# Building a Simple Back Propagation Network

In the following exercises, we will build and work with a back propagation network that can be used to classify flowers (irises). We will then spend some time examining the network and its parts. Finally we will train and test the network, adjust some parameters, and see how those parameters effect learning.

You should not change program default settings as much as we do in these exercises. The defaults were very carefully chosen to work well on a wide range of problems. For example, while you might want to experiment with different learning rules, it is often less productive to change transfer functions and learning rates.

The specific exercises we'll do are:

1. Build a back propagation network using InstaNet.
2. Reposition network objects in the NeuralWorks window.
3. Save the network to a disk file.
4. Examine the connections and PEs in the network.
5. Hide and show PE connections.
6. Examine layer parameters.
7. View settings that effect the entire network.
8. Set up the training and test files.
9. Look at the instruments created with the Back Propagation command.
10. Train the network on iris classification data.
11. Test the network against a different set of iris data.
12. Randomize and initialize weights in a network.
13. Adjust network learning parameters and check the effect on learning.
14. Use checkpoints to save a network automatically.
15. Retrain the network.
16. Optimize the Iris network.
17. Run the enhanced Iris network.

## Building the Network

In this exercise, we'll use the Back Propagation command to create the back propagation network we'll use throughout this chapter. This network will classify iris species into one of three categories: *Setosa, Versicolor* or *Virginica.* The criteria the network will use for classifying the species are the length and width of the sepals and petals.

The Back Propagation command provides a dialog box you can use to generate back propagation networks by simply selecting parameters. We're using the back propagation network type because it's applicable to a wide range of problems. Also, it contains the basic elements of many more complicated network types, making it a good place to begin.

In typical usage you would carefully alter any default settings; changes here are for illustrative purposes only.

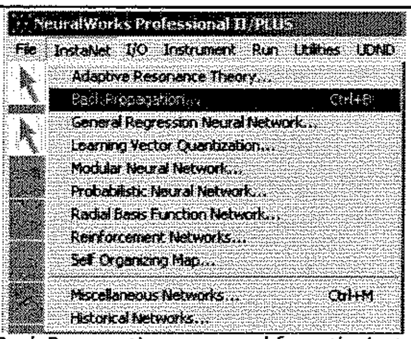
Before beginning this exercise, start NeuralWorks. Refer to the *System Guide* for details.

To create your back propagation network:

1. Select the InstaNet menu. (For versions with pop-up menus, click the right mouse button.)

   For more information concerning using the mouse, see page 15.

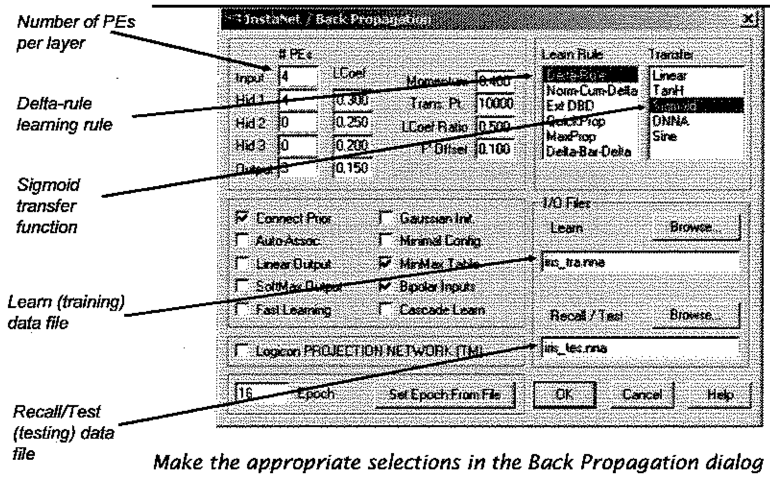
2. Select the Back Propagation command.


NeuralWorks Professional II/PLUS
File InstaNet I/O Instrument Run Utilities UDND
Adaptive Resonance Theory...
Back Propagation... Ctrl+B
General Regression Neural Network...
Learning Vector Quantization...
Modular Neural Network...
Probabilistic Neural Network...
Radial Basis Function Network...
Reinforcement Networks...
Self Organizing Map...
Miscellaneous Networks... Ctrl+M
Historical Networks...
*Selecting the Back Propagation command from the InstaNet menu.*

The Back Propagation dialog box provides a complete tool kit for creating back propagation networks. For now, we'll just use a few of these capabilities and build a very simple network. After we construct and train the network, we'll discuss the capabilities that we used.

3. Enter the following values for the number of processing elements in each layer in the appropriate text entry boxes.

| | |
|---|---|
| 4 | Input |
| 4 | Hid1 |
| 0 | Hid2 |
| 0 | Hid3 |
| 3 | Output |


Number of PEs per layer
Delta-rule learning rule
Sigmoid transfer function
Learn (training) data file
Recall/Test (testing) data file
InstaNet / Back Propagation
Browse...
Browse...
Set Epoch From File
OK
Cancel
Help

| # PEs | | LCoef | |
|---|---|---|---|
| Input | 4 | | |
| Hid 1 | 4 | 0.300 | |
| Hid 2 | 0 | 0.250 | |
| Hid 3 | 0 | 0.200 | |
| Output | 3 | 0.150 | |

Momentum 0.400
Trans. Pt. 10000
LCoef Ratio 0.500
F' Offset 0.100

Learn Rule: Delta-Rule, Norm-Cum-Delta, Ext DBD, QuickProp, MaxProp, Delta-Bar-Delta
Transfer: Linear, TanH, Sigmoid, DNNA, Sine

[✓] Connect Prior
[ ] Auto-Assoc.
[ ] Linear Output
[ ] SoftMax Output
[ ] Fast Learning
[ ] Gaussian Init.
[ ] Minimal Config
[✓] MinMax Table
[✓] Bipolar Inputs
[ ] Cascade Learn
[ ] Logicon PROJECTION NETWORK (TM)

I/O Files
Learn: iris_tra.nna
Recall / Test: iris_tes.nna

16 Epoch

*Make the appropriate selections in the Back Propagation dialog box.*

The number of layers and PEs per layer are important decisions that will not be addressed in detail in this material. However, most back propagation networks will have one or two hidden layers, with the number of PEs in the hidden layers usually falling somewhere between the total number of input and output PEs.

For fully-connected, feed-forward networks with one hidden layer (which describes most back propagation networks), there are some general guidelines for deciding how many PEs should be placed in the hidden layer.

The number of hidden PEs depends primarily on the nature of your data. Data can be broadly categorized as physical or behavioral. Temperature, pressure, and rate of flow measurements, for example, would be considered physical data. A stock market index would be considered behavioral data because it is indirectly a measure of how people behave and is subject to many external unmeasurable circumstances.

The number of hidden units in a back-propagation network that models behavioral data must typically be very small compared to the number of training cases, otherwise the network will learn spurious relationships and will not generalize successfully. A very rough rule of thumb for this is:

$$h = \frac{\#\ of\ training\ cases}{5 \times (m + n)}$$

where:

- *cases* is the number of records in the training file.
- $m$ is the number of PEs in the output layer.
- $n$ is the number of PEs in the input layer.
- $h$ is the number of PEs in the hidden layer.

Another approach with behavioral data is to start with about 5 hidden PEs, use the Run/Save Best option to find the best performance, then vary the number of hidden PEs up or down to improve performance.

Neural network models of physical data do not usually need to be as constrained as behavioral models and do not require so much data, or, equivalently, they can use more hidden PEs. More hidden PEs will allow the network to better reduce the RMS error.

Starting with a network with zero PEs in the hidden layer, we can improve network performance by adding PEs to the hidden layer, but only to a certain point. Beyond some number of PEs in the hidden layer, performance typically begins to decline. We want to test multiple networks with differing numbers of PEs in the hidden layer to help determine an appropriate architecture for the network.

We typically start by building two networks, one with five PEs in the hidden layer, the other with 25 PEs. We then compute their relative performance. If the network with 25 PEs outperforms the one with five, we build a third network with 50 PEs in the hidden layer. At this point we compare the performance of the networks with 25 PEs and 50 PEs in the hidden layer. If the network with 50 PEs in the hidden layer outperforms the network with 25, we would build another network with, say, 100 PEs in the hidden layer. We continue this building and testing cycle until we build a network where the performance on the network with fewer PEs outperforms the network with more PEs.

Once we have determined that adding additional PEs to the hidden layer reduces performance, we start to develop a more precise estimate of the best number of PEs to use. For example, lets assume that the above network with 50 PEs did not perform as well as the network with 25 PEs in the hidden layer. At that time we would start to develop a more precise estimate.

We know that our best performing model is going to have 25 or more PEs in the hidden layer, but less than 50. So, we might start by building a network with 35 PEs in the hidden layer and comparing its performance with the 25 PE network. If the 25 PE network is still better we might build another network with 30 PEs in the hidden layer. If the 35 PE network is better, we might build another network with 40 or 45 PEs. We continue splitting the difference between our best performing network to date, and the next larger tested network.

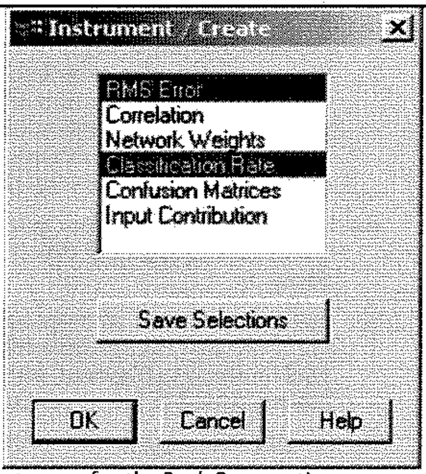
For networks modeling behavioral systems, testing to the nearest five PEs is probably sufficient. You might want to get more precision when modeling physical systems. This lack of precision in behavioral model takes into consideration the noise and inconsistencies that are often inherent to behavioral data.

4. Select the Delta-Rule learning rule in the Learn scroll window. This learning rule is rarely used in real applications.
5. Select the Sigmoid transfer function in the Transfer scroll window. Better networks are often obtained using TanH instead of Sigmoid, but we will first look at a Sigmoid function (a later exercise will try TanH).
6. Specify the Learn source as the file iris_tra.nna by using the Browse button next to Learn. The product remembers the last directory you've accessed, so you may need to Browse to the Professional II/PLUS installation directory (typically C:\Program Files\NeuralWare\NeuralWorks\Professional) to locate our data file.
7. Set the Recall/Test file name to iris_tes.nna. You can type the name in the text entry box, or you can select that data file through the use of the Browse button next to Recall/Test.
8. Note that the default Epoch setting is 16.
9. Once you have made these entries, click the OK button to leave the dialog box and create your network.

10. In the instrument's list box that appears, select the following instruments. Note that RMS Error is already selected (highlighted) by default; be careful that you do not de-select it.

| | |
|---|---|
| RMS Error | This instrument plots the RMS error of the output layer. As learning progresses for this particular network, we should see this graph slowly converge to an error near zero. |
| Classification Rate | This instrument provides a visual indication of the percentage of correct matches. |

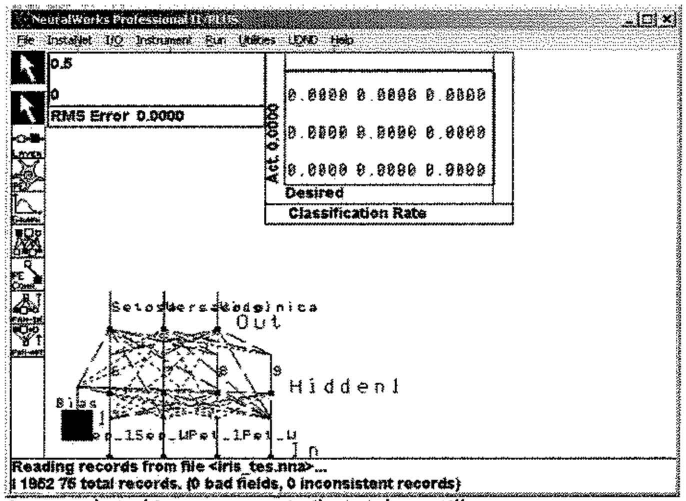
To select an instrument, click on it; to de-select an instrument, click it again.



*The Instrument menu for the Back Propagation command.*

11. Click OK.

NeuralWorks constructs a back-propagation network that should look like the network in the following figure. It also builds two instruments that allow us to monitor the progress of the network. We'll discuss the instruments in detail a bit later. For now, let's look over the network you've just created.



```
Reading records from file <iris_tes.nna>...
1952 75 total records. (0 bad fields, 0 inconsistent records)
```

*A network and instruments similar to these will appear on your screen.*

The *input layer* is at the bottom of the screen, and the *output layer* is at the top. Any layers between the input and output layers are known generically as *hidden layers*. The layers are labeled along the right hand side of the network (although this may be obscured by the instruments we created), and you change these names to anything you like.

Each box within a layer is a processing element (PE). The color, size and fill of the PEs indicate their output values. You can set the colors and fill colors through the Utilities/Display Style command.

PEs in the input and output layers can display field names, and in this example the names overlay. We will adjust the layers later so that the names are readable.

The dotted and solid lines between PEs are the connections. The color, as well as whether the line is solid or dotted, is an indication of a connection's strength. These parameters can be set through the Utilities/Display Style command.

