CSC321: Neural Networks

Lecture 6: Learning in recurrent networks

Geoffrey Hinton

Backpropagation with weight constraints

- It is easy to modify the backprop algorithm to incorporate linear constraints between the weights.
- We compute the gradients as usual, and then modify the gradients so that they satisfy the constraints.
  - So if the weights started off satisfying the constraints, they will continue to satisfy them.

$$To\ \ constrain: \quad w_1 = w_2$$
$$we\ \ need: \quad \Delta w_1 = \Delta w_2$$

$$compute: \quad \frac{\partial E}{\partial w_1} \quad and \quad \frac{\partial E}{\partial w_2}$$

$$use \quad \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2} \quad for\ w_1\ and\ w_2$$

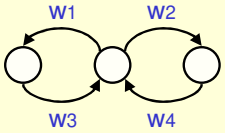
Recurrent networks

- If the connectivity has directed cycles, the network can do much more than just computing a fixed sequence of non-linear transforms:
  - It can oscillate. Good for motor control?
  - It can settle to point attractors.
    - Good for classification
  - It can behave chaotically
    - But that is usually a bad idea for information processing.
  - It can remember things for a long time.
    - The network has internal state. It can decide to ignore the input for a while if it wants to.
  - It can model sequential data in a natural way.
    - No need to use delay taps to spatialize time.

An advantage of modeling sequential data

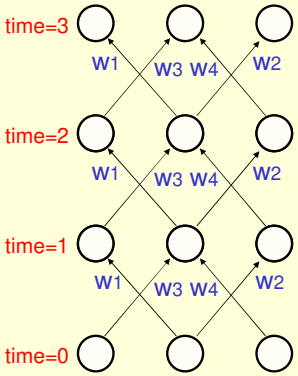
- We can get a teaching signal by trying to predict the next term in a series.
  - This seems much more natural than trying to predict one pixel in an image from the other pixels.

The equivalence between layered, feedforward nets and recurrent nets



Assume that there is a time delay of 1 in using each connection.

The recurrent net is just a layered net that keeps reusing the same weights.

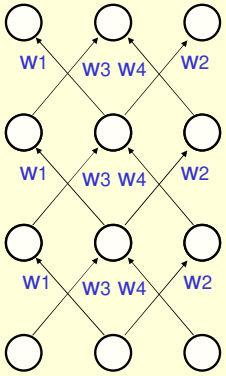


Backpropagation through time

- We could convert the recurrent net into a layered, feed-forward net and then train the feed-forward net with weight constraints.
  - This is clumsy. It is better to move the algorithm to the recurrent domain rather than moving the network to the feed-forward domain.
- So the forward pass builds up a stack of the activities of all the units at each time step. The backward pass peels activities off the stack to compute the error derivatives at each time step.
- After the backward pass we add together the derivatives at all the different times for each weight.
- There is an irritating extra problem:
  - We need to specify the initial state of all the non-input units. We could backpropagate all the way to these initial activities and learn the best values for them.

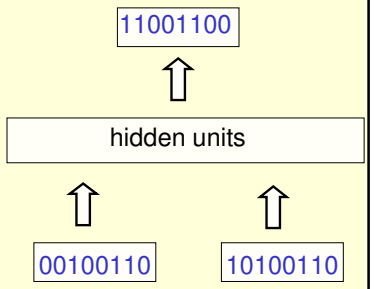
Teaching signals for recurrent networks

- We can specify targets in several ways:
  - Specify desired final activities of all the units
  - Specify desired activities of all units for the last few steps
    - Good for learning attractors
    - It is easy to add in extra error derivatives as we backpropagate.
  - Specify the desired activity of a subset of the units.
    - The other units are "hidden"

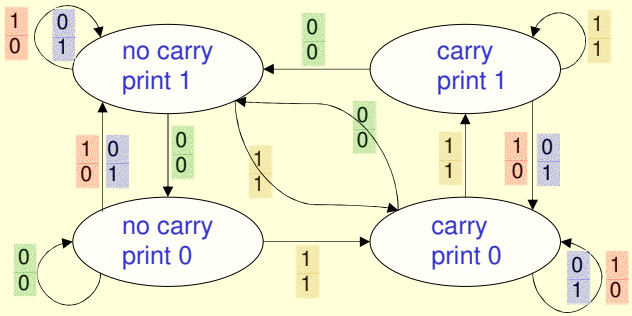


A good problem for a recurrent network

- We can train a feedforward net to do binary addition, but there are obvious regularities that it cannot capture:
  - We must decide in advance the maximum number of digits in each number.
  - The processing applied to the beginning of a long number does not generalize to the end of the long number because it uses different weights.
- As a result, feedforward nets do not generalize well on the binary addition task.



The algorithm for binary addition



This is a finite state automaton. It decides what transition to make by looking at the next column. It prints after making the transition.
It moves from right to left over the two input numbers.

A recurrent net for binary addition

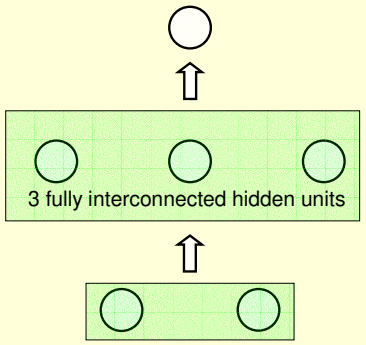
- The network has two input units and one output unit.
- The network is given two input digits at each time step.
- The desired output at each time step is the output for the column that was provided as input two time steps ago.
  - It takes one time step to update the hidden units based on the two input digits.
  - It takes another time step for the hidden units to cause the output.

0 0 1 1 0 1 0 0
0 1 0 0 1 1 0 1
1 0 0 0 0 0 0 1

← time

The connectivity of the network

- The 3 hidden units have all possible interconnections in all directions.
  - This allows a hidden activity pattern at one time step to vote for the hidden activity pattern at the next time step.
- The input units have feedforward connections that allow then to vote for the next hidden activity pattern.



What the network learns

- It learns four distinct pattern of activity for the 3 hidden units. These patterns correspond to the nodes in the finite state automaton.
  - Do not confuse units in a neural network with nodes in a finite state automaton. Nodes are like activity vectors.
  - The automaton is restricted to be in exactly one state at each time. The hidden units are restricted to have exactly one vector of activity at each time.
- A recurrent network can emulate a finite state automaton, but it is exponentially more powerful. With N hidden neurons it has 2^N possible binary activity vectors in the hidden units.
  - This is important when the input stream has several separate things going on at once. A finite state automaton cannot cope with this properly.