DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

COMPSYS726

Using the PyBrain Package to Simulate MLP Networks

Assignment 2 – George Coghill (g.coghill@auckland.ac.nz)

Introduction

This assignment aims to investigate the properties of supplied data sets. We will be using the PyBrain Neural Network Simulator for this purpose. PyBrain was written in the Python Programming language and supports a small number of Artificial Neural Network models. The problem is that only experienced Python programmers may use the system as it stands. The tool supplies a number of functionalities (Modules) that may be embedded in the user's implementation. Here we provide a ready made program with the functionalities needed to successfully complete the assignment. Your goals are:-

- to obtain a good assignment grade
- to learn about the nature of Artificial Neural Networks as part of your examination preparation
- to assess whether Backpropagation is a useful tool for analysing the given data sets
- to give feedback on PyBrain to your lecturer, in order to further our knowledge of this software and to evaluate whether, or not it is likely to be worth pursuing as a research tool in the future

Our intention here is not to burden students who have no knowledge of Python with any significant extra load. The plan is to enable students to investigate the nature of ANN's with as little distraction as possible. The GUI may prove to be a little delicate in terms of its use and so, please treat it gently (any feedback is welcome - see email address above).

Resources Required to perform the Assignment

If you wish to run this software on your own system, try http://pybrain.org for further information. You can run the package on either Windows, or Linux, although we shall be using remote Linux within the Department. To run the software yourselves, you will need the following resources :-

- a Python Interpreter with Tkinter (for the GUI)
- the PyBrain module
- the MatPlotLib package (in particular pylab), SciPy, Numpy and a specific toolset package
- the Assignment2012Part2.py python program
- irisN.dat and other datasets

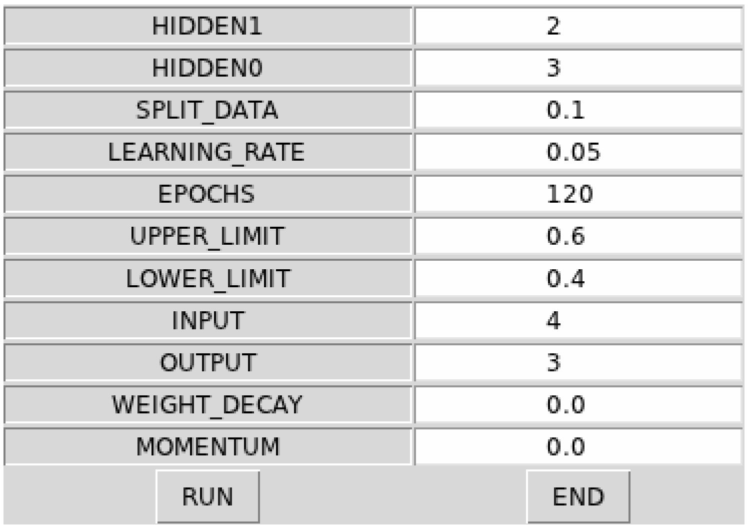
I have done this on my desktop at home, but I'm not an expert on such tasks. I have put together a sheet that may help you perform an installation on your own computer (in the PyBrainDownloads.zip file on CECIL with Assignment2012Part2.py and the data sets). This was written last year and so I may have to modify it if it is not adequate.

The AssignmentPart2.py Program

In this section, I shall try to describe how to use this program to perform a Backpropagation simulation session using our Remote Ubuntu Linux. You can just execute the program and it should run as it stands. The command line should be:

```
python Assignment2012Part2.py irisN.dat
```

This will result in the following window being displayed:

