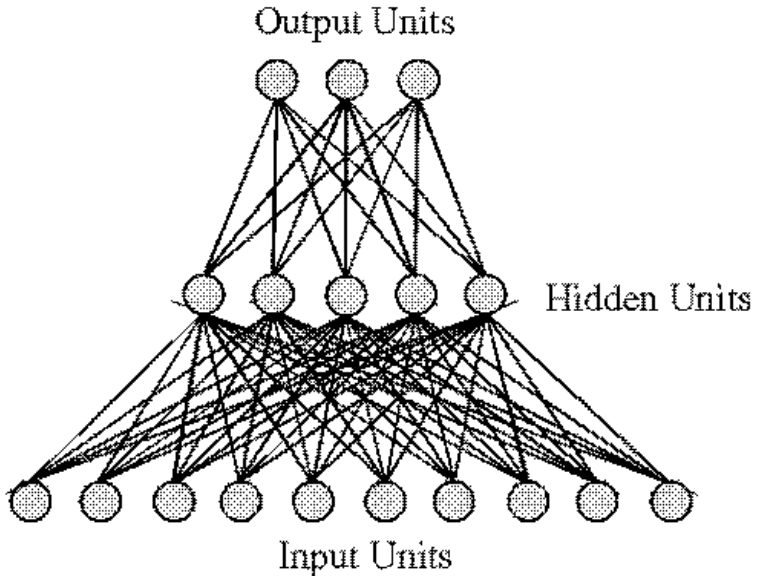
User Manual For The Rumelhart and RumelhartLite Multilayer Perceptron Programs

Michael R.W. Dawson and Vanessa Yaremchuk
February 26, 2003
Biological Computation Project
University of Alberta
Edmonton, Alberta, Canada
http://www.bcp.psych.ualberta.ca



INTRODUCTION

Rumelhart is a program written in Visual Basic 6.0 for the demonstration and exploration of multilayer perceptrons. It is designed for use on a computer based upon a Microsoft Windows operating system. The program is part of a multimedia support package for a book by Michael R.W. Dawson, *Minds and Machines: Connectionism and Psychological Modeling,* which is currently in production by Blackwell Publishing. Michael Dawson and Vanessa Yaremchuk programmed the current version of Rumelhart. A second program, RumelhartLite, is identical to Rumelhart with the exception that it does not include the capability to save network results in Microsoft Excel workbooks. In this document, Rumelhart will be the only program referred to, as the user interface for it is identical to the interface for RumelhartLite. Both programs are distributed as freeware from the following website:

http://www.bcp.psych.ualberta.ca/~mike/Book2/

The purpose of the multilayer perceptron program is to learn a set of stimulus/response associations, which are usually interpreted in the context of pattern classification. This means that network responses are usually interpreted as representing names or categories that are applied to stimuli. The current program explores pattern classification with two main types of processing units: integration devices, which use the typical logistic activation function, and value units, which use a Gaussian activation function. These two processing units can be combined to create four different network types. The first is all integration devices. The second is all value units. The third uses value units as outputs, and integration devices as hidden units. The last uses integration devices as outputs, and value units as hidden units. Furthermore, the user always has the option of including direct connections from the input units to the output units. These variations of the multilayer perceptron are described in more detail in Chapter 11 of the book for which this multimedia site has been constructed.

INSTALLING THE PROGRAM

Rumelhart is distributed from the above website as a .zip file. The following steps will result in the program being installed on your computer:

1. Download the file Rumelhart.zip to your computer by going to the website, click on the program icon, and save the file in any desired location on your computer.
2. Go to the saved Rumelhart.zip file on your computer, and unzip it with a program like WinZip. The result will be three different objects: setup.exe, setup.lst and Rumelhart.cab.
3. Run the setup.exe program. This will call an Install program that will complete the installation of the program on your computer, which will include the installation of an Examples folder with a few sample training files.

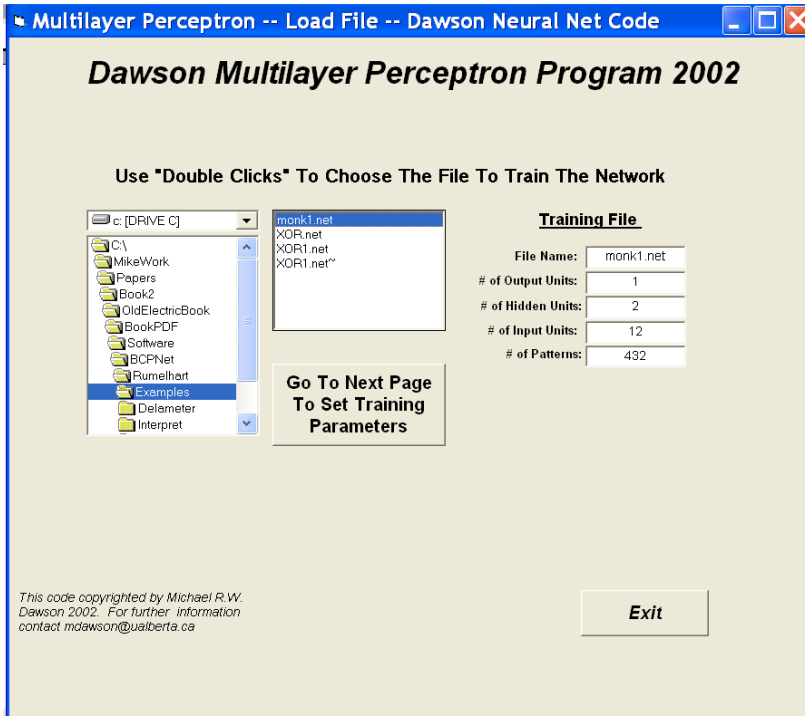
TRAINING A MULTILAYER PERCEPTRON

Starting The Program

The program can be started in two different ways. First, one can go into the directory in which the program was installed and double-click on the file “Rumelhart.exe”. Second, one can go to the start button on the computer, choose programs, scroll to the program group BCPNet, and select the program Rumelhart.exe.