## Repositioning Network Objects

We'll now position the network objects (the instruments and the network itself) to more convenient places on the screen. We'll be doing the following in this exercise:

❖ Moving the instruments to the side of the NeuralWorks window.

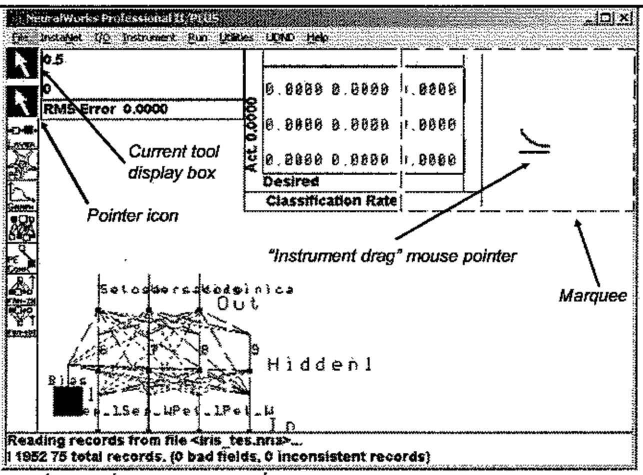
❖ Resizing the instruments (this is optional, but we've included this to give you a feel for how this works).

❖ Moving the network up a bit from the lower left corner.

To drag the instruments up within the NeuralWorks window:

1. Make sure the mouse pointer is in pointer (menu) mode. In other words, the mouse pointer is in the shape of an arrow head, and the current tool display box shows the pointer. If it isn't in pointer mode, click on the pointer icon at the top of the object palette.
2. Move the mouse pointer over the instrument you would like to move.
3. Press and hold the mouse button (the left button if you have more the one).

   The mouse pointer will change shape to a small graph symbol (the *instrument drag* mouse pointer) and an outline (called a *marquee*) will appear around the instrument as soon as you start to drag the instrument with your mouse.
4. Drag the marquee to the desired location, i.e., toward the right and down.
5. Release the button to drop the instrument into place.



Reading records from file <iris_tes.nna>...
1 1952 75 total records. (0 bad fields, 0 inconsistent records)

*Dragging an instrument on the screen.*

You can also make instruments larger or smaller if you like. To resize an instrument:

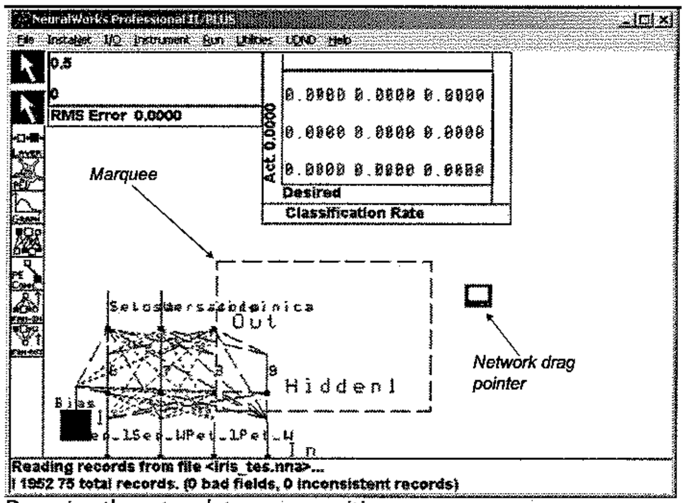
1. Move the mouse pointer over the corner of the instrument you would like to resize. (If you move the mouse pointer outside of the instrument, your mouse will drag the network and not the instrument.)
2. Press and hold the mouse button.

   The mouse pointer will change to the *instrument size* pointer (this looks like a small corner) and the marquee will appear around the instrument.
3. Drag the corner away from the instrument to stretch it, or drag the corner into the instrument to shrink it.
4. When the size is to your liking, release the mouse button.

   An instrument does not always resize exactly as you specified with your mouse, because NeuralWorks makes some adjustments to maintain the uniformity of the graph.

We'll also move our network up and to the right a bit. To move the network:

1. Move the mouse pointer well away from the network, then press and hold the mouse button. The mouse pointer changes to the *network drag* pointer.



Reading records from file <iris_tes.nna>...
| 1952 75 total records. (0 bad fields, 0 inconsistent records)

*Dragging the network to a new position.*

2. While holding the button, drag the network marquee to the desired position.
   - If you were far enough away from all the objects in the network, a dotted line (marquee) appears around the network, and the pointer changes to a screen symbol.
   - If you are too close to a layer within the network when you press the mouse button, only the layer will be outlined for movement. **Do not change the vertical order of layers on the screen or you will disrupt network function.** Release the mouse button, then move the pointer farther away and click again.
3. Release the mouse button and the network will drop into place.

## Saving Your Network

Next we'll save the network you've created, so that you can stop after any of these exercises and have saved a copy of your network to use when you want to proceed. NeuralWorks has two commands for saving files: Save and Save As. Save is generally used to update a network file that you've previously saved. Save As is used to initially save a network, or save the current version of the network to a new file.

Networks can be saved in three different formats:

- Binary
- ASCII
- Annotated ASCII

Binary networks take up the least amount of room, and contain all the information needed to recreate that network. However, binary networks are computer dependent, which means you can't move your binary networks between different types of computers. For example, you can't move a binary network created on a PC to a SUN. When transporting a network to another type of computer isn't necessary, binary format is the preferred format, since in addition to taking up the least amount of room on your disk, it takes the least amount of time to save and load.

Binary networks may not be compatible across major NeuralWorks releases. Only ASCII networks are guaranteed to be upwardly compatible.

ASCII networks are in an ASCII readable form. ASCII files are computer independent, which means this is the preferred method of transferring networks between different types of computers.

Annotated ASCII files are ASCII files with comments on each line. As such, they take up lots of room on your disk, and are rarely used.

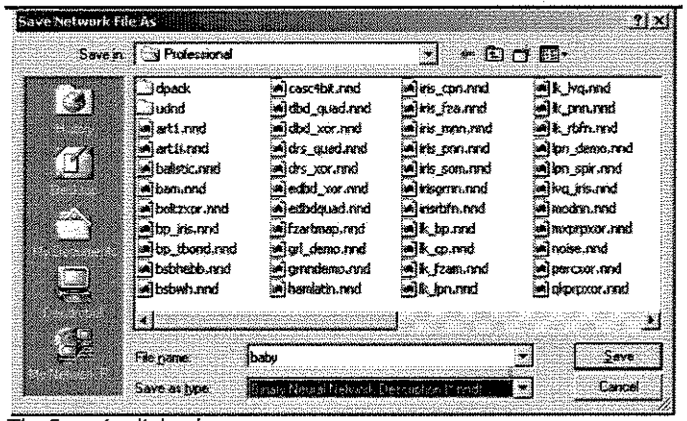
To save your network to disk:

1. Select the Save As command from the File menu.

   A standard Windows file dialog box appears. Notice that the 'Save as type' drop down list provides the ability to save files in Binary, ASCII and Annotated ASCII formats.

2. Type the file name **baby** into the text field. The .nnd extension is assumed for all network types.

   When saving networks, be careful to not overwrite any of our distributed sample networks.



Save Network File As

Save in: Professional

| | | | |
|---|---|---|---|
| dpack | casc4bit.nnd | iris_cpn.nnd | lk_lvq.nnd |
| udnd | dbd_quad.nnd | iris_fza.nnd | lk_pnn.nnd |
| art1.nnd | dbd_xor.nnd | iris_mnn.nnd | lk_rbfn.nnd |
| art1i.nnd | drs_quad.nnd | iris_pnn.nnd | lpn_demo.nnd |
| ballstic.nnd | drs_xor.nnd | iris_som.nnd | lpn_spir.nnd |
| bam.nnd | edbd_xor.nnd | irisgmn.nnd | lvq_iris.nnd |
| boltzxor.nnd | edbdquad.nnd | irisrbfn.nnd | modnn.nnd |
| bp_iris.nnd | fzarbmap.nnd | lk_bp.nnd | mxprpxor.nnd |
| bp_tbond.nnd | grl_demo.nnd | lk_cp.nnd | noise.nnd |
| bsbhebb.nnd | gmndemo.nnd | lk_fzam.nnd | percxor.nnd |
| bsbwh.nnd | hamlatin.nnd | lk_lpn.nnd | qkprpxor.nnd |

File name: baby

Save as type: Binary Neural Network Description (*.nnd)

*The Save As dialog box.*

3. Click OK.

4. Notice the confirmation of the save in the message area at the bottom of the NeuralWorks window.

It is also possible to use the Save command to save a new file. If a name has not been specified for a file (file name of untitled), the Save As dialog box will be displayed automatically when you select Save.

## Examine the PEs and Connections

In this exercise, we'll examine the values and weights related to a particular PE. You'll become familiar with what parameters you can change and get your first experience with the tool palette.

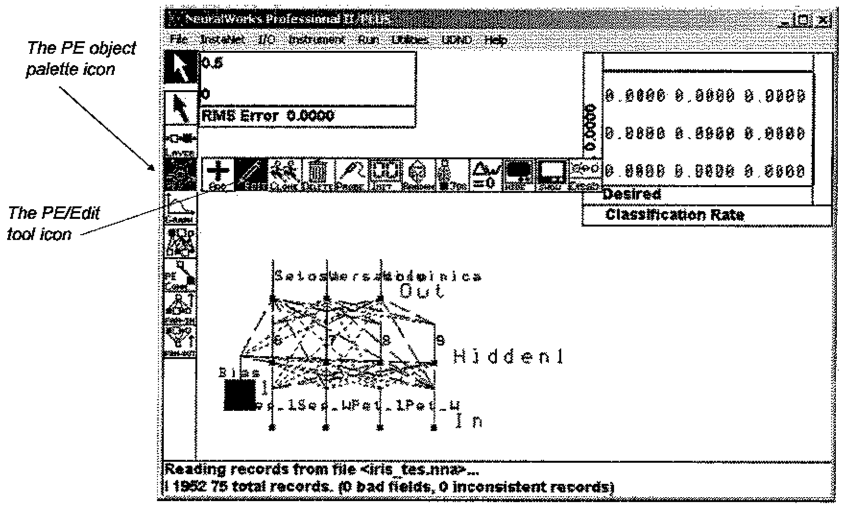
The PE/Edit tool can also be accessed through the following shortcut: simply double-click on any PE.

To examine the values and weights associated with a PE:

1. Click and hold on the PE icon in the object palette.
2. Slide the mouse pointer over the Edit icon in the fly-out tool palette, then release the mouse button. You just selected the PE/Edit tool.

   A number of things just happened.

   - The PE object icon is highlighted to remind you of the object category you are in.
   - The current tool display box now shows which tool you have selected (Edit).
   - The mouse pointer changed shape. In this case, the mouse pointer appears as a circle with an arrow in it, which means the pointer is now a *source selector*. In other words, you must now select an object in the network.
   - The message box displayed a message telling you what to do next (in this case, "Point to a PE to EDIT.").



Reading records from file <iris_tes.nna>...
1952 75 total records. (0 bad fields, 0 inconsistent records)

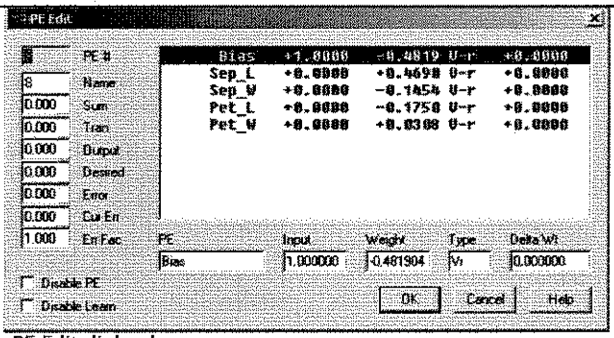
*The PE Edit icon.*

3. Click on any PE in the hidden layer.

   The PE/Edit dialog box appears.

## Quick Tour of the PE/Edit Dialog Box

You are now looking at the PE/Edit dialog box. The values of the fields for the PE are listed along the left side of the dialog box. You can edit these fields (however, for the moment leave them alone). Some of the fields have different meanings for different network types, and for different phases of learning and recall.



| | |
|---|---|
| PE # | |
| Name | 8 |
| Sum | 0.000 |
| Tran | 0.000 |
| Output | 0.000 |
| Desired | 0.000 |
| Error | 0.000 |
| Cur Err | 0.000 |
| Err Fac | 1.000 |

```
Bias   +1.0000  -0.4819 V-r  +0.0000
Sep_L  +0.0000  +0.4698 V-r  +0.0000
Sep_W  +0.0000  -0.1454 V-r  +0.0000
Pet_L  +0.0000  -0.1750 V-r  +0.0000
Pet_W  +0.0000  +0.0308 V-r  +0.0000
```

| PE | Input | Weight | Type | Delta Wt |
|---|---|---|---|---|
| Bias | 1.000000 | -0.481904 | Vr | 0.000000 |

☐ Disable PE
☐ Disable Learn

*The PE Edit dialog box.*

For back propagation, the fields are usually defined as:

| | |
|---|---|
| PE # | Each PE is assigned a unique location in memory, and these locations are assigned numbers starting with 1 and extending to the maximum allowable number of PEs. |
| Name | The PE name, which defaults to the PE number unless you change this. |
| Sum | The value of the summation |
| Tran | The output value of the transfer function |
| Output | The output value of the PE |
| Desired | The desired output value for the PE |
| Error | The back-propagated error |
| Cur Err | The raw error field |
| Err Fac | A scaling factor for the error |

For a full description of the PE fields, see the *Reference Guide*.

