Tutorial 5: CSNLP

Autoassociation & Cluster Analysis

Date: 5 December 2001

Student's name:

In this tutorial, we use autoassociative networks to investigate how networks can learn new pattern encodings, and also introduce the hierarchical cluster analysis. This tutorial is based primarily on Chapter Five of Plunkett & Elman.

Part I: Autoassociation

Copy the Chapter 5 directory of Plunkett & Elman to your desktop, and open the project `auto1`. This network associates a 4 unit localist input pattern with itself, via a layer of 2 hidden units. This forces the network to find a distributed representation of the inputs.

Ex 1: Assuming a rounding criterion.What global level of error is required to guarantee tlearn has found a solution to the encoding problem?

Ex 2: Train the network, with a seed = 1, learning rate = 0.3, momentum = 0.9, updating the pattern after every sweep. Has the network learned successfully? What distributed representation is assigned to the 4 localist input patterns?

Ex 3: Create a set of noisy inputs, e.g. (.8 0 0 0 .1) (0 .7 .2 0 .1) etc. In testing options, tell the network to use `auto1.3000.wts`, and test the network on the noisy inputs using the `Verify the Network has Learned`. How does the network perform? Why might this behaviour be useful?

Ex 4: Do you think the network would learn successfully to autoassociate 5 input/outputs via 2 hidden units? Define a new project, to try this. What is your result?

Part II: Feature Discovery

Open the project auto3. This network autoassociates 7 bit patterns (distributed representations) via a hidden layer of 3 units. To solve the problem, the network must find similarities in patterns, or discover structure in the input so the patterns can be encoded more abstractly and therefore compactly.

Ex 5: Look at the patterns to see if you can find any obvious was of grouping. One technique we can use for examining similarity is hierarchical clustering. Tlearn does this for us, given two file: one with the patterns we're interested in clustering, then other with labels for the patterns. Create two files:

auto3.inp

```
1 1 0 1 0 0 1
1 0 0 1 1 1 1
1 0 1 1 1 0 0
1 1 0 1 0 0 0
0 1 0 0 0 0 1
0 1 0 0 0 1 0
0 0 0 0 1 0 0
0 0 1 0 1 1 0
```

auto3.lab

```
p1:1101001
p2:1001111
p3:1011100
p4:1101000
p5:0100001
p6:0100010
p7:0000100
p8:0010110
```

Now select Cluster Analysis from the Special menu, and enter these file names in the dialog box. Execute the cluster analysis, and examine the resulting hierarchy. Sketch it below.

Ex 6: Now train the network: seed = 1, rate = 0.3, momentum = 0.3, 4000 sweeps. Has the network learned to regenerate the patterns? Look at the activations of the hidden units for each pattern: can you see how the network has solved the problem?

Ex 7: We can do a cluster analysis on the hidden units. To create the necessary file, select `Probe Selected Units`. The output of this operation give the hidden unit activations for each input pattern. Use this output to create a file `hidden3.inp`, by removing the top two lines and saving it under a new name. Now perform the cluster analysis again using `hidden3.inp` and `auto3.lab`. Are the patterns similarly clustered?

Ex 8: Extra (not to be handed in).
Look at the network in Chapter 7 of Plunkett & Elman. This network attempts to solve the problem of translation invariance: responding to a pattern which may occur anywhere in the input field. Look at the shift project. Try training the network. This should be possible. But now test the performance on unseen examples in `novshift.data`, you can set this in the `Training Options`. Refer to the chapter for more details about this problem, and one solution is given in the `shift2` project.