NEURAL NETWORKS STARTER KIT

The Neural networks code used to construct this release was developed as an exercise to teach myself about OOP and MS Windows, and to design my own tools for the construction of neural networks.

This release is intended to help people get started, and contains :

1) A working backprop neural network application for windows (SLUG.EXE), together with the resource files (and an example for DOS).
2) The source code for these programs, to illustrate the use of the starter kit.
3) Full documentation on the basic units contained in the starter kit, so that you can see exactly what it contains.

These are shareware. Please distribute these as you see fit. If you find them useful or instructive, please fill in the form below and mail it, together with **$10,** to the above address. Questions can be directed to me via E-mail : CHUCK@PSIPSY.UCT.AC.ZA

THE STARTER KIT is not shareware, costs **$40** for people who register this release ($50 otherwise), and contains the DOS and Windows executables (TPU and TPW for BP7) of the basic unit and the dynamic matrices and vectors unit. You also get the source code for the BPNet unit, which shows how to use the basic units to construct a network object. Included is a Croupier object which allows you to randomly "deal" from a deck of data, and some other small things.

You can pay for SLUG or order the starter kit by phone/fax using a Visa or Mastercard. The phone/fax numbers are (27)-21-788-6613/788-2248. If you want to use E-mail or fax, please use the order form below and remember your phone/fax number and your credit card number. If you use papermail, you must include a bank order drawn on a South African bank.

The Implementation of the basic units, NNUNIT and DYNA2

The basic unit implements a Neuron object and a Neuralnet object, hopefully in a very general way, and defines an underlying paradigm for specific network constructions. Creating a network is easy, as the following code shows:

```
neuralnet.init(incount+hiddencount+outcount+1);
                               {fully connected...}
                               {insert fields}
addfield(inputfield,1,incount);
addfield(hiddenfield,incount+1, incount+hiddencount);
addfield(outputfield,incount+hiddencount+1,count-1);
addfield(offset,count,count);

setfieldsignal(hiddenfield,sigmoid);
setfieldsignal(outputfield,sigmoid);
setfieldsignal(offset,one);

setconnections;
calcallstates;    {essentially switches on offset neuron}
```

```
{--------------------------------}
procedure simpleBPnet.setconnections; {connect feedforward net}
{--------------------------------}
begin
    nofeedback;
    disconnectbetween(inputfield,outputfield);
    disconnectbetween(outputfield,inputfield);
    disconnectbetween(outputfield,hiddenfield);
    disconnectbetween(hiddenfield,inputfield);

    disconnectbetween(offset,inputfield);
    disconnectbetween(inputfield,offset);
    disconnectbetween(hiddenfield,offset);
    disconnectbetween(outputfield,offset);

    disconnect(inputfield);
    disconnect(outputfield);
    disconnect(hiddenfield);

end;
```

It essentially sees a network as a 'bag of neurons', completely connected, but divided into 'neuron fields' (think of these as subsets, not necessarily disjoint) which, for instance, you can use to define neural net layers higher up in the object hierarchy. Neurons can easily change their transfer funtions, and networks can easily change their connectivity. Several neuron transfer funtions are provided, and you can easily add you own.

The Dyna2 unit contains dynamic matrices and vectors, which are used in NNUnit to define connectivity, and generally run things like data presentation and training.

Extensive use is made of the Tcollection object in the implementation (see the included documentation), and all objects are streamable.

<u>HOW TO USE SLUG</u>

SLUG is a front end for a backprop net with 3 layers, using a sigmoid transfer function in the hidden layer and linear transfer functions in the output layer. SLUG uses the steepest descent optimization method (simple backprop). You can set the number of nodes in each layer, and training parameters.
SLUG saves networks on dos streams.

SLUG tries to be very friendly to other Windows apps, and thus slows down a little. Still, you may find that it hogs too much CPU - let me know.

Most of the functionality of the menu is duplicated with buttons on the control panel. This area is divided into two panels, one to report progress (for this release, only the error box and data count box will show change during training) and another to reflect current static training settings. The data count box shows how many times the dataset has been presented.

The **training parameters panel** shows the current training parameters. These are edit controls (they'll be static in the next release), but you can only enter the data through the parameters dialog box.

Most buttons are self explanatory. The **shake button** randomly perturbs the weights slightly. This is useful if you find your net stuck in a local minimum during training. Hit the shake button a couple of times to try to get out. It is also good to do this before training starts.

When you open a logfile, you can first check **the append box** to append training info to an existing logfile. The logfile holds a record of the training progress, and reports the weights matrix and performance of the net. Both data and log files must be open before training can commence.

The **pause button** pauses training, if you wish to do this for some reason. It also records the pause in the log file.