You can also disable the PE (freeze its output), which prevents weight updates for the PE during the learning phase. On the right side of the dialog box, you'll see all the connections belonging to that PE in a scroll window. Connections that "belong" to a PE are those connections that provide input to that PE. The source PE for the connection, current input, weight, weight type, scope, and delta weight are shown for each incoming connection.

The following weight types exist:

| | |
|---|---|
| V | Variable weight (this type of weight can change during training, and therefore learn). |
| F | Fixed weight (doesn't change during learning). |
| M | Mod weight (a fixed weight that is modified with a Mod factor). |
| S | Set weight (similar to a mod weight, except that it is multiplied by the input clamp value). Also, under certain circumstances the control strategy can treat this type of weight as a special class of input connections. Refer to the Reference Guide for a complete explanation. |
| D | Disabled |

Weight types in a back-propagation network are usually all of the Variable type.

In addition, a connection can have a scope, which refers to how it is specified in relation to PEs. The choices are:

| | |
|---|---|
| a | Absolute |
| r | Relative |
| l | Local |

Scope is only of importance when you are doing advanced editing of the network architecture using the clone tools. By default, all weights are relative.

These weight types and connection scopes are only important when working with more complex networks. Weight types and connection scope is fully explained in the Reference Guide.

The information available within a PE can be quite valuable when trying to diagnose network problems. However, neural networks are composed of many PEs, and in a bit we will see a way to look at the values of many PEs at the same time.

Also note that there are text entry fields below the connections scroll window. Selecting an entry in the scroll window allows you to modify that entry.

Leave the PE dialog box by clicking on Cancel. Cancel closes the dialog box but doesn't implement any changes you may have made (with the exception of changes to the connection weight type and scope).

## View the Connections for a Particular PE

In networks with lots of connections, it's good to see how an individual PE or layer of PEs is connected to the rest of the network. Also, in fully connected networks (such as back propagation where there are large numbers of connections), it's common to show only enough connections for the scope of the network to be represented in a clear fashion.

NeuralWorks allows you to view connections selectively. In this exercise, we'll use the Hide Connections command to hide all the connections in the network, and then use the tool palette to examine the connections in our back-propagation network more closely.

1. From the Utilities menu, select the Hide Connections command.

   All the Connections in the network will now be hidden (not deleted); i.e., the dotted or solid lines will not be displayed.

2. Click and hold on the PE icon in the object palette.

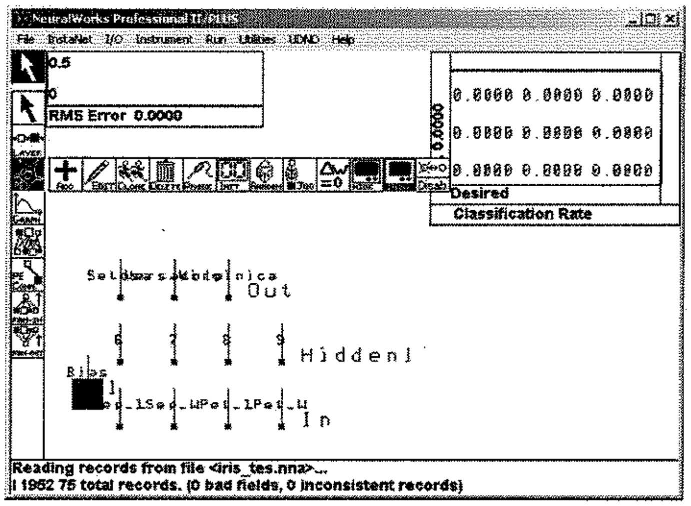
   The fly-out tool palette for the PE object type will appear.

3. Slide the mouse pointer over the Show Connections tool and release the mouse button.

4. Click on the left-most PE in the input layer.

   A dialog box will appear, where you can select which connections to show.

5. Click the Output Connections radio button; then click OK.


NeuralWorks Professional II/PLUS
File InstaNet I/O Instrument Run Utilities UDND Help
Desired
Classification Rate
Out
Hidden1
Bias
In

```
0.5
0
RMS Error 0.0000

0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
0.0000 0.0000 0.0000

Reading records from file <iris_tes.nna>...
I 1952 75 total records. (0 bad fields, 0 inconsistent records)
```

*Selecting the PE/Show Connections tool*

All of the output connections for that PE will be shown.

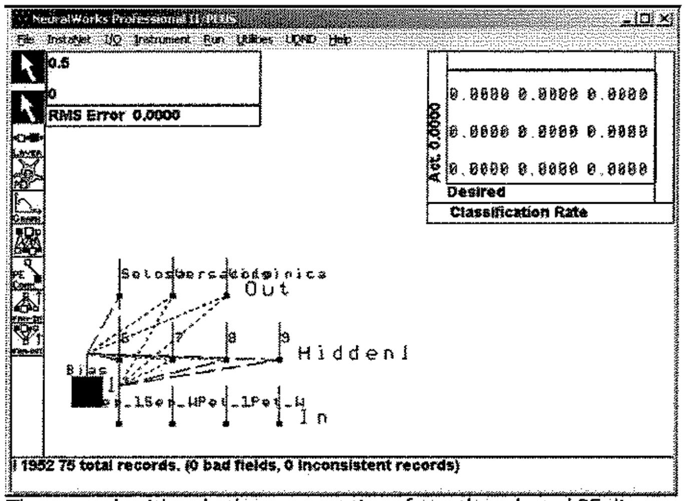
Since the Show Connections tool is still active, you can continue to click on PEs and be prompted for which connections to show.

1. Click on the leftmost PE visible on the screen (the one labeled Bias).
2. From the dialog box, click the Output Connections radio button.

   This shows all the outgoing connections for this PE.
3. Click on the Pointer icon in the object palette to drop the Show Connections tool.

   NeuralWorks will return to pointer (menu) mode.

Using the Show and Hide connections tools, you can produce an uncluttered display showing the flow of data from the input of a fully-connected back-propagation network. Each fly-out tool palette in the object palette has its own Show and Hide connections tools.


NeuralWorks Professional II/PLUS
File InstaNet I/O Instrument Run Utilities UDND Help
0.5
0
RMS Error 0.0000
Act. 0.0000
Desired
Classification Rate
Out
Hidden1
Bias
In

```
0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
```

```
1952 75 total records. (0 bad fields, 0 inconsistent records)
```

*The network with only those connections from the selected PE displayed.*

## Examine the Layer Parameters

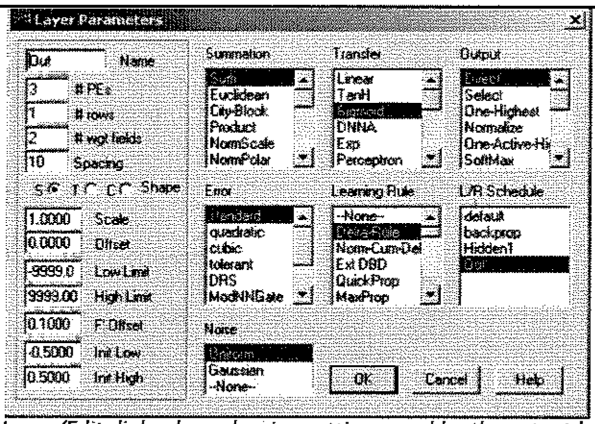
The math functions and learning rules for the PEs in a network are chosen on a layer basis.

To view the layer parameters:

1. Click and hold on the Layer icon in the object palette. The fly-out tool palette for the layer object will appear.
2. Slide the mouse pointer over the Edit icon and release to select the Edit tool.
3. Click on any PE in the output layer. The Layer Edit dialog box appears.
4. Click Cancel when you finish looking at the dialog box.

Make the PE names more readable:

5. Again select the Layer Edit icon (repeat steps 1 and 2).
6. Click on any PE in the input layer.
7. On the left side of the Layer Edit dialog, change the 'Spacing' entry from 5 to 8, then click OK.
8. Click on the hidden layer and also change its spacing from 5 to 8, then click OK.
9. Click on the output layer and change its spacing from 5 to 10 to show the longer output names, then click OK.
10. Press the Escape key to return to pointer mode.



Name: Out
# PEs: 3
# rows: 1
# wgt fields: 2
Spacing: 10
Shape: S [✓] T C
Scale: 1.0000
Offset: 0.0000
Low Limit: -9999.0
High Limit: 9999.00
F' Offset: 0.1000
Init Low: -0.5000
Init High: 0.5000

Summation: [✓ Sum] Euclidean City-Block Product NormScale NormPolar
Transfer: Linear TanH [✓ Sigmoid] DNNA Exp Perceptron
Output: [✓ Direct] Select One-Highest Normalize One-Active-Hi SoftMax
Error: [✓ Standard] quadratic cubic tolerant DRS ModNNGate
Learning Rule: --None-- [✓ Delta-Rule] Norm-Cum-Del Ext DBD QuickProp MaxProp
L/R Schedule: default backprop Hidden1 [✓ Out]
Noise: [✓ Uniform] Gaussian --None--

*The Layer/Edit dialog box, showing settings used by the output layer.*

## Quick Tour of the Layer/Edit Dialog Box

The layer Name is shown in the upper left-hand corner of the dialog box. Below the Name is the field for the number of PEs in the layer.

The # rows (number of rows) text entry box allows you to control how many rows on the screen the layer will be divided into during display. This allows a layer with a two-dimensional layout (such as a biological center-surround array of neurons) to be graphically shown.

The # wgt fields (number of weights fields) entry assigns the necessary memory to each connection in the layer. This number is normally set to one, two or three depending on the learning rule, to allow for the current weight value of the connection—as well as other values such as an accumulated weight value, and delta weight. For network types built with User-Defined Neuro-Dynamics, you can specify the number of weight fields without restriction.

Some learning rules like Delta-Bar-Delta require many more weight fields per connection. Once you have created a network, you can't change the number of weight fields, and the Layer/Edit dialog box will only display those learning functions which are valid for the given number of weight fields.

The Spacing field specifies how close together the PEs in the layer will appear on the screen.

The seven scroll windows in the main body of the dialog box allow you to choose from a wide range of math functions and learning rules. Scroll windows are available for:

| | |
|---|---|
| Summation | The method of accumulating input; usually the sum of the inputs multiplied by the connection weights (Sum). |

| | |
|---|---|
| Transfer | The method of transforming the input; usually some type of non-linearity, such as the Hyperbolic tangent function (TanH) or sigmoid. We encourage you to try TanH for most real-world, back propagation applications, and we'll work with TanH in a later exercise. |
| Output | The function that decides who the winner will be in competitive layers. For back propagation, usually set to Direct (no competition). |
| Error | The function that can further transform the error, such as squaring it or cubing it. This field may also contain the name of an error table, which can apply a specific error transformation based on the actual and desired output for the PEs in the layer. This is described in the Reference Guide. |
| Learning Rule | The learning rule used for the weight updates. For most back propagation, the Normalized-Cumulative-Delta rule (Norm-Cum-Delta) is a common choice, and we'll illustrate its use in a later exercise. |
| L/R Schedule | The name of the file that holds many of the adjustable parameters that will be used by the network. |
| Noise | Noise adds a random value to a PE's summation value before a transfer function is called. Noise functions can be added on a layer-by-layer basis through the Layer/Add and Layer/Edit tool dialog boxes. You can also specify a noise function to be applied to all layers through the Back Propagation command dialog box. There are three noise functions: Uniform, Gaussian and None (and these are explained in detail in the Reference Guide). |

Other fields in the layer dialog box are:

| | |
|---|---|
| Shape | The shape of the PEs in the layer can be all circles, triangles or squares. |
| Scale and Offset | Used for transferring data from one range to another. |
| Low and High Limit | Sets a minimum and maximum value for the PEs in the layer. |

| | |
|---|---|
| F′ Offset | A parameter that prevents the learning process from saturating in back propagation networks. |
| Init Low and High | Layer specific weight randomization limits used during initialization. |

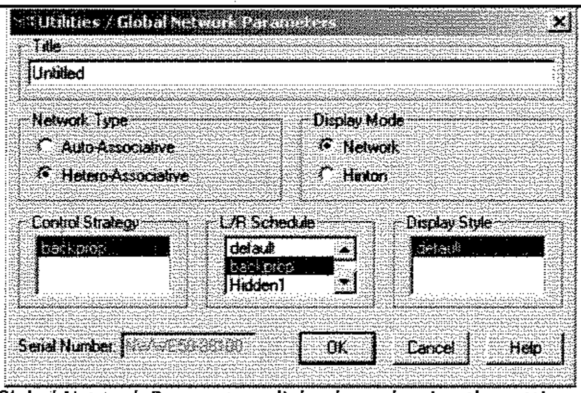
You can scroll through the list of available functions and learning rules to get a feel for the wide ranges of choices available. However, you cannot choose the functions indiscriminately. Some functions are general in nature, but some are designed to be used only with specific network types.