Loading A File To Train A Network

After the program is started, the first form that appears is used to select a file for training the distributed memory. This form is illustrated on the right. By using the left mouse button and the drive selection tool located in the upper left of the form, one can choose a computer drive on which directories and files are located. The available directories on the selected drive are listed in the directory selection tool that is immediately below the drive selection tool. One opens a directory by double-clicking it with the left mouse button. If the directory contains any files that end with the extension .net, then these files will be displayed in the file selection box located in the upper middle of the form. The properties of .net files are described later in this manual. These files have a particular format that the Rumelhart program is designed to read, and only files that end in this extension can be used to train the network.



One chooses a .net file by double-clicking one of the file names that is displayed in the file selection box. When this is done, the program reads the desired file, some of the file's properties are displayed, and another button appears on the form. In the figure on the right, the file "monk1.net" has been selected (and read). On the right of the form its general properties are displayed, and the button permitting the user to proceed to the next part of the program is displayed under the file selection box.

In this example, if "monk1.net" has been selected, but is not really the file that is desired, one can simply go back to the file selection tools and choose another file. When its file name is double-clicked, the new file will be read in, and will replace the properties of the previous (undesired) file.

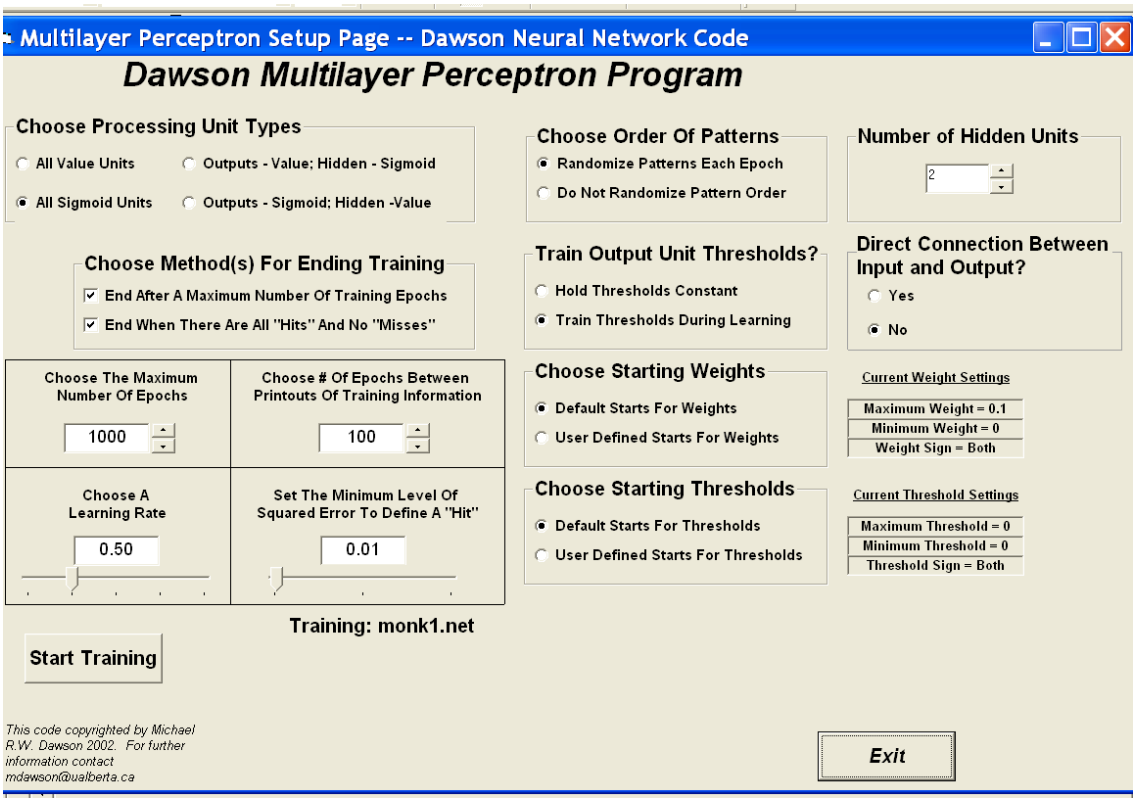
Once the desired file has been selected, all that is required is to press the "Go To Next Page To Set Training Parameters" button with a left-click of the mouse. If instead one desires to close the program, then one can left-click the "Exit" button displayed on the bottom right of the form.

Setting The Training Parameters And Training The Network

When the program reads in the .net file, this only determines how many processing units are connected in the network, and defines the input and desired output patterns that are used in training. It is up to the user to define what learning rule to use, and to specify the value of the parameters to control (and stop) learning. The second form displayed by the program allows the user to choose these parameters. The paragraphs below describe how this is done. If the reader wishes to learn more about what exactly is accomplished by setting these values on this form, then he or she should look through Chapter 11 of *Minds And Machines: Connectionism And Psychological Modeling*.

The second form consists of a number of different tools that can be used to quickly control the kind of learning that will be carried out by the multilayer perceptron. The first tool is used to choose which of four general architectures are going to be used to construct the multilayer perceptron. In essence, this tool determines the type of processor that will be used in the output units (integration device or value unit) as well as the type of processor that will be used in the hidden units (integration device or value unit).

