Tutorial 2: CSNLP

Training 2-layer networks in Tlearn

Date: 7 November 2001
Student's name:

In this tutorial we investigate the training of simple 2-layer networks in Tlearn to learn boolean functions:

Input Activations		**Output Activations**		
Node 0	Node 1	Node 3		
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

For this tutorial, you will need the files from Chapter Three of Plunkett & Elman.

- Open existing project, and select 'and' from the chapter 3 directory of Plunkett.
- Examine the network configuration files, and network architecture to acquaint yourself with the network and training data.
- Under Training Option, set the network to:
  - 1000 sweeps,
  - seed = 1
  - train randomly
  - learning rate = 0.1
  - momentum = 0
- Train the network
- Verify the network has learned: the values in the "output" window should correspond to those in the "output" column below.

Input	Target	Output	(t-o)^2
0 0	0	0,147	0,022
0 1	0	0,297	0,088
1 0	0	0,334	0,112
1 1	1	0,552	0,201
		RMS:	0,325

Ex 1: Look at the error display graph (select line display). Does the final point on the error line appear to correspond to the above calculation of RMS error?

Ex 2: How many times has the network seen each input pattern after 1000 sweeps?

Ex 3: Examine the node activations for each of the input patterns (check this against the output activations in the "output" window). Has the network successfully learned the AND operation? What was your criterion for this decision?

Ex 4: Now try training the network (with the same parameters as above) for 10000 sweeps. Select "Verify the network has learned" to get the output activations.

- Calculate the RMS error (as above), and confirm whether or not it agrees with the value on tlearn's error display.
- Has the networks performance improved with the additional training? How?

Ex 5: Assume the following criterion for success: i.e. activation > 0.6 corresponds to 1, and activation < 0.4 corresponds to 0 (i.e. error within 0.4). Assume you are not allowed to inspect the performance of the network on individual patterns, but can only use the global RMS error.

- How low must the global RMS error be to *guarantee* the network has learned the problem?
- Under training options, select the "more" button. You can set training to stop once global RMS error falls below the some value. Enter your computed value for the desired RMS, and train the network again. How many sweeps were required to reach the error threshold?

Ex 6: Try training the network (again for 1000 sweeps) using three different random seeds for the weights (e.g. 1,3,6). Examine the error display.

- Does the initial start state (set of weights) affect the error?
- Is the end result (after 1000 sweeps) substantially different?

Ex 7: Open the OR project.

- Is tlearn able to learn the solution to this problem?
- Try varying the learning rate, and briefly report you observation for 3 different parameters.

Ex 8: Copy the 3 main files for the OR network, and name them: `xor.teach xor.data xor.cf`. In tlearn, create a new project, and call it `xor`. Adjust the xor.teach file accordingly.

- Try training the network as above, with a 1000 sweeps. Do ed the network learn XOR?
- Try longer training, varying the seed, and varying the learning rate. Can you make the network learn XOR?

Things to think about:

For these networks, look at the solution the network arrives at (i.e. the weights) and confirm for yourself why these settings of weights work for AND and OR.