Leave the dialog box unchanged by clicking on Cancel.

## Viewing Global Settings

In this exercise, we'll take a brief look at some of the global settings for the network we've created. Global settings apply throughout the network, not on a layer by layer basis. We'll deal with these settings individually in later chapters. To view these settings:

1. From the Utilities menu, choose the Global Network Parameters command.


Utilities / Global Network Parameters

Title: Untitled

Network Type: Auto-Associative [✓ Hetero-Associative]

Display Mode: [✓ Network] Hinton

Control Strategy: backprop

L/R Schedule: default, backprop, Hidden1 (selected: backprop)

Display Style: default

Serial Number: NeVVE50-38100

*The Global Network Parameters dialog box, showing the settings for our example network.*

The network parameters dialog box shows that the network is of the Hetero-Associative type (input and output need not be the same).

The display mode of the network is Network mode. We will look at a network in Hinton mode later in this tutorial.

The Control Strategy, which was chosen by the Back Propagation command, is correctly set to Backprop. The Control Strategy controls the flow of processing through the network.

The L/R Schedule (learn/recall schedule) is where many parameters and coefficients are stored.

Finally, the Display Style controls the colors and sizing of objects on the display.

2. Leave the dialog box unchanged with Cancel.

## Setting up the Training and Test Files

Now that you've taken a brief look at the network and what makes it "tick," we'll take a look at the data files used to train and test it. When we created the network with the Back Propagation command, we selected the training and testing/recall files. Now we'll discuss these files and set up a few parameters associated with them.

In supervised learning networks (such as back propagation), the inputs are presented along with desired outputs during the training phase. The test phase consists of presenting new inputs to the trained model, along with desired outputs, to see how well the trained network performs. In most cases, you don't want to test and train a network on the same data, since this might only show how well the network memorized the training data. You are interested in creating a network that will generalize to new input situations. For that reason, the data you have available is normally broken down into a training set and a test set.

The training phase can also be called the learning phase.

### A Brief Discussion of Data Sources

Input data for a neural network can be presented from the keyboard, an ASCII or binary file, or from a user-written program. We will begin by discussing the ASCII file format. ASCII input data files contain data in row and column format. Each logical row contains the inputs and (optionally) desired outputs for one example. For instance, if there were 4 inputs, and 3 possible outputs, there would be 7 numbers (or fields) for each logical row. Each number (field) would be separated from the others with at least one space or a comma. This is exactly the case with our flower classification network.

User IO programs are covered in a later chapter.

A logical row can consist of many physical rows. The continuation character is the ampersand (&) and must begin in column 1 of the continuation line.

ASCII data files can have comments. A comment line begins with an exclamation point (!). Comments can also appear at the end of a line, again using the exclamation point as the comment start character.

For example, here is a data set with 4 fields in each logical row:

```
! This is an example of a full-line comment line.
0.2 0.2 0.1 0.2    ! This is an end-of-line comment for a 4 field record.
0.2 0.1 0.0
&0.3               ! An end-of-line comment on a continuation line.
```

ASCII input data files may have an extension of .nna, or .txt, or any other extension, but they must have a file extension ("myfile" would become altered to "myfile.nna").

As mentioned before, we'll use the back propagation network we created to classify iris species into three categories: Setosa, Versicolor, and Virginica. These are the three possible desired outputs for our network. We will represent the desired outputs to the network as either 0 or 1. For example, if the iris is a Versicolor, the desired outputs presented to the network for that example would be: 0 1 0. The classification is based on sepal length and petal length, and on sepal width and petal width, and these are the four inputs to our network. The four inputs for each example will appear as four analog numbers.

Input data for a back propagation network can be integer or real, such as "1", "2", "0.5" and so on.

The data for our flower-classifying back propagation network has been divided into a training (learn) set and a test (recall/test) set. The training file is called iris_tra.nna. There are 75 examples we will be training our network on. The test file is called iris_tes.nna. There are 75 test cases.

## Taking a Look at the Training and Test Data Files

Use your favorite ASCII editor to edit the training file iris_tra.nna and note the format of the data.

On a PC under Windows®, you can use editors like Notepad or Wordpad to view or edit a data file.

Open the file iris_tra.nna using your editor. It resides in the NeuralWorks install directory. The format of this training file is the same as it's associated test file, iris_tes.nna.

Editing a data file that NeuralWorks has opened (with a Run menu command) only changes the copy of the data file on disk, not the data file NeuralWorks has opened. You must close the data file in NeuralWorks before you can edit it and have these changes take effect. Data files are closed in NeuralWorks using the I/O Rewind Input command (see the Reference Guide).

```
! Irist Training Set
!@ Sep_L  Sep_W  Pet_L  Pet_W Setosa Versacolor Virginica
   0.224  0.624  0.067  0.043
&    1.0    0.0    0.0
   0.749  0.502  0.627  0.541
&    0.0    1.0    0.0
   0.557  0.541  0.847  1.000
&    0.0    0.0    1.0
   0.110  0.502  0.051  0.043
&    1.0    0.0    0.0
   0.722  0.459  0.663  0.584
&    0.0    1.0    0.0
   0.776  0.416  0.831  0.831
&    0.0    0.0    1.0
```

A more typical view of the same file is shown below. Few data files use continuation lines as shown above; you may use them if you need to, but they need not be used. The above and below file segments are functionally identical.

```
! Irist Training Set
!@ Sep_L  Sep_W  Pet_L  Pet_W Setosa Versacolor Virginica
   0.224  0.624  0.067  0.043   1.0    0.0    0.0
   0.749  0.502  0.627  0.541   0.0    1.0    0.0
   0.557  0.541  0.847  1.000   0.0    0.0    1.0
   0.110  0.502  0.051  0.043   1.0    0.0    0.0
   0.722  0.459  0.663  0.584   0.0    1.0    0.0
   0.776  0.416  0.831  0.831   0.0    0.0    1.0
```

Close the data file and editor, and return to the Neural-Works application.

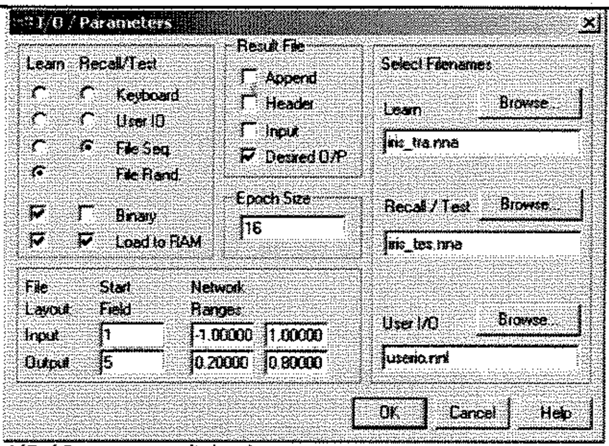
## Working with Input/Output Parameters

Now that you have seen the format of the input files, we'll look at the parameters that affect these data files. These are found in the Parameters dialog box, accessed through the I/O menu.

Short cut: double-click away from the network, over an empty region, to display the I/O / Parameters dialog.

To view these parameters:

1. Select the Parameters command in the I/O menu. The dialog box appears.



I/O / Parameters

Learn Recall/Test
- Learn: File Rand. (selected); Recall/Test: File Seq. (selected)
- Binary: Learn ✓, Recall/Test ☐
- Load to RAM: Learn ✓, Recall/Test ✓

Result File: Append ☐, Header ☐, Input ☐, Desired O/P ✓

Epoch Size: 16

Select Filenames
- Learn: iris_tra.nna
- Recall / Test: iris_tes.nna
- User I/O: userio.nnl

| File Layout | Start Field | Network Ranges | |
|---|---|---|---|
| Input | 1 | -1.00000 | 1.00000 |
| Output | 5 | 0.20000 | 0.80000 |

*The I/O / Parameters dialog box.*

2. Under the Learn heading in the upper left section of the dialog box, click the File Rand. radio button (it may already be selected).

   This tells NeuralWorks that you wish to have the examples presented to the network in a random manner, as opposed to sequentially. Random presentation of training data helps the network to avoid less than optimal solutions (local minimum).

3. Note that the Binary check box for Learn is selected. This option converts your data file into a binary file for speed reasons. We will discuss this option in a later section. For now, it can remain selected.

4. In the Recall/Test column, near the upper left of the dialog box, click the File Seq. radio button.

5. Note that the name of the training file, iris_tra.nna, appears under the Learn heading on the right side of the

dialog box. We selected this as the training file when we built the network using the Back Propagation command.

You could, using the Browse button for Learn, select a different data file.

6. Also note the name of the test file, iris_tes.nna, appearing under the Recall/Test heading on the right side of the dialog box. Just as for the Learn source, you could use the Browse button to scan and select another file.

   The network output will be written to an ASCII results file with a name similar to the Recall/Test file, but with an extension of nnr. For example, running a Test command upon "mydata.nna" produces "mydata_nna.nnr"; "mydata.txt" yields "mydata_txt.nnr".

   You can decide if the Results file will be overwritten or appended to. The results file may also contain, in addition to the network results: header information, the input, and (if running in test mode) the desired output.

7. Change the File Layout Start Fields, if necessary, so that the starting input field is 1, and the starting output field is 5.

8. Record your dialog box choices by clicking OK.

A major issue we have not discussed yet involves the pre-processing of input data to lie in suitable ranges for the network type being used. The data we are using is being automatically pre-processed. In a later chapter we will look at how the MinMax Table, and Network Ranges fields in this dialog box, do this type of pre-processing for us.

The Epoch Size text entry field will also be discussed in a later chapter.

## Looking at the Instruments We Created

One of the keys to training a neural network is in being able to monitor the weights, processing element values and network performance as the network evolves. With this data, you can make informed decisions about the proper network topology, math functions, training times, learning parameters, etc... NeuralWorks has a powerful, flexible and easy-to-use diagnostic facility called instrumentation. Instrumentation sends diagnostic information from a network to the screen, a file or a user-written program. A few commonly used instruments can be added directly from the Back Propagation command and we'll look at those that we created for our iris classifying network.

NeuralWorks also has additional facilities for creating instruments. The EasyProbe tool provides a quick way to create a wide variety of instruments. InstaProbe provides powerful and flexible capabilities for creating and modifying probes and instruments, and you can manually create probes and instruments using the various graph and probe tools and instrument commands.

Almost every network you build will benefit from the use of instrumentation. We highly recommended that you become familiar with instrumentation and EasyProbe, and use instrumentation in all your networks.

Instrumentation is composed of two components:

❖ Probes are the contact points of instrumentation with the network. Probes can be created to gather data from a specific PE, sets of PEs, connections, the network as a whole or almost any combination that you can imagine.

❖ Instruments are the windows into the network that appear on the screen. Instruments get their data from probes.

Using instruments in your network, while recommended, can slow performance. The instruments can be disabled or deleted when you are comfortable with the way a network is training.

## A Close Look at the RMS Error Instrument

First, we'll examine the instrument that the Back Propagation command created to measure the error at the output layer: the RMS Error instrument. Before continuing, make sure that the iris classifying back propagation network we previously created is loaded into NeuralWorks.

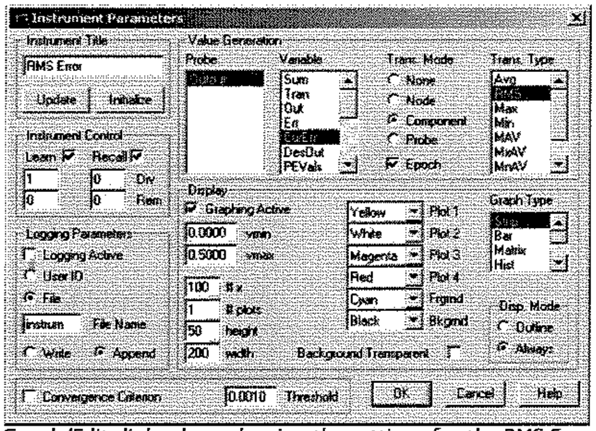
When you selected the RMS Error Graph in the follow-up instruments dialog box to Back Propagation command, you created a probe and instrument to measure the Root Mean Square (RMS) Error for all the PEs in the output layer. The root mean square error adds up the squares of the errors for each PE in the output layer, divides by the number of PEs in the output layer to obtain an average, and then takes the square root of that average. Hence the name root mean square.

The squaring of the errors gets rid of the sign of the error, but increases the magnitude. The square root removes the magnitude increase of the squaring operation. This RMS error is a valuable and common measure of the performance of a network during training. RMS error can also represent the error over many training examples, as we will see in a later section. We will see this instrument in action when we move to the training phase.

To examine the RMS Error instrument:

1. Click and hold on the Graph icon in the object palette.
2. Select the Edit tool.
3. Click on the RMS Error instrument.

Short cut: double-click on the instrument to display the Graph/Edit dialog.