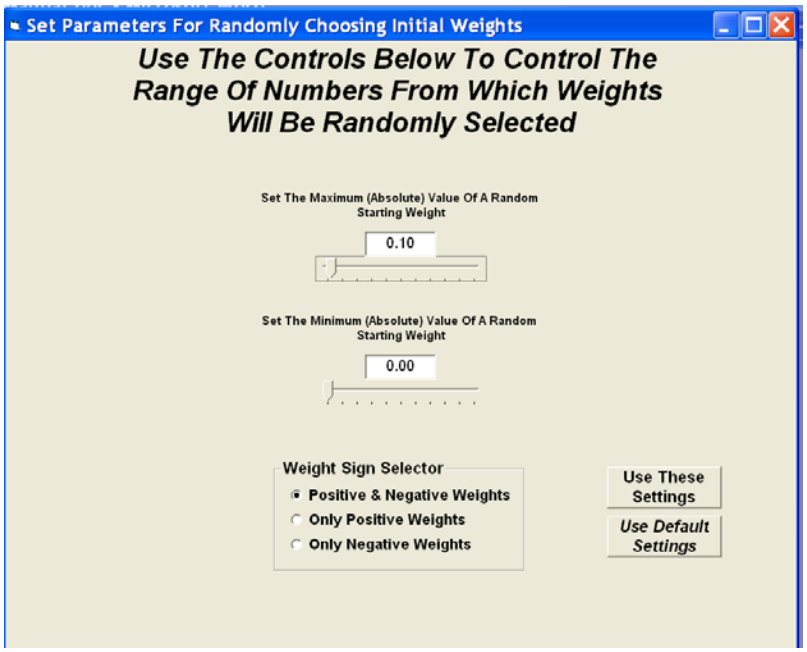
When a particular architecture is selected, default values for learning rates will also be set. The user can change these later by the user if desired. If all integration devices are used, then the learning rule that will be adopted is the gradient descent rule proposed by Rumelhart, Hinton, and Williams (1986). If all value units are used, then the learning rule will be the modification of the gradient descent rule proposed by Dawson and Schopflocher (1992). For the other two architectures, the learning rule that is applied will be defined by the choice of output unit.



A second tool is used to choose a method for stopping training. In the first method, training stops after a maximum number of epochs (this value is set by the user). In the second method, training stops when there is a "hit" for every pattern and every output unit. This means that when each output is generating an acceptably accurate response for each pattern, training will stop. A left-click of the mouse is used to select either of these methods; when a method has been selected, a check mark appears in the tool. Importantly, the user can select both methods to be used in the same simulation. When this is done, then the simulation will stop as soon as one of the two conditions is met. This is the default situation, and it is recommended

A third tool determines the order in which patterns will be trained. The program is epoch-based, which means that each epoch or "sweep" of training involves presenting every pattern once to the perceptron. When a pattern is presented, output unit error is used to modify the weight values. One can have the program present patterns in a random order each epoch, which is the recommended practice. However, if pattern order is being manipulated, you can turn this option off with a left-click of the mouse. When this is done, the patterns will always be presented in the order in which they are listed in the .net file that has been input.

A fourth tool determines whether unit thresholds (i.e., the logistic function's bias, or the value unit's mu) is to be trained. The default is to train this value, because this permits the output unit to "translate" its "cut" through pattern space. However, in some situations it may be required to hold this value constant, which can be done with a left-click of the mouse button.



A fifth tool is used to determine the starting values of the connection weights, which are randomly selected from a distribution. In the default situation, the maximum value of a weight is 0.1, the minimum value is 0, and the sign option is "both", which means that negative and positive weights are possible. These defaults are displayed to the right of the weight-start tool. With these default values, weights will be randomly selected from a rectangular distribution that ranges from –0.1 to +0.1. However, in some cases it may be desirable to explore different starting states. This can be accomplished by left-clicking the "User defined starts for weights" option. When this option is selected, a new form appears, as is shown on the right. This form is used to set the minimum (absolute) value for a weight, the maximum (absolute) value for a weight, and the desired sign for the weight (positive, negative, or either). When the desired settings have been selected, the "Use These Settings" button will select them, and close the form. If it is decided that the default settings are desired, then this can be accomplished by using the "Use Default Settings" button. Whatever settings have been selected will be updated on the right of the settings form.

A sixth tool is used to determine the starting values of the randomly selected thresholds for the output units. The default is to assign every output unit a threshold of 0, regardless of which activation function has been selected. If different randomly selected starts are desired, then a left-click of the "User defined starts for thresholds" option will reveal a form similar to the form described above for manipulating the starting parameters for the weights.

A seventh tool allows the user to change the number of hidden units in the network, overriding the number of hidden units prescribed by the .net file that was read in. If the user wishes to alter the number of hidden units, then he or she can type in a new value in the text box, or manipulate the arrow tools with the mouse to increase or decrease the value. This manipulation will not change the .net file that was input. We use this tool to try and find the minimum number of hidden units required by a multilayer perceptron to solve a problem of interest.

An eighth tool permits the user to include direct connections between input and output units. The default situation does not include such connections, but they can be included by selecting the "Yes" value on this tool. This will increase the power of the network, and when selected, you might also consider reducing the number of hidden units used by the network.

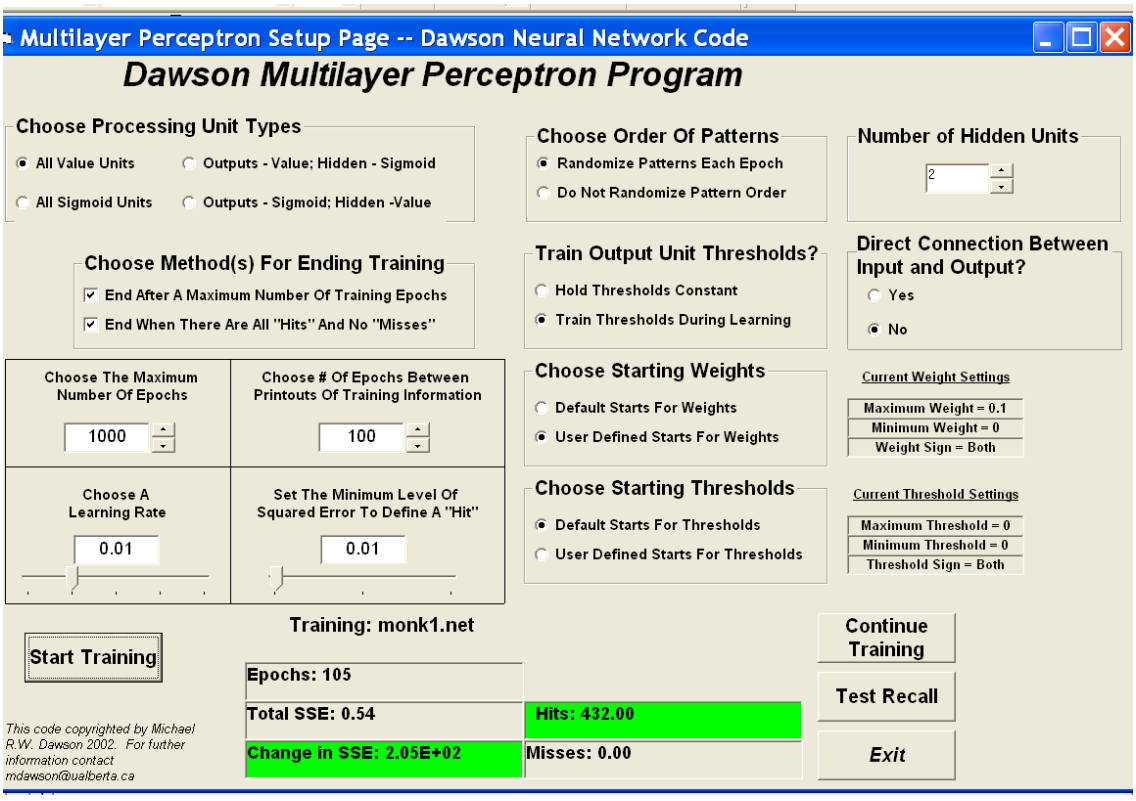
The four remaining tools on the form are used to set numerical values that control training.

