_{max} = .25$

- We find that if $w_{max} < 4/n$, we get exponential decay proportional to

  $$(n * w_{max} / 4) \wedge q$$

  for the gradient after q timesteps

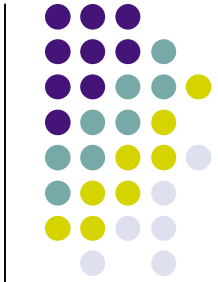
Ways to Avoid the Vanishing Gradient

1. Gradient free methods (random guessing, evolution)

2. Better gradients (Newton style methods, discrete error propagation)

3. Special Architectures(TDNN, LSTM, multiplicative units or delay lines, simulated annealing)

Network Comparison

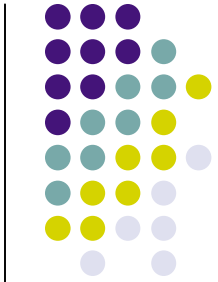


Experiment 1: Recognizing a grammar-

Net tries to predict next character in a sequence generated by some probabilistic rule.

There are no long time lags, so gradient based networks should be effective.

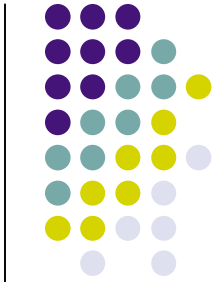
Network Comparison, cont



Experiment 1: Results-

Network	Hidden Units	% Success	Avg. Epoch
RTRL	3	Some %	175,000
RTRL	12	Some %	25,000
ELM	15	0	>200,000
RCC	7-9	50	180,000
LSTM	3x2 blocks	100	8,500

Network Comparison, again



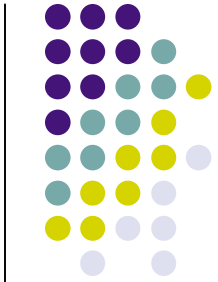
Experiment 2: Long Time Lags-

Train on the sequence $(x, a_1, \ldots, a_n, x)$

Involves minimal time lag of n + 1

For large n, gradient based nets may have problems finding a correlation between the start and end of the sequence.

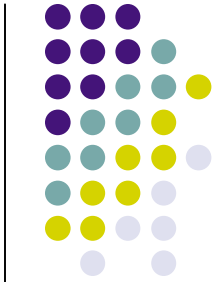
Network Comparison, cont



Experiment 2: Results-

Network	Delay	% Success	Avg. Epoch
RTRL	4	78	175,000
RTRL	10	0	25,000
BPTT	100	0	>200,000
CH	100	33	180,000
LSTM	100	100	8,500

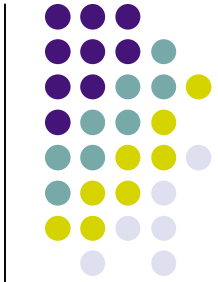
LSTM



- The LSTM is a gradient based network. Why does it still work so well?

- For dependencies to be found after 100 time steps, gradient must be preserved somehow.

LSTM, cont

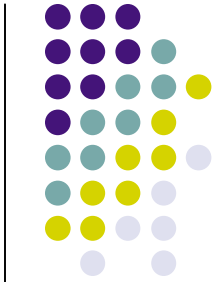


- Constant Error Propagation: naïve approach

  Make a neuron with activation function f(x)=x and with the weight to itself 1.

  Error is perpetuated back through time with derivative 1 and every epoch, the neuron passes the same value to itself + the other neuron's contributions.

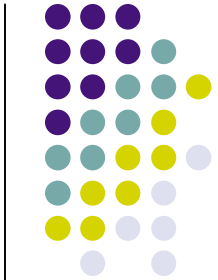
LSTM, cont



- Other neurons tend to disrupt values being perpetuated through these cells.

- Useless input can clutter these cells quickly.

LSTM, cont



- Constant Error Propagation: Memory Cells
- The solution that the LSTM model provides is to create similar memory cells to the naïve approach, then add gates to control input and output values.
- Gates let similar error values into the cell, and can provide output when needed.

References

- S. Hochreiter. *Recurrent Neural Net Learning and Vanishing Gradient*, International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems, 6(2):107-116, Munchen, 1998.

- S. Hochreiter and J. Schmidhuber. *Long Short-Term Memory*, Technical Report FKI-207-95, Technische Universitat Munchen 1995. Revised 1996.