Instrument Parameters

Instrument Title: RMS Error

Update | Initialize

Instrument Control: Learn [✓] Recall [✓]; 1, 0 Div; 0, 0 Rem

Logging Parameters: [ ] Logging Active; ( ) User ID; (•) File; instrum File Name; ( ) Write (•) Append

Value Generation:
Probe: [illegible]
Variable: Sum, Tran, Out, Err, [illegible], DesOut, PEVals
Trans. Mode: ( ) None, ( ) Node, (•) Component, ( ) Probe; [✓] Epoch
Trans. Type: Avg, RMS, Max, Min, MAV, MxAV, MnAV

Display: [✓] Graphing Active
0.0000 ymin
0.5000 ymax
100 # x
1 # plots
50 height
200 width
Yellow Plot 1; White Plot 2; Magenta Plot 3; Red Plot 4; Cyan Frgrnd; Black Bkgrnd
Background Transparent [ ]
Graph Type: Strip, Bar, Matrix, Hist
Disp. Mode: ( ) Outline (•) Always

[ ] Convergence Criterion 0.0010 Threshold

OK | Cancel | Help

*The Graph/Edit dialog box, showing the settings for the RMS Error instrument.*

The parameters of particular interest are:

| | |
|---|---|
| Probe | This is the probe that provides input to the instrument. A probe must be attached to either processing elements or connections (or both). In this case it's attached to all of the PEs in the non-input layers. By setting the # plots value to 1 in the Graph/Edit dialog box, you limit the display to the top or output layer. |
| Variable | This is the type of data that the instrument is monitoring. In this case it's the current error. |
| Trans. Mode/Epoch | An epoch is the number of sets of training data presented to the network (learning cycles) between weight updates. This value is set through the IO/Parameters command. When the epoch check box is selected, values gathered by the probe are combined across the probe component and over the number of cycles set by the epoch. |
| Trans. Type | This defines how the values are combined. In this case the value is the RMS of all values across the probe component (the output layer). |

| | |
|---|---|
| Instrument Control Learn Divisor | Normally, this defines the number of learning cycles between instrument updates. However, when epochs are used this becomes the "epoch divisor." |
| vmin | This is the minimum plot value, and the Back Propagation command has set it to 0 by default. This is also shown on the instrument itself. |
| vmax | This is the maximum plot value, and this has been set to .5. Again, this is shown on the instrument. |
| Convergence Threshold | This is only used by RMS error instruments and is the value that, when reached, is used to stop learning. This must be set to a positive number. The Back Propagation command set this value to 0.001. |
| # X | This is the number of points plotted along the X axis of the strip chart. In this case, there are 100 points (however this value changes from instrument to instrument). |
| # plots | Because the output probe is attached to all of the non-input layers, we need a way to limit the instrument to displaying data from the output layer. We accomplish this by setting the number of plots to 1. If we set it to 2, the hidden layer outputs in the instruments would be displayed as well, because TransMode is set to Component for this probe. A component is a layer, so values are transformed across each layer yielding a "plot" for each layer. |

Now you see how this instrument functions, and the flexibility NeuralWorks provides in using instruments. When we run the iris classification network, the RMS Error instrument will give us a visual indication of just how well the network is learning. For this particular network, the RMS error converging toward zero shows that the network is learning.

Close the dialog box by clicking the Cancel button.

## A Closer Look at the Classification Rate Instrument

The Classification Rate instrument we created with InstaNet/Back Propagation provides a two-dimensional comparison of desired results to actual network predictions. This instrument is appropriate for classification problems where a 1-of-n transformation has been used on the output layer. For other output representations, or for prediction problems, we would choose the Confusion Matrix.

The histogram at the top of the columns represents the desired results. The histogram of the rows represents the actual network predictions. The body of the Classification Rate matrix represents the intersection of desired results and actual network predictions.

A value of 1.0 for any given cell means that all desired outputs for that category were predicted by the network to be a member of that category. Similarly, a value of 0.0 in a cell means that none of the desired outputs were predicted to be a member of that category. The ideal is to develop a network with a value of 1.0 on the diagonal running from lower left to upper right and 0.0 elsewhere.

In a network with one or two outputs, the Classification Rate instrument will be a 2 x 2 matrix. In a network with n output PEs, the Classification Rate instrument will be an n x n matrix. With two or more output PEs represented, the PE with the largest predicted value is considered to be the category predicted. In a network with only one output PE, the Classification Rate instrument determines the category by rounding. Rounding is done at .5 for networks with a Sigmoid transfer function, and at 0 for networks with a TanH transfer function.

## Training the Network

To this point we've:

- ❖ Built a network using InstaNet / Back Propagation.
- ❖ Selected and examined the input files.
- ❖ Created and examined diagnostic tools to monitor the learning phase.

Now we're ready to train the network. But before we do, it's important to talk a bit about learning coefficients.

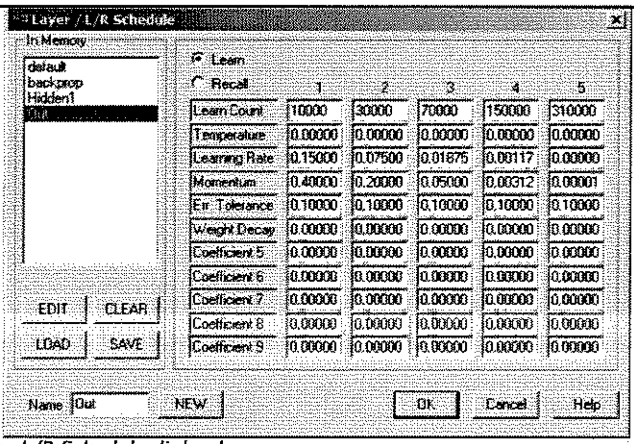
### The Importance of Learning Coefficients

We've only briefly mentioned what is sometimes an important step in training a neural network: setting the learning coefficients. The default learning parameters chosen by the InstaNet/Back Propagation command may need modified, in some cases, to suit the particular problem. These can be set when you use the InstaNet/Back Propagation command to construct a network. We didn't set the learning coefficients earlier so that the following exercises can show you the effects of using the default coefficients, and then the change in learning caused by adjusting the coefficients.

The InstaNet/Back Propagation command sets up Learn and Recall Schedules for each layer that learns.

If you want to view the learning parameters for the output layer:

1. Select the LRS tool from the Layer palette.
2. Point to the output layer and click the left mouse button to bring up the L/R Schedule dialog.



[✓ Learn] Recall

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Learn Count | 10000 | 30000 | 70000 | 150000 | 310000 |
| Temperature | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Learning Rate | 0.15000 | 0.07500 | 0.01875 | 0.00117 | 0.00000 |
| Momentum | 0.40000 | 0.20000 | 0.05000 | 0.00312 | 0.00001 |
| Err. Tolerance | 0.10000 | 0.10000 | 0.10000 | 0.10000 | 0.10000 |
| Weight Decay | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 5 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 6 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 7 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 8 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 9 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |

Name: Out

*The L/R Schedule dialog box.*

3. In the L/R Schedule dialog, make sure the Learn option is set. The values of the Learning Rate and momentum can be seen in 2 rows. The different columns in the schedule allow the learning parameters to change over time.
4. Click Cancel to leave the dialog box unchanged.

## Start the Training

Now that we have looked at the Learning Coefficients that will be used, we're ready to start the training.

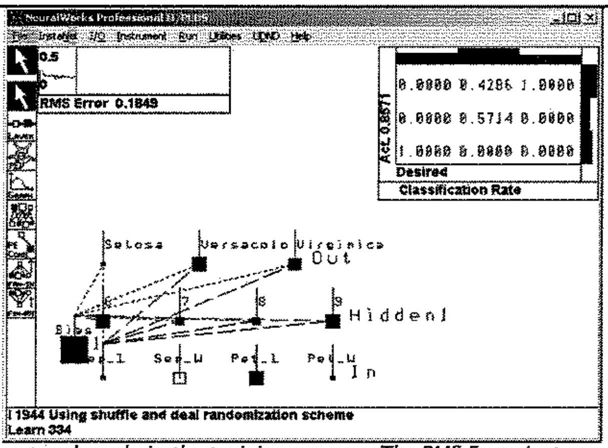
1. From the Run menu, select Learn.
2. Highlight the radio button labeled For.
3. Specify a For count of 15000 by typing this value into the text field.

   This tells NeuralWorks to present 15,000 training examples to the network in the Learn mode.
4. Click on OK to start training.

As each training example is presented to the network, the network produces an output. The difference between what the network produced and what was specified as the desired output is the network error. This error (transformed by the RMS equation), is displayed for you on the RMS error strip chart you built with the Back Propagation command. In this example the calculation is made every tenth epoch (there are 16 training cycles to the epoch).

You can interrupt the training at any time with the Escape key. Repeat the above steps to resume training.

A counter in the message box indicates the number of examples that have been presented to the network. The network diagram is not updated after each example presentation, as this would slow down the learning process considerably. (However, the network diagram can be redisplayed as often as desired with the Checkpoints facility from the Run menu.)

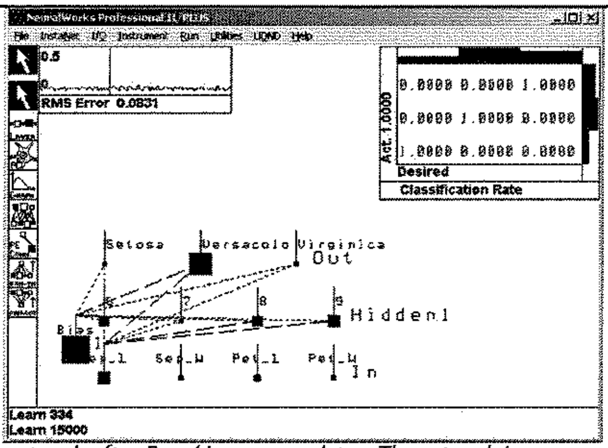


RMS Error 0.1849

```
0.0000 0.4286 1.0000
0.0000 0.5714 0.0000
1.0000 0.0000 0.0000
```

```
1944 Using shuffle and deal randomization scheme
Learn 334
```

*The network early in the training process. The RMS Error instrument shows relatively large errors.*



RMS Error 0.0831

```
0.0000 0.0000 1.0000
0.0000 1.0000 0.0000
1.0000 0.0000 0.0000
```

```
Learn 334
Learn 15000
```

*The network after Run / Learn completes. The network is now trained.*

The Utilities / Edit Counters command can be used to display a dialog box where the total number of examples that have been presented to the network are displayed as the Learn Count.

The error generated from each example is used to update the weights in the entire network. Initially, the weights start out close to zero, due to the initial randomization ranges. As the network is trained, some weights move away from their near zero starting points.

Depending on the speed of your computer, the presentation of learning examples may take several seconds.

## Testing the Network

A frequently asked question about Neural Network training is: How long should I train my network? The answer is that training time is application specific. You judge how long to train a network based on the performance you require of the network. The test phase is one way of determining how well the network has learned, and how well the network will perform.

Remember from our previous discussion of input files, that it's common practice to set aside a percentage of examples to serve as test cases. The network is trained on the training cases, and then the test cases serve as a way of measuring network performance. During this test phase, the test cases are presented to the network and the network provides results. If you know what the correct response should be, you can measure the network performance. If the test cases are representative of data the network will see in the real world, you will have a good idea of how well the network will perform its desired task.

A major difference between training and testing is that in the test case, the weights in the network are not updated. Let's go through the test phase, and see how well our back propagation network has learned to classify iris species.

To test the trained network:

1. Select the Parameters command from the I/O menu.
2. In the dialog box, make sure that the following parameters are set:

   The Recall/Test source should indicate iris_tes.nna, and the type of access for recall should be File Seq.

   The Results File should include desired output (check the box under Results File to indicate Desired O/P).
3. Click OK.
4. Select the Test command from the Run menu.

Run/Recall and Run/Test commands are similar. Both process a data set through your network without adjusting network weights. However, only the Test command reads desired outputs from your data set, and so only Test can write desired output into the results file, and update on-screen instruments which measure performance using desired outputs. The other Recall choices only read input, ignoring any desired outputs in data.

5. Click One Pass/All in the dialog box.
6. Click OK.

NeuralWorks will now read the entire test file and present all the examples to the trained network. The number of examples presented will be periodically displayed in the message box. The desired outputs, along with the actual network results, will be written to the results file.

Result files have an appended .nnr extension. As examples, input data files named data.nna, data.txt and data.mine become data_nna.nnr, data_txt.nnr and data_mine.nnr respectively.

When the network has completed the Recall, the mouse pointer will reappear, and the menu system and tool palette will again be accessible.

The Recall Count, as displayed in the Utilities/Edit Counters dialog box, is always reset before each Recall example is presented. The Recall Count only increments in network types that internally iterate during the recall phase. We will not be discussing these iterative (or as they are sometimes called, energy settling) network types in this tutorial, however, they are covered in the Neural Computing Guide.

The Recall Count differs from the Test count shown in the message box.