The first is a tool for specifying the maximum number of training epochs by left-clicking either arrow beside the value's box. This will either increase or decrease the value of this parameter, depending upon which arrow is selected. The maximum number of training epochs can also be set directly by left-clicking the value's box with the mouse, and typing in the desired value. Note that it if the user chooses a value for this variable, then the "End After A Maximum Number Of Training Epochs" selection should also be selected. If this latter option does not have a check mark beside it, then the program will ignore this number when it is run! The default value (shown above) is 1000.

The second is a tool for specifying the number of training epochs between printouts of training information. During training, the program will periodically print out information to tell the user how things are progressing. This includes information about what epoch has been reached, what the network SSE is, and the degree to which network SSE has changed since the last printout. The frequency of these printouts is controlled by the number displayed in this tool, which can be set in a fashion similar to that described for the previous tool. The default value (displayed in the figure) is 100. If this value is selected, then every 100 epochs the user will receive updates about network learning. The value selected for this parameter also defines the spacing of the x-axis of the "SSE by Epochs" plot that can be created from a form described later in this document.

The third is a tool for specifying the learning rate used in the learning rule. More details on the role of learning rate in the equations can be found in Chapter 10 and Chapter 11 of *Minds And Machines: Connectionism And Psychological Modeling*. In setting the learning rule, two rules of thumb should be followed. First, if the learning rate is 0, then no learning will be accomplished. Second, it would not be typical to set learning rates greater than 1, although the user is free to explore the behavior of the network when this is done. The learning rate can be set in two different ways. One is to left-click on the arrow of the slider tool that is beside the value, hold the mouse button down, and use the mouse to slide the value of the learning rate up or down. The other is to select the box in which the learning rate is displayed, and to type in the desired learning rate.

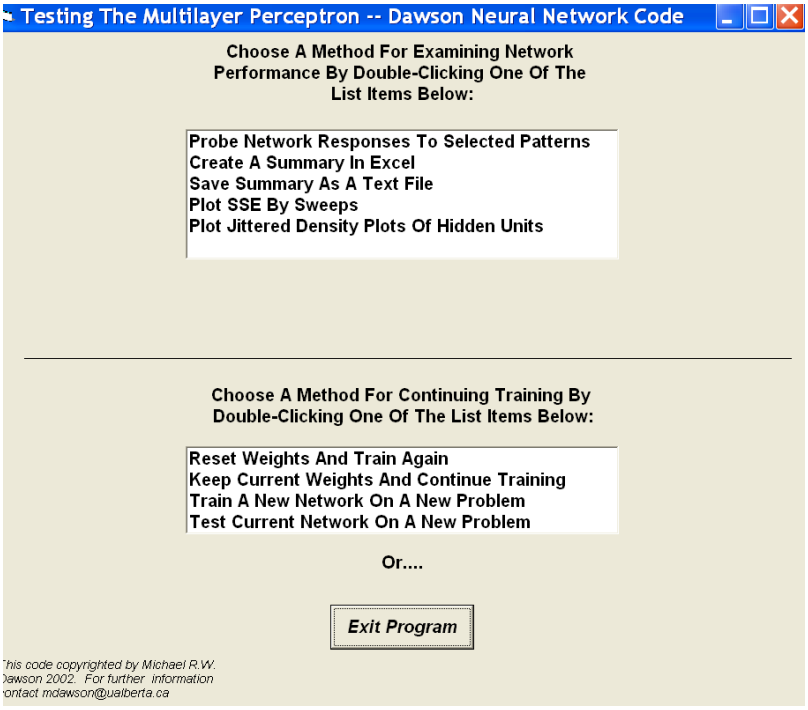
The fourth is a tool for specifying the minimum level of error (that is, SSE) to define a "hit". The default value for this setting is 0.01. With this setting, this means that if the desired value of an output unit is 1.00, then if the unit generates activity of 0.9 or higher, a "hit" will have occurred. This is because 1.00 – 0.9 = 0.1, and the square of 0.1 is 0.01. Similarly, if the unit generates activity of 0.1 or smaller for a desired output of 0.00, then a "hit" will have occurred. If a more conservative definition of "hit" is desired, then this tool should be used to make the minimum SSE value smaller. If a more liberal definition is required, then this value should be made larger. The smaller the value, the longer it will take learning to occur. However, if this value is too large, learning will end quickly, but the network's responses to stimuli will be less accurate.