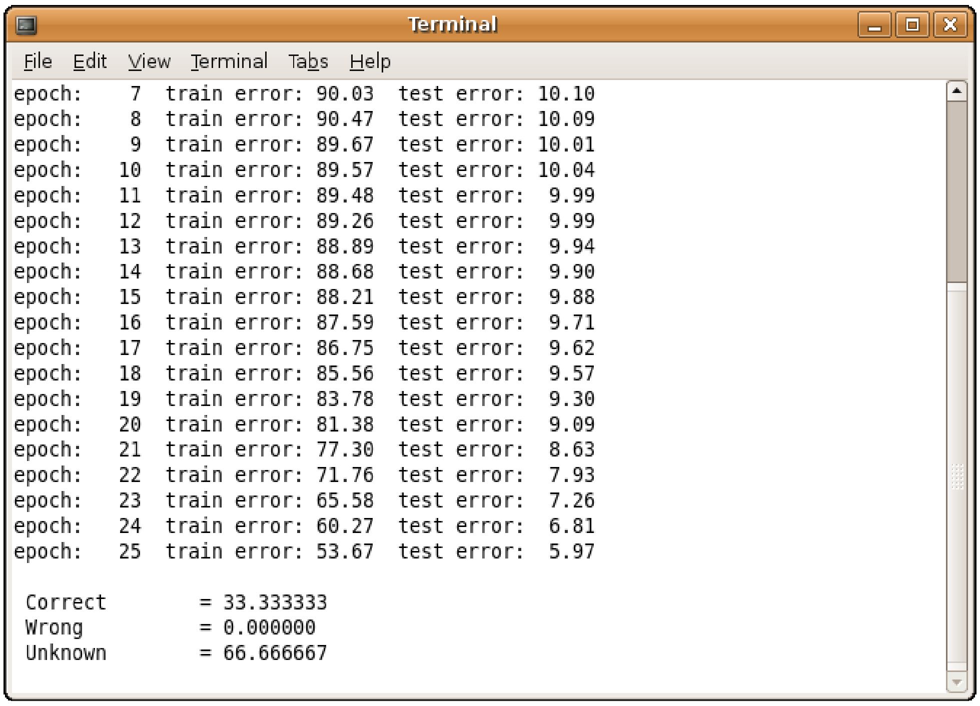


HIDDEN1	2
HIDDEN0	3
SPLIT_DATA	0.1
LEARNING_RATE	0.05
EPOCHS	120
UPPER_LIMIT	0.6
LOWER_LIMIT	0.4
INPUT	4
OUTPUT	3
WEIGHT_DECAY	0.0
MOMENTUM	0.0

Note that the 'EPOCHS' is set to 120 (above) - you will probably require a different value. As an exercise in using the GUI, when you run the simulator for the first time, try changing the displayed value of 120 at the start, to 25:

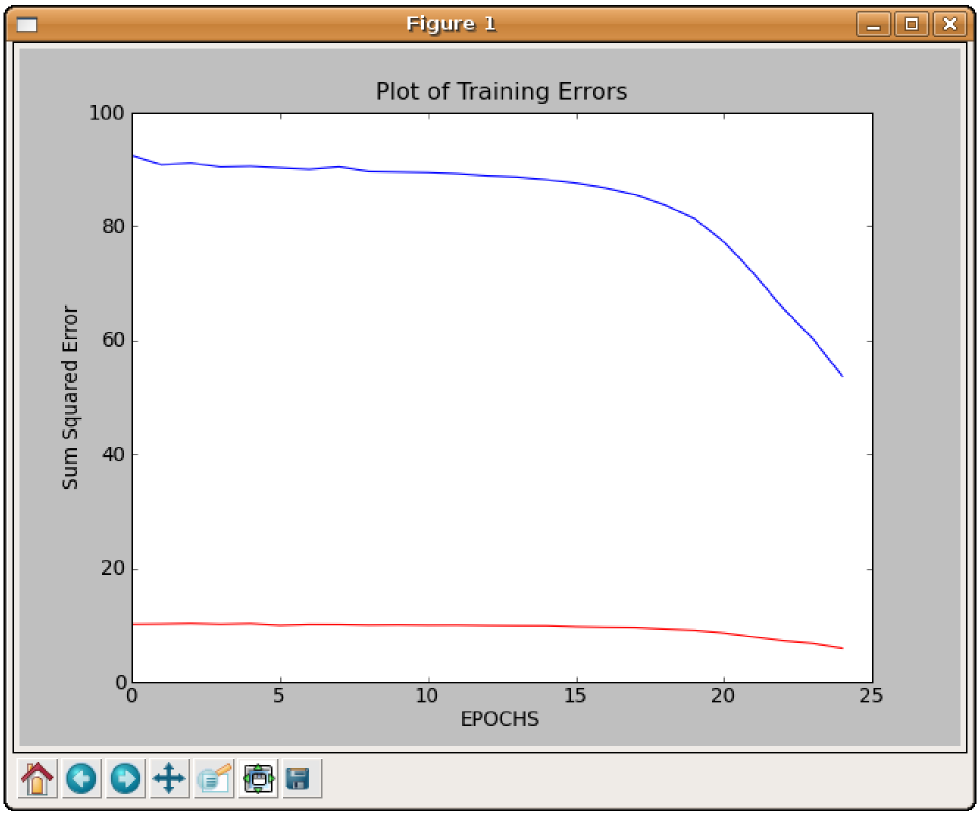
Left Click on the box containing 120. The value should disappear and the cursor position will move to the left hand side of the box. Type in 25 and press the 'Return' key. The new value should move to the centre of the box and you are done.
If you click on the 'RUN' button the following information will be displayed:

- The dataset filename given as the third argument in the command line above
- The epoch number, training error and testing error for each of the default 25 epochs
- A list of comparisons between the actual output test vectors at the end of training, with the desired output value – will be saved to a file called 'result.dat'. If you wish to keep this particular file instance, you must rename it (in Linux) before you run the simulator again
- The percentage, Correct, Wrong and Unknown statistics for the test set are printed to the screen
- A plot showing the Sum Square Error for the training data (blue) and test data validation curve (red) for the 25 epochs is given (below):



```
epoch:    7  train error: 90.03  test error: 10.10
epoch:    8  train error: 90.47  test error: 10.09
epoch:    9  train error: 89.67  test error: 10.01
epoch:   10  train error: 89.57  test error: 10.04
epoch:   11  train error: 89.48  test error:  9.99
epoch:   12  train error: 89.26  test error:  9.99
epoch:   13  train error: 88.89  test error:  9.94
epoch:   14  train error: 88.68  test error:  9.90
epoch:   15  train error: 88.21  test error:  9.88
epoch:   16  train error: 87.59  test error:  9.71
epoch:   17  train error: 86.75  test error:  9.62
epoch:   18  train error: 85.56  test error:  9.57
epoch:   19  train error: 83.78  test error:  9.30
epoch:   20  train error: 81.38  test error:  9.09
epoch:   21  train error: 77.30  test error:  8.63
epoch:   22  train error: 71.76  test error:  7.93
epoch:   23  train error: 65.58  test error:  7.26
epoch:   24  train error: 60.27  test error:  6.81
epoch:   25  train error: 53.67  test error:  5.97

 Correct        = 33.333333
 Wrong          = 0.000000
 Unknown        = 66.666667
```