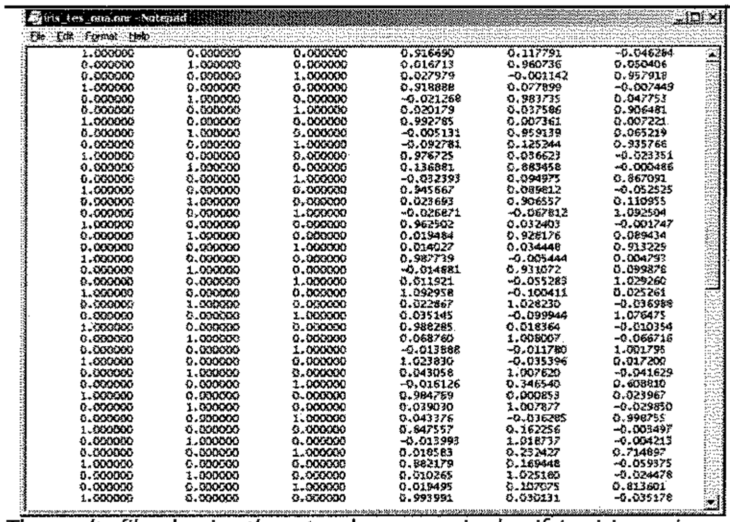
## Reading the Results

Because this is a classification problem, you can get a good assessment of the success of the network by looking at the classification rate matrix. If you want to examine the output values from the network, you will need to view the results file.

To read the results file:

1. Use your favorite text editor to view iris_tes_nna.nnr.

   The desired outputs for an example appear as the first 3 numbers in each row, and the output from the 3 PEs in the output layer appear as the last 3 numbers in each row. There is a row for each example in the test file.



```
     1.000000     0.000000     0.000000     0.916690     0.117791    -0.046284
     0.000000     1.000000     0.000000     0.016713     0.960736     0.050406
     0.000000     0.000000     1.000000     0.027979    -0.001142     0.957918
     1.000000     0.000000     0.000000     0.918888     0.077899    -0.007449
     0.000000     1.000000     0.000000    -0.021268     0.983735     0.047753
     0.000000     0.000000     1.000000     0.020179     0.037586     0.906481
     1.000000     0.000000     0.000000     0.992785     0.007361     0.007221
     0.000000     1.000000     0.000000    -0.005131     0.959139     0.065219
     0.000000     0.000000     1.000000    -0.092781     0.125244     0.935766
     1.000000     0.000000     0.000000     0.976725     0.036623    -0.023351
     0.000000     1.000000     0.000000     0.136881     0.883458    -0.000486
     0.000000     0.000000     1.000000    -0.032393     0.094975     0.867091
     1.000000     0.000000     0.000000     0.945667     0.089812    -0.052525
     0.000000     1.000000     0.000000     0.023693     0.906557     0.110955
     0.000000     0.000000     1.000000    -0.026871    -0.067812     1.092504
     1.000000     0.000000     0.000000     0.962502     0.032403    -0.001747
     0.000000     1.000000     0.000000     0.019484     0.928176     0.089434
     0.000000     0.000000     1.000000     0.014027     0.034448     0.913225
     1.000000     0.000000     0.000000     0.987739    -0.005444     0.004793
     0.000000     1.000000     0.000000    -0.014881     0.931072     0.099878
     0.000000     0.000000     1.000000     0.011921    -0.055289     1.029260
     1.000000     0.000000     0.000000     1.092958    -0.100411     0.025261
     0.000000     1.000000     0.000000     0.022867     1.028230    -0.036989
     0.000000     0.000000     1.000000     0.035145    -0.099944     1.076475
     1.000000     0.000000     0.000000     0.988285     0.018364    -0.010354
     0.000000     1.000000     0.000000     0.068760     1.008007    -0.066716
     0.000000     0.000000     1.000000    -0.013888    -0.011780     1.001795
     1.000000     0.000000     0.000000     1.023830    -0.035396     0.017200
     0.000000     1.000000     0.000000     0.043058     1.007620    -0.041629
     0.000000     0.000000     1.000000    -0.016126     0.346540     0.608810
     1.000000     0.000000     0.000000     0.984769     0.000853     0.023967
     0.000000     1.000000     0.000000     0.039030     1.007877    -0.029850
     0.000000     0.000000     1.000000     0.043376    -0.036285     0.998755
     1.000000     0.000000     0.000000     0.847557     0.162256    -0.003497
     0.000000     1.000000     0.000000    -0.013993     1.018737    -0.004213
     0.000000     0.000000     1.000000     0.018583     0.232427     0.714897
     1.000000     0.000000     0.000000     0.882179     0.169448    -0.059375
     0.000000     1.000000     0.000000     0.010265     1.025180    -0.024478
     0.000000     0.000000     1.000000     0.019495     0.107075     0.813601
     1.000000     0.000000     0.000000     0.993991     0.030131    -0.035178
```

*The results file, showing the network accuracy in classifying iris species.*

If you want to run the Test more than once, make sure that you make the proper Append and Header selections in the I/O /Parameters dialog box.

2. Give a spot check as to how often the network gave the correct response. For example, if the desired output indicated that the iris type is a Virginica (the first and second desired outputs were 0, and the third desired output was 1), then the network should have given an actual output of 0.0, 0.0, and 1.0.

   Neural Networks rarely give the exact result you desire. For example, the actual outputs for our Virginica example above are 0.02, -0.00, and 0.95. It is up to the network designer to decide if this result is accurate enough. For our example, we might judge "yes," the network is correct, since the third output (0.95) is correctly the highest.

In a later chapter, we will describe additional methods of analyzing network performance.

## Notes About Saving Networks

It's often desirable to save a network at a known point. For example, before making modifications to a model that has been serving you well, you should save it so that if the modifications don't work out, you can revert back.

To save your network, simply select the Save command from the File menu.

## Initializing a Network

Now that we have saved our network, we can go back and do some experimentation without losing our previous work. Since back propagation uses a gradient descent method of reducing error (you can only travel downhill), if there is an "uphill" section between where you are and where you want to go, you'll never get there. One method of solving this problem is to start from a different place (the other side of the hill, where by traveling downward, you can reach the global minimum). NeuralWorks gives you this capability by allowing you to initialize weights to different starting points. (You can also jog the weights at any time to get yourself over small "bumps.") To bring your network back to a new starting point, use the Run / Initialize Network command.

1. Select the Run menu, and click on Initialize Network.

This command initializes the weights in each layer to random values between the Init low and Init high values specified in each layer edit dialog box.

In addition:

❖ The input data files are rewound.

❖ The counters are reset.

❖ The instruments, if present, are initialized.

❖ Delta weights are zeroed.

❖ Disabled weights are re-enabled.

Weights can also be randomized or jogged both globally and locally between a low and high limit with the Randomize Weights and Jog Weights functions. These functions are available at the Network level using menu commands, or at the Layer and PE level using the tool palette.

To start from the exact same initial random weight set, use the I/O / Rewind Input menu. Activate (select) three settings: Close Files, Set Random Seed, and Initialize Network. When you OK this dialog, all data files are closed, the seed is reset to a known value, and the network is initialized using that known random seed. Now future networks which start out this same way can be exactly re-created.

## Adjusting Learning Coefficients

We have already taken a quick look at the learning and recall schedules. Now we'll look at these schedules more deeply. One of the milestones in the evolution of neural networks was the discovery of a method of updating weights in multi-layer networks. It has been our experience at NeuralWare that using different learning coefficients for each layer in a multi-layer network can decrease learning time. In particular, having a larger learning coefficient at the hidden layer than for the output layer allows the hidden layer to form feature detectors during the early stages of training. These feature detectors can then be combined to form more complex detectors at the output layer.

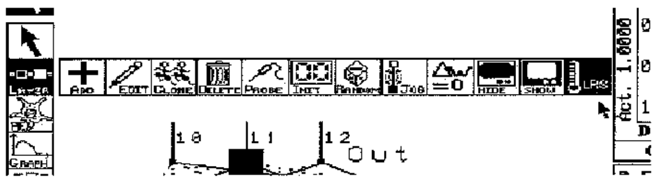
In the last example, we used the default learning coefficient and momentum term. In this next exercise we will experiment with different values. This is not meant to show that certain size coefficients are better than others, but more to show how to change these coefficients, and what effect these coefficients have on learning.

With large learning rates, a network may go through large oscillations during training. In fact, if the rates are too large, the network may never settle or converge. Smaller rates tend to be more stable. Actually, the theory of back propagation requires rates approaching zero. However, with rates that small, you could be training forever.

We'll now use the tool palette to create new schedules assigned to the output and hidden layers.

To create a schedule for the output layer:

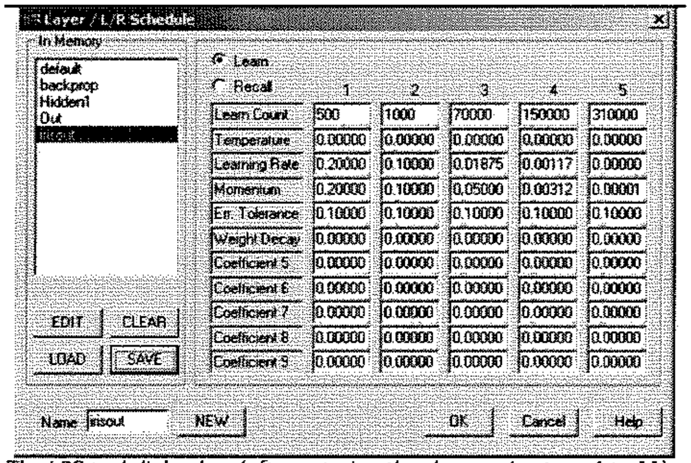
1. Click and hold on the Layer icon in the object palette.
2. Select the LRS tool in the Layer object fly-out palette.



*Selecting the Layer/LRS tool.*

The message line will prompt you to click on the layer that will be associated with the L/R Schedule.

3. Click on any PE in the output layer. The L/R Schedule dialog box appears (you have seen it once before).



Name: irisout

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Learn Count | 500 | 1000 | 70000 | 150000 | 310000 |
| Temperature | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Learning Rate | 0.20000 | 0.10000 | 0.01875 | 0.00117 | 0.00000 |
| Momentum | 0.20000 | 0.10000 | 0.05000 | 0.00312 | 0.00001 |
| Err. Tolerance | 0.10000 | 0.10000 | 0.10000 | 0.10000 | 0.10000 |
| Weight Decay | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 5 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 6 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 7 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 8 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |
| Coefficient 9 | 0.00000 | 0.00000 | 0.00000 | 0.00000 | 0.00000 |

*The LRS tool dialog box (after entering the changes in steps 4 to 11).*

4. In the name text field, type the name irisout. This name will be assigned to our schedule.
5. Change the learn count in column 1 to 500. This indicates that the Coefficients in column 1 will be valid until the Learn Count increases beyond 500.
6. Change Learning Rate in column 1 to 0.2.
7. Change Momentum in column 1 to 0.2.
8. Change the learn count in column 2 to 1000. This indicates that when the learn count is greater than the value specified in column 1, but not greater than the count in column 2, the coefficients in column 2 will be used.

   If the learn count is greater than the largest column count, the coefficients in the largest non-zero column will be used.
9. Change Learning Rate in column 2 to 0.1
10. Change Momentum in column 2 to 0.1
11. Click the NEW button at the bottom of the screen.

    Your new schedule will now appear in the In Memory window with the name irisout.
12. Save this learning and recall schedule to a file on disk by clicking the Save button.
13. Leave the dialog box by clicking the OK button.

To create a schedule for the hidden layer:

1. Click on any PE in the hidden layer (your Layer LRS tool should still be active).

The L/R Schedule dialog box appears.

2. In the name text field, type the name irishid1.
3. Change the learn count in column 1 to 500.
4. Change Learning Rate in column 1 to .8.
5. Change Momentum in column 1 to .2.
6. Change the learn count in column 2 to 1000.
7. Change Learning Rate in column 2 to 0.3.
8. Change Momentum in column 2 to 0.1.
9. Click the NEW button at the bottom of the screen.
10. Save this learning and recall schedule to a file on disk by clicking the Save button.
11. Leave the dialog box by clicking the OK button.
12. Drop the LRS tool by clicking the Pointer tool.

## Setting a New Global Learning and Recall Schedule

Since both the output and the hidden layer now have an L/R schedule (LRS) associated with them, it was not necessary to assign a global schedule. The global LRS is used only by layers which do not have a local LRS assigned to them. For example, Back Propagation Input and Bias layers typically use the global LRS. Since Input and Bias layers have no incoming weights, and hence they do not train any weights, LRS parameters are not important). Note that if you were to manually add a new layer to the network, it would use the global LRS unless you also assign a LRS to that new layer.

1. Select the Global Network Parameters command in the Utilities menu.
2. Set the global learning and recall schedule to irisout. This schedule will serve as the default schedule for all layers that do not explicitly name an L/R schedule.
3. Click OK.

When creating networks using the InstaNet menu, some learning rules (Delta-Rule and Norm-Cum-Delta) create a unique LRS for hidden and output layers. Others (Ext DBD, QuickProp, LVQ ) allow the various layers to simply use the network's global LRS. If in doubt about which schedule is in use by which layer, use the Layer / Edit and Layer / LRS icons to display the names of the layers, and the LRS contents, for each layer of a given network. In some networks you'll see several layers using the same LRS.

## A Bit More About Learning and Recall Schedules

Schedules are saved on disk as stand-alone entities when you click the SAVE button in the L/R dialog box. To use an L/R Schedule that has been previously saved on disk, bring it into memory with the LOAD button in that dialog box.

Caution: Schedules saved on disk are not related schedules that are in memory, which get saved within a network when a network is saved. Loading a network causes schedules saved within that network to be loaded into memory.