Once these tools have been used to select the desired training parameters, associations (memories) can be stored in the network by pressing the "Start Training" button with a left-click of the mouse. When this is done, new boxes appear on the form to show the user how training is proceeding (see the figure above). When training stops, two new buttons appear on the form. By pressing the "Continue Training" button, more training occurs using the settings that have already been selected on this form. By pressing the "Test Recall" button, the user moves to a new form that can be used to explore the performance of the trained network. The details of this form are described below. Of course, pressing the "Exit" button terminates the program. Note that as training proceeds, information about the number of sweeps, the total network SSE, and the number of hits and misses is displayed. In the preceding figure, training stopped after 105 epochs because there were 432 hits and 0 misses on the training patterns for the monk1 problem.

TESTING WHAT THE NETWORK HAS LEARNED

Once training has been completed, the perceptron has learned to classify a set of input patterns. With the press of the "Test Recall" button of the form that has just been described, the program presents a number of options for examining the ability of the network to retrieve the information that it has stored. Some of these options involve the online examination of network responses, as well as the plotting of learning dynamics. Other options permit the user to save properties of the network in files that can be examined later. One of these file options enables the user to easily manipulate network data, or to easily move the data into another program (such as a statistical analysis tool) for more detailed analysis (e.g., factor analytic analysis of final connection weights).

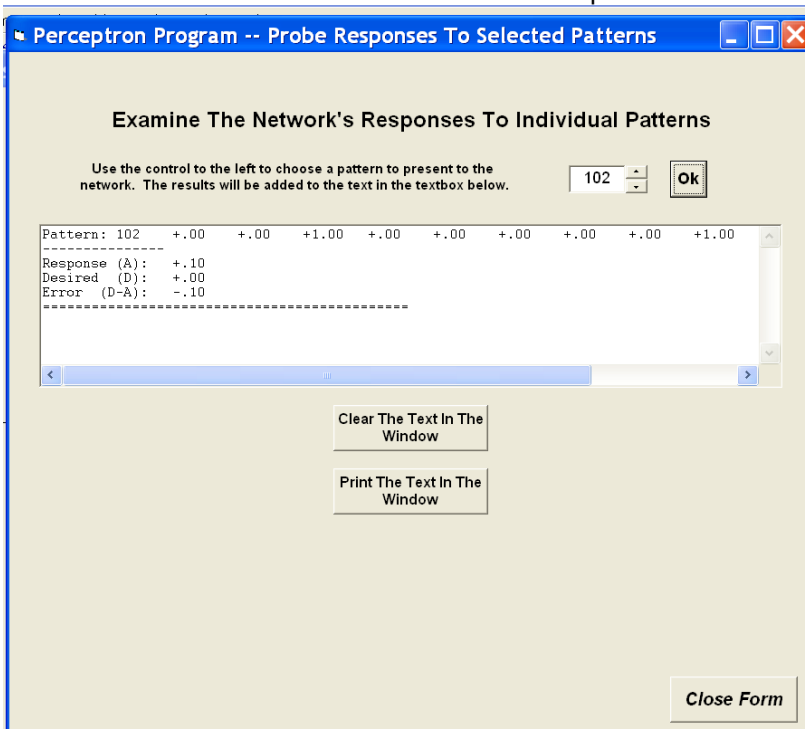


The "Test Recall" causes the program to present a form to the user that permits him or her to do two general types of activities. The first is the study/saving of network properties, which is described in more detail below. The second is the ability to return to previous forms to either continue network training on the same problem, or to read in a new problem for training and study. For either of these two classes of activity, the user selects the specific activity to perform from either list that is illustrated in the figure on the right. Double-clicking the list item with the left mouse button results in the activity being carried out. The sections that follow first describe the different activities that are possible by selecting any one of the four actions laid out in the control box on the upper part of the form. Later sections describe the result of double-clicking any one of the three actions made available in the control box on the lower part of the form. Again, an "Exit Program" is also provided to allow the user to exit the program from this form.

Testing Responses To Individual Patterns

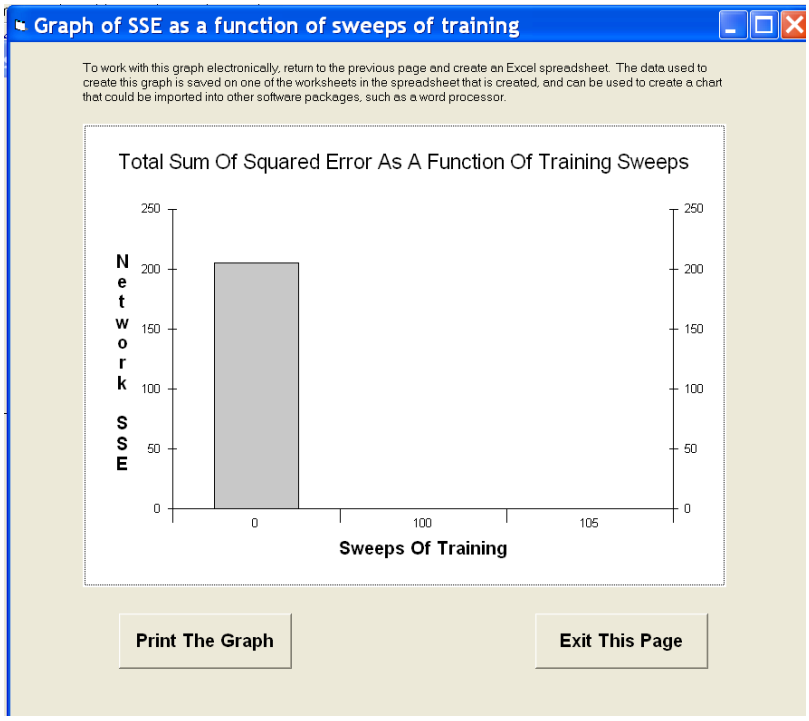
After the network has learned some classifications, it may be of interest to the user to examine the particular responses of the network to individual cue patterns in the training set. For instance, in cases

where the network is not performing perfectly, it could be that it is responding correctly to some cues, but not to others. By double-clicking on the list item "Probe Network Responses To Selected Patterns", the user causes the program to provide a form that allows the network to be tested one cue pattern at a time.



