

31YB: Biologically Inspired Computing:

Lab Sheet 1 (Thurs, 2:00-3:00, 23rd October)

The Stuttgart Neural Net Simulator (SNNS)

Introduction

SNNS includes implementations of a wide range of network architectures and training algorithms. Many example networks and data files for use with SNNS are held in:

`/usr/local/lib/SNNS/examples`

Each example comes with a '.README' file.

SNNS calls data files 'pattern files' and these have the filename extension '.pat'. They contain lots of input patterns along with any corresponding target outputs. Take a look at 'xor.pat' (a data file for the XOR problem) to get the idea.

SNNS creates other files which contain information specifying a network's architecture ('.net') and its configuration for GUI display ('.cfg').

If in difficulty consult the WWW manual which is available from:

<http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html>

Running snns: The XOR problem

To run the UNIX based simulator, run Exceed from Start->Programs->Exceed and log on using your username and password. Then simply type: snns in the xterm window.

An snns banner window will appear and disappear when you click in it. This will leave you with the snns manager window.

First, we'll build a simple 2-2-1 feed-forward MLP network and use it to learn the XOR problem.

<u>In Window</u>	<u>Select Button</u>	<u>Notes</u>
snns-manager	select BIGNET FeedForward	<i>we want a feed-forward network</i>
BigNet	click POSition and see how Rel. Position varies set POS to 'right'	<i>left/right/below Arrange layers from left to right</i>
	click TYPE and see how Edit Plane Type varies set TYPE to 'input' enter units in x-direction 1 enter units in y-direction 2	<i>input/hidden/output input layer .. 2 input units</i>

select ENTER

set TYPE to 'hidden'
enter units in x-direction 1
enter units in y-direction 2
select ENTER

*hidden layer ...
2 hidden units*

set TYPE to 'output'
enter units in x-direction 1
enter units in y-direction 1
select ENTER

*output layer ...
1 output unit*

In lower sub-window under 'Edit Link' select:

Source 1 Target 2

Select ENTER

Select FULL CONNECTION

Fully connects input layer to hidden layer

Select FULL CONNECTION

Fully connects hidden layer to output layer

Select CREATE NET

Confirm YES (if asked)

Select DONE

Close the BigNet window

*You should now visualise the network you have built by selecting **DISPLAY** in the snns-Manager window. Check that it is a fully-connected 2-2-1 network. You can use **SETUP** to turn 'links' ON in order to visualise the connections.*

Save your network architecture as follows:

snns-manager **FILE**

Navigate to your own directory and save
the network as 'myxor' by selecting SAVE

Save network files to your own directory

Now we need some data:

snns-Manager **FILE**

navigate to SNNS/examples -By typing the full path, /usr/local/lib/SNNS/examples , in the
File path bar.

Next click on the PAT (pattern) button, and select the xor file. Then click on the LOAD button
(the xterm window should confirm that the pattern file has been loaded) and finally press
DONE.

Now set up the training algorithm and its parameters using the Control window:

snns-Manager **CONTROL**

snns-Control Cycles 1000
 Select **SHUFFLE**
 PATTERN USE xor

*Train for 1000 epochs
with patterns in random order
Use these data (xor.pat) to train*

LEARN	SEL_FUNC	Std_Backpropagation	<i>Train using back-prop</i>
LEARN	0.9 0.0		<i>Set the learning rate to 0.9</i>
UPDATE	SEL_FUNC	Topological_Order	
INIT	SEL_FUNC	Randomize_Weights	<i>Initialise weights at random</i>
INIT	1.0 -1.0		<i>in the range (-1.0, 1.0)</i>

Now we're finally ready to train the network !

snns-Manager **GRAPH**

Plot the network error

snns-Manager **WEIGHTS**

Show the weights

snns-Control

INIT

Initialise the weights

ALL

Train the network on all patterns

You should see a plot of the network error decreasing and a visualisation of the weight values changing.

Try running the network a few more times by repeatedly selecting:

snns-Control

INIT

ALL

- Note that performance differs each time. **Why is this ?**
- The graph can plot the sum-squared error (SSE) or the mean-squared error (MSE). Try both.
- Now verify that the network has learned the XOR data set by selecting **TEST** on the snns-control panel.
- Selecting **TEST** several times will cycle through the data set. Check that the network outputs are correct by observing the snns-display window.

Questions

- (1) Have a look at the *xor.pat* file (in /usr/local/lib/SNNS/examples) which for this example, is used for both training *and* testing the network, and make sure you understand it. You can view the *xor.pat* file by copying it from the above directory to your home directory (type in your *xterm*:

```
cp /usr/local/lib/SNNS/examples/xor.pat /home/your_username
```

) and then open the copied file using the TextPad Editor in Windows (Start->Programs->TextPad).
- (2) How does the learning rate affect training ? What is a good value ? (Try a range of values like 0.01, 0.1, 1.0, 10.0, 100.0 and run the net several times for each choice of learning rate, being sure to initialise the weights each time (**INIT**))
- (3) Having decided upon a learning rate, train the net again. What is the value of the weight connecting unit 1 to unit 4 ?
- (4) Could a network *without* a hidden layer (i.e. 2 inputs connected to 1 output unit) learn this XOR data set ?

Next question is optional (for your interest!)

- (5) Now build and train such a single-layer network and verify your answer to question 3.

Any comments, suggestions, please email: Dr. A. Hussain a.hussain@cs.stir.ac.uk