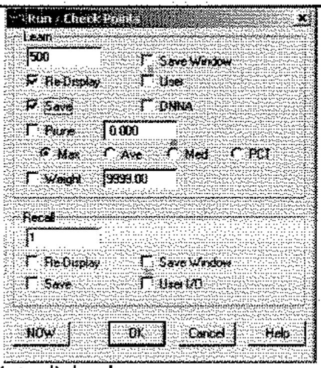
Schedules in memory can be removed from memory using the CLEAR button. Clearing a schedule does not delete it from disk. You can't clear a schedule that is being edited or is being used by the currently loaded network.

In summary, until you are quite familiar with the aspects of creating, loading, saving and editing schedules, always check that you have the schedules you want in place in each layer before starting the learning phase.

## Using Checkpoints to Automatically Save Your Network

You are familiar with how to adjust the learning parameters over time, based on the learn count. You have also learned how to save a network manually. Wouldn't it be great if you could automatically save your network at those times that learning coefficients are being changed, so that if the new co-efficients don't work out, you don't have to start training over again from the beginning?

NeuralWorks gives you a simple way of saving a network automatically. Called checkpoints, the method is based on the learn count. For example, during training, you may want to save the network every 500 example presentations.



Run / Check Points

Learn
500
☐ Save Window
☑ Re-Display ☐ User
☑ Save ☐ DNNA
☐ Prune 0.000
● Max ○ Ave ○ Med ○ PCT
☐ Weight 9999.00

Recall
1
☐ Re-Display ☐ Save Window
☐ Save ☐ User I/O

*The Run/Check Points dialog box.*

To save a network automatically:

1. Select the Check Points command from the Run menu.
2. Type 500 into the Learn text field. This is the interval, in training cycles, for saving the network.
3. Select the check box labeled Re-Display.
4. Select the check box labeled Save. The network display will be updated on the screen at the same time the network is being saved
5. Click OK to leave the dialog box.
6. Select the Save As command from the File menu.
7. Type the name baby00 (two zeros) into the text field.
8. Click on the Save button.

The digits at the end of the name you entered in the Save As dialog indicates to NeuralWorks that each time the network is saved with the checkpoints facility, the name used will be unique. During each consecutive save, the number at the end of the name will be incremented by 1. In this example, after 500 examples have been presented, the network is saved with the name baby01. When the learn count reaches 1000, the network is saved with the name baby02, and so on.

The number of digits at the end of baby00 is significant. If none, baby.nnd is overwritten at each save. For baby0, up to 10 networks (0 through 9) are saved before overwriting baby0.nnd upon the 11th save. Baby00 saves the last 100 networks!

The automatic save facility can quickly consume large quantities of disk space; use it with care. The way to delete a network from disk is from the operating system.

A very important and related task to automatically saving networks is the NeuralWorks Run / Save Best command. Save Best is a very useful way to keep networks from 'overtraining'... See the Reference Guide for details.

## Retraining your Network

Now that we've reset the learning coefficients and set up the automatic save function, go ahead and retrain your initialized network with the new learning coefficients.

The questions you'll want to answer are:

- Do these new coefficients cause the RMS error to drop faster?
- What about stability?

- ❖ Do you think 15,000 iterations was enough with these new coefficients?

After training, access Run / Check Points and deselect the Re-Display and Save buttons.

## Checking for PE Saturation

When a value coming into a PE is beyond the PE's transfer function range, that PE is said to be "saturated." The sigmoid function, which we've discussed in the previous example, will accept values only between -6 through +6. The TanH function, which you'll use in the next exercise, accepts value between -3 and +3. Saturation occurs when a PE's summation value (the sum of the inputs times the weights) exceeds this range.

During saturation, the sigmoid transfer function produces 0 or 1 and the TanH produces either -1 or 1 for any out of range value, no matter how great the magnitude. For example, a PE will produce a value of 1 for either 7 or 70,000. Also, a saturated PE will produce an error of 0 during error back propagation, which means no learning can occur.

Saturation can occur immediately if the initial weights are too large. When NeuralWorks creates a back propagation network, it sets reasonable ranges for the initial weights depending on the number of input nodes and assuming the input values lie between -1.0 and 1.0. Saturation can also occur early in the learning if learning rates are too high.

A saturated PE late in the learning is valid and corresponds to a learned feature in the data.

You can use EasyProbe to create a simple instrument to monitor saturation. EasyProbe is one of the many instrument making facilities within NeuralWorks. You've already used the instrument dialog box that follows the back propagation command to create the RMS error graph. EasyProbe is almost as simple to use.

In the following exercise, you'll:

- ❖ Initialize the network.
- ❖ Use EasyProbe to create an instrument to monitor saturation.
- ❖ Check if saturation is occurring.
- ❖ Adjust the Vmin and Vmax settings for the Min/Max Table and see what effect these have on saturation.

1. Select the Initialize Network command in the Run menu. This resets the network to its "untrained" state.
2. Select the EasyProbe icon in the Graph Palette.

3. In the dialog box that appears, select the following:

   In the Variable Graphed column, select Summation Value.

   In the Probe Type column, select Entire Network.

   In the Graph Type list, select Bar.
4. Deselect the Transform across Epoch check box..
5. Click OK.
6. The mouse cursor will change into an "X," prompting you to find a position for the graph. Click over the area where you want the graph.

You've now created an instrument using Easy Probe. You may want to click and hold inside one of the instrument corners and stretch the instrument a bit wider. Since we're dealing with a sigmoid transfer function, you'll need to reset the vmin and vmax values to -6 and +6 respectively. To do this:

1. Double-click the Summation Value instrument. This will produce the Graph Edit dialog box.
2. Reset the vmin value to -6.
3. Reset the vmax value to 6.
4. Click OK.

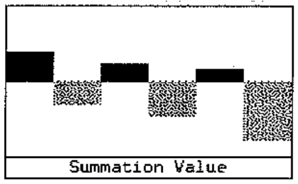
OK, now you're ready to retrain the network and watch what happens.

1. Select the Learn command under the Run menu. The learn value should still be set at 15,000.
2. Click OK.

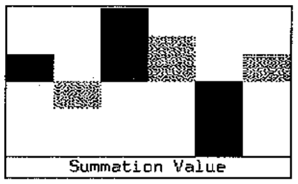
Watch the summation value graph. If everything is going well , the bars shouldn't collide with the top or the bottom of the graph during the first few hundred learn iterations. On the other hand, there should be action in the bar movements.

Your network should show nearly ideal activity in the Summation chart. However, if there were too much or too little activity, you can correct this using the Parameters command in the IO menu. To increase the level of activity, expand the Network Ranges for the Input. For example, instead of 0 and 1 you might try -.2 and 1.2. If saturation is occurring, you can reduce activity by decreasing the range for the Input (to .2 to .8 for example). You can experiment with resetting the Network Ranges for the Input and re-running the network. Remember to initialize the network before each learning run.

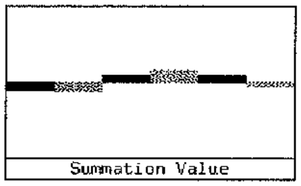
*The Summation Graph to the right shows "good behavior," with the bars moving but without reaching the top or bottom of the graph.*



*This Summation Graph shows saturation. Note how the bars are hitting the top and bottom of the scale.*



*This Summation Graph shows almost no movement, and therefore very slow learning.*



*Possible variations in the Summation Graph, and their meanings.*

## Some Additional Things to Try

- Try different learning coefficient values.
- Try jogging the weights a bit if the network appears to be stuck in a local minimum (RMS error no longer dropping). You can jog or initialize the weights for individual PEs or connections using the tool palette.
- The F′ Offset (in the Back Propagation dialog) can be used to control saturation. A value of 0.1 is recommended for networks with a sigmoid transfer function. A value of up to 0.3 can be used if a TanH transfer function is used.

## Optimizing the Iris Network

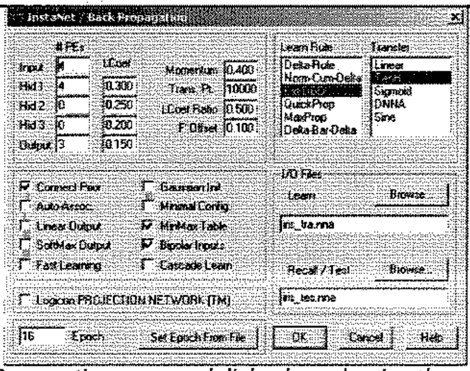
In the previous exercises, we built and experimented with a back propagation network that classifies irises by species. Now we'll build a more sophisticated back propagation network for the same application. We'll also introduce these NeuralWorks features:

| | |
|---|---|
| The Extended Delta-Bar-Delta Learning Rule | A variation from standard back propagation that automatically adjusts learning coefficients. |
| Hyperbolic Tangent (TanH) transfer function | An alternative to the sigmoid transfer function. |
| Connect Prior | Makes connections from each layer to ALL prior layers. For example, this additionally adds connections between an input and output layer, 'jumping' across any hidden layers. |
| MinMax Tables and Network Ranges | NeuralWorks built-in pre-processing capability. |
| The Confusion Matrix | A powerful instrument that allows you to see how close your outputs match network predicted outputs. |

To build the optimized network:

1. Select the InstaNet / Back Propagation menu.
2. Enter the following number of PEs:

| | |
|---|---|
| Input | 4 |
| Hid 1 | 4 |
| Output | 3 |


InstaNet / Back Propagation

| # PEs | | LCoef |
|---|---|---|
| Input | 4 | |
| Hid 1 | 4 | 0.300 |
| Hid 2 | 0 | 0.250 |
| Hid 3 | 0 | 0.200 |
| Output | 3 | 0.150 |

Momentum 0.400
Trans. Pt. 10000
LCoef Ratio 0.500
F' Offset 0.100

Learn Rule: Delta-Rule, Norm-Cum-Delta, [illegible], QuickProp, MaxProp, Delta-Bar-Delta
Transfer: Linear, [illegible], Sigmoid, DNNA, Sine

[✓] Connect Prior
[ ] Auto-Assoc.
[ ] Linear Output
[ ] SoftMax Output
[ ] Fast Learning
[ ] Gaussian Init
[ ] Minimal Config
[✓] MinMax Table
[✓] Bipolar Inputs
[ ] Cascade Learn
[ ] Logicon PROJECTION NETWORK (TM)

I/O Files
Learn: iris_tra.nna
Recall / Test: iris_tes.nna

16 Epoch

*The Back Propagation command dialog box, showing the settings for the enhanced iris network.*

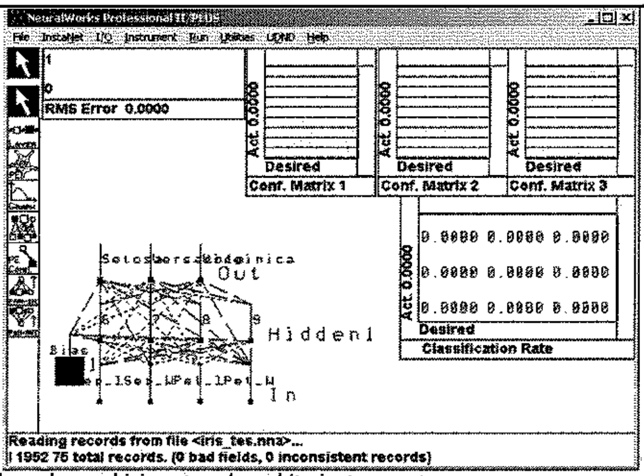
3. In the upper Learn Rule scroll window, select the learning rule named 'Ext DBD'. This is our default learn rule.
4. In the Transfer scroll window, select the TanH transfer function. TanH is our default transfer function.
5. Type, or browse and select the training file iris_tra.nna.
6. For Recall/Test, select the testing file iris_tes.nna.
7. Leave the following check boxes selected:

   Connect Prior
   MinMax Table
   Bipolar Inputs
8. Click OK.

   If you have a network already on the screen, you'll be asked to verify if it's OK to delete that network. Click the Yes button to remove the current un-saved network from memory.
9. Next, you'll see the instruments dialog box of suitable instruments for Extended Delta-Bar-Delta (EDBD) networks. In this box, select the following:

   RMS Error
   Classification Rate
   Confusion Matrices
10. Click OK

Arrange the instruments and network on the screen so that they look similar to the following figure.



```
0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
0.0000 0.0000 0.0000
```

```
Reading records from file <iris_tes.nna>...
| 1952 75 total records. (0 bad fields, 0 inconsistent records)
```

*The enhanced iris network and its instruments.*

## A Bit of Explanation

The following sections explain some of the options you selected.

### Extended Delta-Bar-Delta Learning Rule

The 'EDBD' learning rule overrides learning coefficients specified in the InstaNet Back Propagation dialog, and instead load it's starting coefficients from a disk file (edbd.nnt). EDBD then trains by self-adjusting coefficients on each and every weight in the network (each weight has it's own alpha learning rate and momentum terms).

### Hyperbolic Tangent (TanH) Transfer Function

The hyperbolic transfer function is quite similar to the Sigmoid transfer function in shape (s-shaped). However, its output range is -1 to +1, as opposed to the sigmoid range of 0 to 1. Because the output of the transfer function is used as a multiplier in the weight update equation, a range of 0 to 1 means a smaller multiplier when the summation is a low value, and a higher multiplier for higher summations. This could lead to a bias to learning higher desired outputs (approaching 1). The hyperbolic tangent gives equal weight to low and high end values.