The form that permits this is depicted on the right. The form provides a large window in which network behavior is printed. When the form is initially presented, this large window is blank. Left-button mouse clicks on the arrow controls at the top of the form are used to select the number of the pattern to be presented to the network. When the desired pattern number has been selected, the "Ok" button is pressed. The cue pattern is then presented to the network, and the network's response is displayed. The display provides details about the cue pattern, the actual network response, the desired network response, and the error of the network. For instance, in the illustration, Pattern 102 of the monk1 problem has just been presented to the network.

More than one pattern can be tested in this way. The new pattern information is always displayed on top of previous pattern information. One can use the two scroll bars on the window to examine all of the information that has been requested. At any point in time, one can send this information to the system's default printer by pressing the button for printing. Also, one can erase the window by pressing the button for clearing the display. When the "Close Form" button is pressed, this form closes, and the user is back to the "Test Recall" list options.



Plotting Learning Dynamics

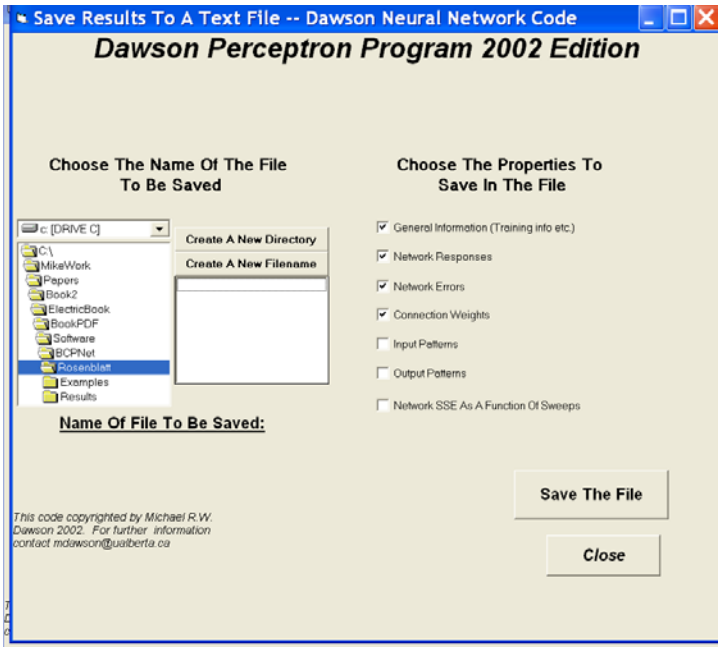
A comparison of the three learning rules for the perceptron might require examining how network error changes as a function of epochs of training. If the user chooses the "Plot SSE By Sweeps" option from the list in the network testing form, then the program automatically plots this information using a bar chart. One can import this chart directly into a word processing document by simultaneously pressing the "Alt" and "Print Screen" keys on the keyboard (which copies the active window into the clipboard), going to the document, and

pasting the clipboard into the document. One can print this chart on the default printer by left-clicking the mouse over the "Print The Graph" button. A left-click of the "Exit This Page" button closes the graph, and returns the user to the page that provides the options for testing network performance.

With respect to the graph produced in this form, the SSE axis is computed automatically, and the sampling of the bars across the Sweeps axis is determined by the choice of epochs between printouts made by the user on the program's second form. If the graph doesn't look quite right, then the user might consider re-running the simulation with a different choice for epochs between printouts. If a different kind of graph is desired, then the user might wish to save the network data to file. The data used to create this graph can be saved when this is done, and imported into a different software package that can be used to create graphs of different appearance.

In the example on the right, this particular graph is not interesting, because the network converged so quickly. In a case like this, if one were interested in the dynamics of learning, then one would re-train the network after setting the number of epochs between printouts to a smaller value.

Saving Results In A Text File



One of the options for storing information about network performance is to save network results as a text file. The form that permits this to be done, illustrated on the right, is accessed by choosing the list item "Save Summary As A Text File" from the "Test Network" page.

There are two sets of controls on this form. The first is a set of drive, directory, and file control boxes that are very similar to those found on the very first form seen when the program starts to run. One uses the drive and directory controls to navigate to a folder in which network data is to be saved. If it is necessary to create a new folder, a left-click of the mouse on the "Create A New Directory" button creates a dialog that permits the new directory to be named and created. Once the desired directory has been opened, the existing text files (.txt) in it are displayed. This is because the network data will be saved in such a file. One can overwrite an existing file by double-clicking it with the left mouse button. If a new file needs to be created, the dialog for doing so is accessed by a left-click of the mouse on the "Create A New Filename" button.

After choosing the location in which information is to be saved, the check boxes on the right of the form are set to determine what kinds of information will be saved. Appendix 1 provides an example of the kind of information that is saved in a file if all of the check boxes have been selected. If a check box is not selected, then the corresponding information is simply not written to the file. To save the file, after the desired check boxes have been selected, the user left-clicks the "Save The File" button with the mouse. The form remains open after this is done, because in some instances the user might wish to save different versions of the network information in different locations. This form is closed by a left-mouse click on the "Close Button", which returns the user to the "Test Network" form.

Saving Results In An Excel Workbook

A second method for saving network performance is to save it in a structured Microsoft Excel workbook. This option is only available in the Rumelhart program, and has been removed from RumelhartLite. It should obviously only be selected by users who also have Microsoft Excel installed on their computer. It is selected by a double-click of the "Create A Summary In Excel" list item that is offered in the "Test Network" form.

When this item is selected, a patience-requesting message is displayed on the "Test Network" form, and a number of different programming steps are taken to build an Excel Worksheet. When this is completed, the Worksheet is displayed as its own window, which will open on the user's computer in front of any of the Rumelhart program's windows. If the worksheet has been created successfully, then the user should see something similar to the screen shot that is presented below.

