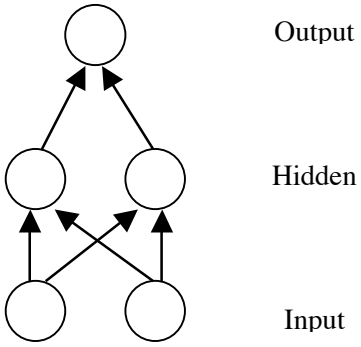
Tutorial 3: CSNLP

Networks with Hidden Layers

Date: 21 November 2001
Student's name:

In this tutorial we investigate the training networks with hidden layers, to compute the XOR function:

Input		Output
0	0	0
0	1	1
1	0	1
1	1	0



This tutorial is based on Chapter Four of Plunkett & Elman. But: **do not use the files for that chapter**

Ex 1: In tlearn, create a **New Project**, called `xor`. This will open the 3 usual file windows, but all will be empty. Write the `xor.cf` file so that it defines the above network architecture (you may want to look at the `xor.cf` you created in tutorial 2, as a starting point). Use the `xor.data` and `xor.teach` files that you created in tutorial 2. Give the configuration:

Ex 2: Under Training Option, set the network to:

- 4000 sweeps,
- seed = 5
- train randomly
- learning rate = 0.3
- momentum = 0.9
- Train the network

Examine the global error curve. At what point in training (approximately) does the network seem exhibit the most success in learning the solution to the problem.

Ex 3: Looking at the global error alone, can you be sure the network can learned the XOR function? Explain the criteria for your solution. Verify your conclusion by looking at the node activations for each pattern.

Ex 4: Under **Training options**, tell the network to dump the weights after every 500 sweeps, and train the network again (as above). You can now examine the state of the network at each of these (500 sweep) intervals:

- Under **Testing options**, select the **Earlier one** button, and select `xor.1001.wts`
- Verify the network has learned
- Repeat for xor.1501.wts and xor.2001.wts

The **Outputs** window shows the activation of the output for each pattern. What do you observe about how the network learns over time?

Ex 5: Examine the **Connection Weights** display after 2001 sweeps.

- Draw the network (including the bias node), and indicate which connections are positive and negative, or zero.
- What is the activation (rounding to 0 or 1) of the hidden layer units for *each* input pattern?

Is the solution basically similar or quite different from that discussed in the lecture.

Ex 6: Try training the network varying the learning rate parameter and momentum. Are you able to improve the performance of the network in learning the solution. If so, what parameters seem to work best?

Ex 7: Try varying the architecture of the network. Alter the `xor.cf` file so that the hidden layer has 3 units, and study the performance of the network. Try again using 5 hidden units.

- Does the learning rate of the network improve with either architecture?
- Examine the connection weights and node activities. How does the network appear to use the additional nodes?