The Plot

The plot may be saved by clicking on the rightmost icon at the bottom. If you wish to plot more than one run on the same graph, just run the simulator again without killing the graph display (by clicking on X in top right corner). Remember to keep the number of EPOCHS constant when you repeat a ‘RUN’. In general, you may save displays for your report by using the “screen shot” *accessory* from the Ubuntu menu.

Parameter Settings

The parameters displayed by the GUI above are default values and would be suitable for training the Iris Data Set. However, they will change depending upon your requirements for experimenting with various Data Sets and Neural Network architectures. The purpose and usage of these parameters is given below.

HIDDEN1 (2)

This specifies the number of neurons in the second hidden layer moving from the input. If this is set to zero (0), there will only be one hidden layer in the network, i.e., HIDDEN0.

HIDDEN0 (3)

This specifies the Number of Hidden Neurons in first layer from input. This should **not** be set to zero as it would mean that there would be no hidden layers.

SPLIT_DATA (0.1)

The portion of dataset reserved for testing. In this case it is equivalent to 10% of the original data set (randomly sampled).

LEARNING_RATE (0.05)

This usually lies between 0 and 1. The higher it is the larger the weight change increment will be during training.

EPOCHS(120)

Number of training cycles used. We try to find the fewest number of cycles to minimize the test set error. A cycle or epoch is one presentation of the training data set.

LOWER_LIMIT (0.4)

When we look at the actual test output of a neuron in the output layer, its value will be between zero and one. The desired values are typically set to zero or one. We have to decide how small an output can be before it may be interpreted as zero. The ‘LOWER_LIMIT value determines the threshold below which values are taken to be zero.

INPUT (4)

Number of input dimensions (in the iris data set this equates to 4)

UPPER_LIMIT (0.6)

When we look at the actual test output of a neuron in the output layer, its value will be between zero and one. The desired values are typically set to zero or one. As for the lower limit, we have to decide how large a test value can be before it is interpreted as being close to one.

OUTPUT (3)

Number of output Classes (number of desired outputs in data set – iris.dat)

WEIGHT_DECAY (0.0)

If used, small positive values will have the effect of reducing the weight values over time (through the training session).

MOMENTUM (0.0)
If used, this usually lies between 0 and 1. It increases the learning rate at the start of the simulation but reduces with time.

The values of the INPUT and OUTPUT parameters depend upon the dimensions of the data set that you are using (see http://www.ics.uci.edu/~mlearn/MLSummary.html for the original datasets - iris, mushroom, faults and blood). You will have to set these up correctly before launching the simulator. Most of the other parameters you may experiment with, to try to evaluate their effects with regards to my earlier descriptions of MLP behaviour. However, in the default set up above, there are two hidden layers. You should try some network configurations that involve only one hidden layer. Frequently, a single hidden layer network will achieve an acceptable accuracy and with a good generalization level more quickly than a two layer network. However, you should check both.

By varying network as described above you should be able to gain some insights into the behaviour of Backpropagation networks in general.

You should experiment with the irisN.dat data file until you are happy that you understand the operation of the simulator. There are three other data files that are of interest, if I find any more that look interesting, I will prepare a data set that is ready for you to run:

mushroomFixed.dat	(INPUT = 22, OUTPUT = 2)
Faults.dat	(INPUT = 27, OUTPUT = 7)
Blood.dat	(INPUT = 4, OUTPUT = 2)

You may use the mushroom and faults data to try find out more about the simulator behaviour, but for your final report, I would like you to concentrate more on the features of the Blood.dat data set. During the first few week, or two I will spend part of the lecture time trying to offer you more guidance on what I am looking for. I will also try to supply, also, more data sets that are normalised in different ways, e.g.,

FaultsN.dat	input attributes normalised to largest overall value across all attributes
FaultsNbC.dat	input attributes that had highest values greater than one are normalized by the highest value in their respective column

5.0 Results

Your output for this assignment will be an electronically submitted Engineering style report introducing the problem and summarising the various results obtained, including your conclusions. This assignment is fairly open ended. Think about questions like, how many cycles does the network take to learn to correctly categorise the training set? How do changes to the network architecture change the network's behaviour? Try to limit your report to about ten pages. We will discuss the issues more fully in class. This second assignment (the report), is due at the end of week 11, **Friday May 25th**. You may discuss your work with others, but **not the written content of your report**. In the original Course Information handout, it specified that there was an assignment due at the end of week 12. This is incorrect, your only output for the Assignment 2 course work is the report.