### MinMax Tables and Network Ranges

A common cause of problems stems from presenting data to a back propagation network as raw values, rather than in values that have been suitably scaled to the neuro-dynamic functions being used. For example, a back propagation network often uses sigmoid or hyperbolic transfer functions, which respond in a nearly linear fashion to summations between about -2 to +2. If a user presents a back propagation network with input values such as 10,000, then even with small weights in the network, the summations will be huge and the sigmoids will become saturated. When saturated, the derivative of the sigmoid (or hyperbolic tangent) is close to zero at large (either positive or negative) summation values. Since the derivative is a multiplier in the weight update equation, learning stops for PEs with such large summation values.

PEs with saturated sigmoids have summations that are either always too low or too high, and the sigmoids output either a 0 or 1.

NeuralWorks addresses this problem with an easy to use pre-processing facility. The pre-processing facility computes the lows and highs of each data field for all the input data files to be used with a given network. These lows and highs are stored in a table called a " MinMax Table". The user then decides what ranges the input and output should be scaled to for presentation to the network. NeuralWorks then computes the proper scale and offset for each data field. Real world values are then scaled to network ranges for presentation to the network. After the network has produced a network scaled result, the result is de-scaled to real world units.

As we discussed in the last section, saturation can still occur, even when using a MinMax Table. Common causes of saturation include learning rates that are too high, initial random values that are too large or relative learning rates for different layers that are inappropriate. Layers with many incoming connections require very small random initial values.

When you clicked OK in the InstaNet / Back Propagation dialog, and because the MinMax Table check box was selected, NeuralWorks automatically created a MinMax Table by reading through both of your training and testing data files. In this exercise the Network Ranges were set to scale data into an input range between -1 and 1, and desired outputs scale into a range of -0.8 and 0.8. The input range is -1 to 1 because the Bipolar Inputs check box was selected (otherwise data would scale into a less efficient 0 to 1 range). The output range is -0.8 to 0.8 because of the use of TanH; if you chose a Sigmoid function, the desired output range would be set to 0.2 to 0.8.

Aside from the InstaNet menus, this data scaling facility is accessed from two other dialog boxes: the I/O / Parameters dialog box to set network ranges and the I/O / MinMax Table dialog box to create and edit MinMax Tables. As with all schedules and tables, MinMax Tables are saved internally when the network is saved, as part of the *.nnd file. They can also be saved independently, and have a .nnm extension.

See the *Reference Guide* for more information about MinMax Tables.

## Bad Fields and Inconsistent Records

The MinMax Table facility will notify you of any discrepancies it finds in the data files. For example, if a value in one of the data files is not a number (called a NAN), you are notified that a bad field exists. Similarly, if a data record contains more or less fields than existed in the very first data record, you are notified that an inconsistent record exists.

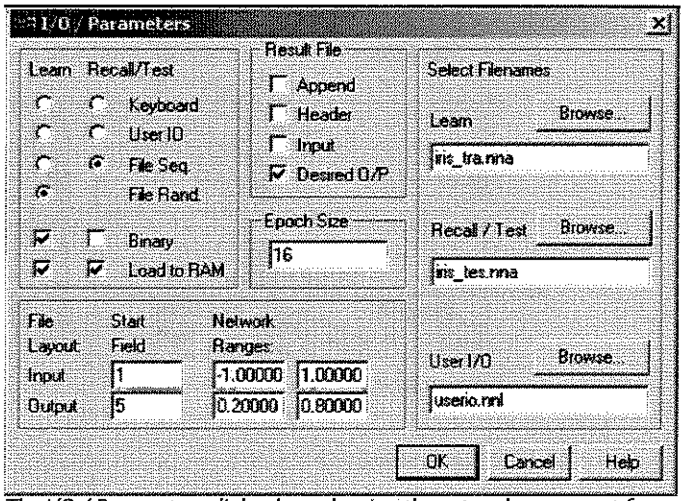
Bad and missing fields (non-numeric values) are properly handled by replacing them with the mid-point of each field's data range - the midpoint between the MinMax Tables' high and low values, for that particular data field.

## Network Ranges

Clicking OK in the InstaNet / Back Propagation dialog also set network ranges appropriately. You can see these through the I/O / Parameters dialog box. In this case, the settings are:

| | |
|---|---|
| Input Network Ranges | -1.0 to +1.0 |
| Output Network Ranges | -0.8 to +0.8 |

Remember that -0.8 to +0.8 is for a hyperbolic tangent—the range for a sigmoidal function would be 0.2 and 0.8. The choice of Network Ranges is closely linked to the number of inputs, the type of transfer function in use, and the initial weight values in the network. The basic idea is to choose a range that produces summations which will not initially saturate the transfer function.



| | Learn | Recall/Test |
|---|---|---|
| Keyboard | ○ | ○ |
| User IO | ○ | ○ |
| File Seq | ○ | ● |
| File Rand | ● | |
| Binary | ✓ | ☐ |
| Load to RAM | ✓ | ✓ |

Result File: ☐ Append ☐ Header ☐ Input ✓ Desired O/P

Epoch Size: 16

Select Filenames
Learn: iris_tra.nna
Recall / Test: iris_tes.nna
User I/O: userio.nnl

| File Layout | Start Field | Network Ranges | |
|---|---|---|---|
| Input | 1 | -1.00000 | 1.00000 |
| Output | 5 | 0.20000 | 0.80000 |

*The I/O / Parameters dialog box, showing the network ranges set for this network configuration by InstaNet / Back Propagation.*

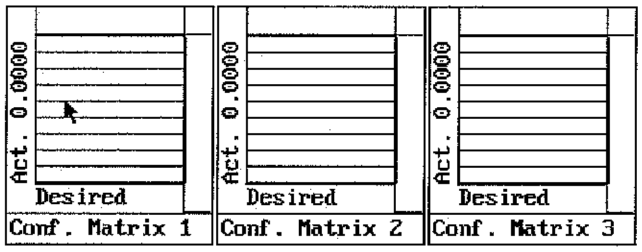
## Confusion Matrix

Up to this point, we've measured how well the network was doing during the Learn phase by observing the RMS error, and for the Recall phase by looking at the results file.

The Confusion Matrix provides an advanced way of measuring network performance during the Learn and Recall phase. It allows you to correlate the actual results of the network to the desired results in a visual display.

The confusion matrix gives you a visual indication of how well your network is doing. What you would like to have are the bins (the cells in each matrix) on the diagonal from the lower left to the upper right showing the highest counts. This indicates that the network is giving you an output that matches the desired output. For example, if the desired output for an example was .8, and the network produced a result of .8, the bin that is the intersection of .8 on the x axis and .8 on the y axis on the instrument would have its count updated. This bin appears on the diagonal. Now, let's say the desired output is .8 and the network produces a .2. The bin that is the intersection of .8 on the x axis and .2 on the y axis is off the diagonal, in the lower right of the instrument. Counts in this off-diagonal bin will visually indicate that the network is predicting low when it should be predicting high.

The bins in the confusion matrix represent values accumulated over the last Epoch. If you want the confusion matrix to show the performance of your entire training set, perform a Run / Test command and select Training File Pass.



*The confusion matrices.*

## A Useful Analogy

A useful analogy for interpreting the confusion matrix display is to think of it as a scatter diagram. For every output the network produces, a point is created on the scatter diagram that shows how the actual output matches against the desired output. However, instead of plotting every point, the scatter diagram is divided into areas, or bins. The number of points that end up in each area or bin are added up. The height of the bar in the bin is a measure of how many points the bin contains. Remember that the bin with the most points will be shown full height, and the other bins will be scaled accordingly.

### Added Features

The histogram that runs across the top of the instrument shows you the distribution of your desired outputs. The histogram along the right side of the instrument shows the distribution of the actual output (predictions). If any outliers exist in your actual output, they will be shown in the uppermost and lowermost bin on the right side of the instrument.

The value on the vertical axis is the Common Mean Correlation coefficient of the desired (d) and actual (predicted) output (y) across the epoch. This is calculated as

$$\frac{\sum (d_i - \bar{d})\ (y_i - \bar{y})}{\sqrt{\sum (d_i - \bar{d})^2 \sum (y_i - \bar{d})^2}}$$

$$\bar{d} = \frac{1}{E} \sum d_i$$

$$\bar{y} = \frac{1}{E} \sum y_i$$

Where E = epoch size, and summations are over the epoch. This is a simple, normalized overall measure of how well desired and actual output correlate:

| | |
|---|---|
| 1 | = perfectly correlated |
| 0 | = completely uncorrelated |
| -1 | = perfectly anti-correlated |

The Common Mean Correlation coefficient is very similar to the Pearson Linear Correlation (R), except that data series with different means are further penalized (R is reduced based on the difference in means).

Remember:

❖ A confusion matrix can only display for 1 output PE at a time.

❖ The confusion matrix always plots what's in the PE error field against what is chosen in the instrument's Variable field.

## Running The Enhanced Iris Network

Now that you have some background in what we selected for the enhanced network, let's run it and see what it does.

To train the network:

1. Select Learn from the Run menu.
2. Click on the For radio button.
3. Specify a value of 15000 in the For text field.

4. Click OK.

Note the confusion matrices as training progresses. In our network, each desired output is a -0.8 or a +0.8. Because the histogram at the top of the instrument ranges from -1 to +1, and it is sectioned into 9 bins, values -0.8 and +0.8 fall into the left-most and right-most bins. We expect to see the histogram at the top of each confusion matrix filled in only the two extreme bins. If the network is doing a good job for us, we should see the bins in the lower left and upper right in the main body of the matrix show the highest counts.

Also note that the RMS Error instrument has a lower error than our previous attempts.

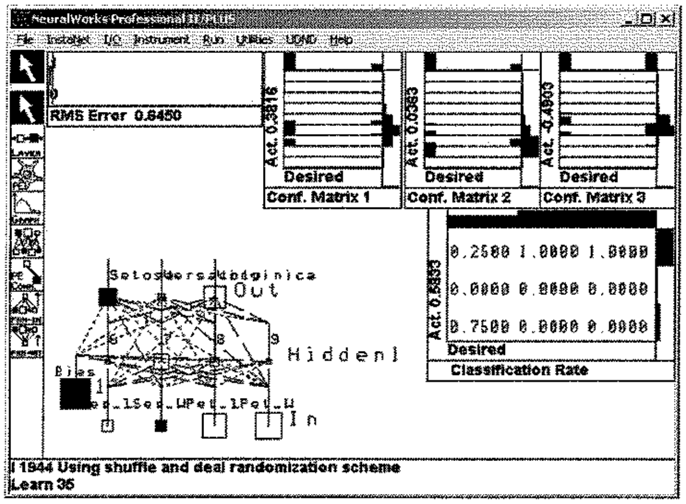
Remember that the instruments show only the last Epoch number of records, which is a subset of your training file. You can record the R Correlation, Average Class Rate and RMS Error metrics at this point by running Test commands.

Test the network's performance on the training data set:

1. Select the Run / Test menu.
2. Click on the Training File Pass radio button.
3. Click OK.
4. Record the numbers shown on each instrument. These are measurements achieved across your entire training file.

Test the performance on the independent test data set:

1. Again select the Run / Test menu.
2. Click on the One/Pass All radio button.
3. Click OK.
4. Record each instrument's results. These are measurements across your entire test file.

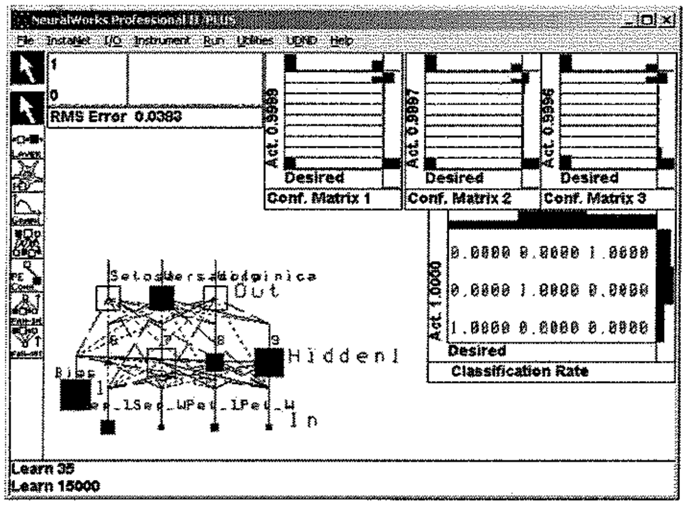


RMS Error 0.6450

```
0.2500 1.0000 1.0000
0.0000 0.0000 0.0000
0.7500 0.0000 0.0000
```

```
I 1944 Using shuffle and deal randomization scheme
Learn 35
```

*The enhanced network at the beginning of training. Note the bins in the confusion matrices.*



RMS Error 0.0383

```
0.0000 0.0000 1.0000
0.0000 1.0000 0.0000
1.0000 0.0000 0.0000
```

```
Learn 35
Learn 15000
```

*The enhanced network after training. Now note how the confusion matrix bins fall in diagonal corners.*