Microsoft Excel - Book1

File Edit View Insert Format Tools Data Window Help

A1 = Multilayer PROGRAM

	A	B
1	Multilayer PROGRAM	
2	Results Of Training With File:	monk1.net
3	Date of Analysis:	26/02/2003
4	Time of Analysis:	11:10:38 AM
5	Type Of Network:	Value
6	Learning Rate:	0.01
7	Sweeps Of Training:	105
8	Hits:	432
9	Misses:	0
10	Minimum Squared Error Defining A Hit:	0.01
11	Weight Start Settings	
12	Maximum:	0.1
13	Minimum:	0
14	Sign Option:	Both
15	Bias Start Settings	
16	Maximum:	0
17	Minimum:	0
18	Sign Option:	Both
19		
20		

All of the possible information that could be saved in the text version of a saved network is saved on this spreadsheet. Each different class of information is saved on its own worksheet in this Excel workbook. One can view different elements of this information by using the mouse to select the desired worksheet's tab on the bottom of the worksheet. The worksheet opens (as illustrated on the left) with the "General Information" tab selected.

When this workbook is open, it is running in Excel as a standalone program that is separate from the Rumelhart software. One can select different tabs in the worksheet to examine network properties. For example, in the figure below, the "Hidden Unit Weights" tab has been selected. After examining the worksheet, the user might wish to save it to disk. This is done by using the Save File utilities from Excel.

One problem with having this information being displayed with a completely separate program is that it begins to use up memory resources on the computer that cannot be directly controlled by either program. For instance, it is possible to leave this workbook open, and to return to the Rumelhart program. This practice is not recommended. Instead, potential system crashes are likely to be avoided by closing the Excel workbook before returning to Rumelhart. When Rumelhart is returned to, the "Test Network" form will still be displayed.

Microsoft Excel - Book1

File Edit View Insert Format To

A1 = Patte

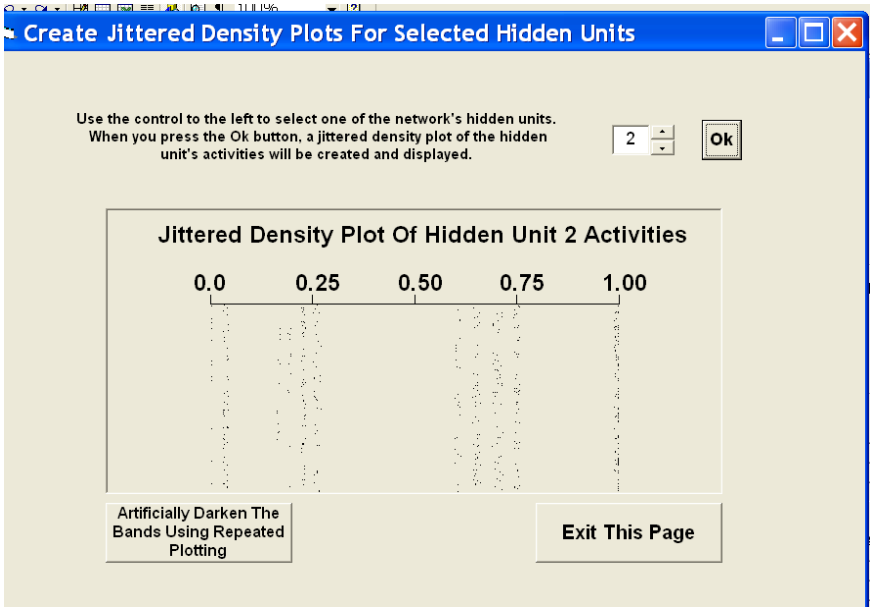
	A	B	C
1	Pattern	HID 1	HID 2
2	Type	Value	Value
3	Bias	-0.04	1.04
4	IN 1	0.25	0.32
5	IN 2	0.55	0.70
6	IN 3	-0.25	-0.32
7	IN 4	0.29	0.38
8	IN 5	0.09	-0.08
9	IN 6	0.00	0.00
10	IN 7	0.00	0.00
11	IN 8	0.00	0.00
12	IN 9	-0.33	0.35
13	IN 10	-0.33	0.36
14	IN 11	0.03	-0.03
15	IN 12	0.00	0.00
16			

If saving Excel files from Rumelhart causes system crashes, it is likely because of memory resource conflicts. The Excel options were built into Rumelhart because they provide a convenient format for working with network data after training has been accomplished. For instance, many of the results that are provided in Chapters 11 and 12 of *Minds And Machines: Connectionism And Psychological Modeling* were created by selecting a table from an Excel worksheet, copying it, and pasting it directly into a Microsoft Word document. The Excel data can also be easily copied and pasted into statistical packages like Systat. However, the Excel capability is not required for the distributed associative memory software to be used productively. If Excel problems are encountered frequently on your computer, our recommendation is to use RumelhartLite instead, and save network performance as text files only.

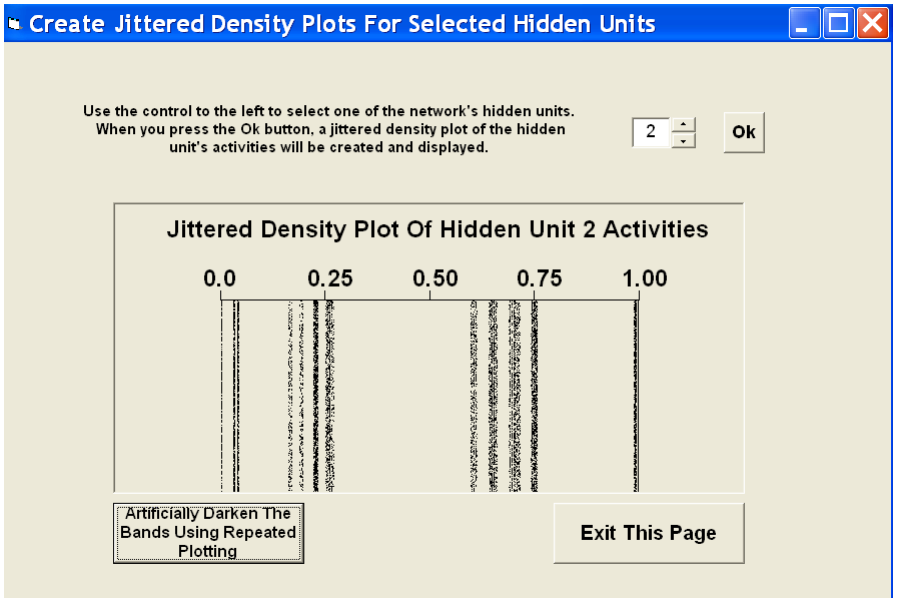
Inspecting Jittered Density Plots

When value units are used, one important characteristic they have is the banding of the jittered density plot of their hidden units. What this banding is, and its importance to network interpretation, is discussed in *Minds And Machines: Connectionism And Psychological Modeling*. The Rumelhart program comes with a tool that lets the user quickly inspect the jittered density plots for each hidden unit, to determine whether banding exists. This might be an important consideration in deciding whether to save the results of a network for later analysis. You can access this tool choosing the list item "Plot Jittered Density Plots Of Hidden Units" from the "Test Network" page.

