Computational Psycholinguistics: Tlearn Tutorial 3

Simple Recurrent Networks

Date: 1 February 2012
Due: 6 February 2012
Student's name:

In this tutorial, the aim is to familiarize yourself with the processing of sequences over time, using simple recurrent networks. This tutorial is based on Chapter 8 of Plunkett & Elman, and takes you through the learning of letter sequences "**ba dii guuu"** as discussed in the lecture.

To begin, the encoding scheme used in this simulation is slightly different from that in Elman's original simulations (this is given in the `codes` file):

b	1	1	0	0
d	1	0	1	0
g	1	0	0	1
a	0	1	0	0
i	0	0	1	0
u	0	0	0	1

As with the original encoding, the first bit represents the consonant feature, the other three, a localist representation of each consonant and vowel.

Ex 1: The `letters` file contains a random sequence of 2993 letters. Open the file and convert it to a vector representation using the `Translate` option under `Edit` and selecting the pattern file `codes` (`letters` must be open and active). Save the resulting training file as `srn.data`.

Ex 2: Also create the file the `srn.teach` by copying of `srn.data` and moving the first line to the end of the file.

Ex 3: Open the project file `srn`. Examine the network architecture. Based on the network configuration `srn.cf`, draw the network as a conventional SRN, indicating which node numbers are the inputs, output, hidden, and context units (do not draw each node separately, but use slabs to represent unit groups).

Ex 4: Why do you think the recurrent connections from the hidden to the context units are one to one with fixed weights?

Ex 5: Train the network (sequentially!) with learning rate = 0.1 and momentum = 0.3 for 70000 sweeps. How many epochs is this?

Ex 6: Monitor the RMS error during training, and explain why it seems to stay so high.

Ex 7: Examine the file `test.data`. What letter sequence is being tested by this file?

Ex 8: Test the network using this file. Make sure the simulator is set to `Calculate error` under `Testing Options`, so you can easily examine the error behavior for each letter in the sequence. Sketch the error plot, annotated with letters at each point.

Ex 9: How well has the network learned to predict the next element in the sequence? Does it correctly predict the vowel following a consonant? Does it correctly predict the number of vowels?

Ex 10: Does the network correctly predict when a consonant will be the next item in the sequence? Explain why you think it does or doesn't.

Ex 11: Examine the network's solution by examining the hidden node activations associated with each input pattern and performing a cluster analysis of the hidden units on the test patterns:

- Clear the output display, then select `Probe selected nodes` under `Network`.
- `Output` now contains the hidden unit activations for the patterns in `test.data`.
- Remove the comment lines at the beginning of the file and save `Output` as `test.hid`.
- Create a names file called `test.lab` which contains **b a d i1 i2 g u1 u2 u3**, one per line.
- Then choose `Special`, `Cluster Analysis`.

Ex 12: Sketch the resulting dendrogram, and comment on what it suggests about the network's solution.