The file menu is used to save and retrieve networks from disk. If a valid network is present, the network icon switches on and the structure of the layers is displayed.
The parameters menu is used to set training parameters. The items presented in the dialog box are:

Learning rate : This parameter effectively determines the step size during the downhill crawl of the error parameter in weight space. Setting this parameter is a black art. All I can tell you is that the larger the network, the smaller this should initially be. Ballpark figure for the 'toy' problem XOR is 0.5. Don't panic if the error sometimes creeps up during training - this happens for various technical reasons, but usually turns around after a while. If it shoots up, you can click the *parameters* button during training and reduce the value. Also remember that the weights are randomized before training starts. In all optimization methods, where you end up depends on where you start, so if your net gets stuck in a local minimum, and shaking it doesn't get you out, you may have success by starting over (i.e. the training sessions are not necessarily repeatable).

Momentum : This is a measure of how much of a previous training step is retained in the current step. Make this betweeen 0 and 1. I usually have it less than 0.9.

Kmod : Unused at present.

Maximum error : This is the convergence criterion. The error is calculated as
SUM(over output layer) (|desired output - current output |)

Maximum iterations : When to give up.

The RUN menu item reads an input file and propagates the data through the net once. This is intended for trained nets.

The structure of the data file :

Training and running data are stored in text files. For the windows app, two ignored lines followed by any number of lines with an input/desired output pair (floating point) on each line. Eg for the XOR problem, the datafile looks like this :

```
test XOR                        ------- line 1 : ignored in SLUG
4 0.5 0.8 0.0 0.1 10000         ------- line 2 : ignored in SLUG
1 1 0                           ------- first IO pair
0 0 0
1 0 1
0 1 1                           ------- last IO pair and last line in the file
```

In the DOS example, the first line is the title of the run, and the second line contains, in order : number of training lines,learning rate,momentum,kmod,maximum error, maximum iterations.

The log file for the above example typically looks like this :

IO MATRIX

```
 1.0000  1.0000  0.0000   ---------- the matrix provided by you
 0.0000  0.0000  0.0000
 1.0000  0.0000  1.0000
 0.0000  1.0000  1.0000

DESIRED MATRIX          ----------- the last column of the IO matrix
 0.0000
 0.0000
 1.0000
 1.0000

INPUT MATRIX
 1.0000  1.0000
 0.0000  0.0000
 1.0000  0.0000
 0.0000  1.0000

Event # 32    0.883410     ------------ the error at intervals

Network response:

inputvec  :    1.00    1.00 response :    0.015
inputvec  :    0.00    0.00 response :    0.003
inputvec  :    1.00    0.00 response :    1.002
inputvec  :    0.00    1.00 response :    1.001

Final Weights

 0.0000  0.0000 -5.2722 -1.4644  0.0000  0.0000
 0.0000  0.0000 -6.7207 -1.4960  0.0000  0.0000
 0.0000  0.0000  0.0000  0.0000 -3.0197  0.0000
 0.0000  0.0000  0.0000  0.0000  3.1998  0.0000
 0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
 0.0000  0.0000  0.7118  1.4773 -0.5769  0.0000
```

---- rows are origin of connections, columns destination, i.e. in row one, one finds that neuron 1 is connected to neurons 3 and 4, and row 5 shows that neuron 5 (the output neuron) does not provide input to anything. The last row is the offset neuron, number 6, which provides inputs to all except the two inputs and itself.

<u>KNOWN BUGS AND HASSLES</u> (TO BE FIXED IN FUTURE RELEASES)

1. **In Slug** :

a) Make text controls which are merely informative static.
b) Perhaps put informative data in a separate dialog window.
c) Icon doesn't show when minimized.
d) Add graphical display of network and network training progress.
e) Add help.

2. **In the starter kit** :

a) Add error checking for failures to allocate things on the heap.
b) Make Croupier object index work with integer, not real dynamic vectors.
c) Priority one : add conjugate gradient training.

d) Work on second derivative training methods.
e) Is it worth it to implement everything using integer arithmetic only?
f) Add automatic data scaling.
g) Maybe get rid of float object - too much memory overhead...
h) Make help files.

Suggestions are not only **very welcome,** but perhaps even ***required.***

---------------------------- CUT HERE ------------------------------

REGISTRATION / ORDER FORM

NAME : ___________________________________

ADDRESS : ___________________________________

____

____

____

PHONE : COUNTRY CODE _____ AREA CODE______ NUMBER ______________

FAX :_______________

E-MAIL :_______________________________________

Mark the correct choices :

____ $10 for registration of SLUG only.

____ $50 for registration of SLUG and the Starter Kit.

____ $40 for the Starter Kit Only (I have registered my copy of SLUG)

____ Bank draft is enclosed.

____ Debit my

____ MASTERCARD NUMBER ________________________________

____ VISA CARD NUMBER ________________________________