When this list item is selected, a mostly blank form is created. In the top right of this form is a number tool and an "OK" button. Use the number tool to select a hidden unit. When the "OK" button is pressed, the form is filled with a jittered density plot. This is illustrated on the right, where the plot for Hidden Unit 2 of the monk1 network has been created. This particular example indicates that several different bands have appeared in this unit.



In some instances, the bands may be faint, because there is a small number of patterns in the training set. To artificially deal with this problem, one can press the "Artificially Darken The Bands Using Repeated Plotting" button. This causes the density plot to be plotted again, with different random values, on the same plot. In the example on the right the bands on the first plot have been darkened by pressing this button just once.



In using this tool, it should be cautioned that the bands that appear due to "artificial darkening" are not real. This tool is just a visualization aid. It is possible that this tool might suggest that some bands exist when they are not actually present. Whenever banding analysis is done on saved network data, it will only be performed on the actual network data – not on "artificially darkened" data.

Leaving The "Test Network" Form

Once the user has finished examining the performance of a trained network, the list at the bottom of the "Test Network" form provides different options for network training. If the "Reset Weights And Train Again" option is selected, then all of the connection weights are randomized, the network is readied to be trained on the same problem that it has just learned, and the user is returned to the form that permits training parameters to be selected. If the "Keep Current Weights And Train Again" option is selected, the network is trained on the same problem, but the weights created from the learning that was just completed are not erased. The user is returned to the form that permits training parameters to be selected. They must be set again if settings other than the default settings are desired. If the "Train A New Network On A New Problem" option is selected, then the user is returned to the program's first form to be able to read in a new problem for training. If the "Train The Current Network On A New Problem" is selected, then the user can read in a new problem, but it will be presented to the network with the weights preserved from the previous training. This option can be used to study the effect of pretraining on learning a new problem, generalization of learning, or savings in learning. If none of these options are desired, then you can close the program by pressing the "Exit Program" button with a left-mouse click.

CREATING NEW TRAINING FILES

When Rumelhart is installed on your computer, a few example files for training the distributed associative memory are also included. Several of these files were used in the examples that are described in Chapter 11 of *Minds And Machines: Connectionism And Psychological Modeling*. However, it is quite likely that the user might wish to study the performance of the distributed associative memory on different problems. In this section of the manual, we describe the general properties of the .net files that are used to train a network. We then describe the steps that the user can take to define their own training sets for further study.

General Structure Of A .net File

In Appendix 1 of this manual, the reader will find a copy of a network's performance when trained on the file monk1.net with a network of value units that uses two hidden units. The first step of training this network is to read in the file monk1.net, which contains three types of information: General network properties, the set of input patterns, and the set of desired patterns. Because of its size, only some of this file is given below:

```
1
2
12
432
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 1 0 0
.
.(this continues for all 432 input patterns)
.
1 0 1 0 0 1 1 0 1 1 0 1
1
1
1
.
.(this continues for all 432 desired outputs)
.
1
```

This information is structured into three different categories, which are highlighted in different colors to aid description. The first category (highlighted in yellow) consists of the first four rows in the file. These rows define the number of processing units in the network, and the number of patterns in the training set. The first number indicates the number of output units (1 in this case). The second number indicates the number of hidden units (2 in this case). The third number provides the number of input units (12). The fourth number provides the number of training patterns (432).

The second category of information (blue) in the file is the set of input patterns. Each input pattern is given its own row. Input pattern 1 occupies the first row, input pattern 2 occupies the second row, and so on. Because the initial information in the file indicates that there are 432 different training patterns in this training set, there are 432 different rows in this section of the file. For the preservation of space, not all are shown. Each row provides the value that will be input, as a cue, to each of the 12 input units used in this network. The first value in the row will be given to input unit 1, the second will be given to input unit 2, and so on. Each of these values is separated from the others by a "space" character.

The third category of information (gray) in the file is the set of output patterns. The first row of this part of the file represents the first output pattern, which is to be associated with the first input pattern from the previous information category. These second row represents the second output pattern, which is to be associated with the second input pattern, and so on. The format of each output pattern row is the same as that used for each input pattern row.

The reason that the input patterns and the output patterns are given different sections of the file, instead of appearing on the same row, is a historical convention. It does permit fairly easy modification of training sets, however. For instance, the same input patterns can be paired with a completely new set of output patterns by saving a copy of a .net file, opening it with an editor, selecting the existing output patterns, and pasting in a new set of desired outputs.

Creating Your Own .net File

All that one needs to do to create their own training set for the Rumelhart program is to create a text file that has the same general characteristics as those that were just described. The steps for doing this are:

1. Decide on a set of input pattern/output pattern pairs of interest
2. Open a wordprocessor (e.g., the Microsoft Notepad program) to create the file
3. On separate lines, enter the number of output units, hidden units, input units, and training patterns
4. On separate rows, enter each input pattern. Remember to separate each value with a space
5. On separate rows, enter each output pattern. Remember to separate each value with a space
6. Save the file as a text file
7. In Windows, rename the file to end with the extension .net instead of the extension .txt. Remember that the Rumelhart program will only read in files that have the .net extension.
8. Use the Rumelhart program to explore how a multilayer perceptron copes with the training set that you have created.

APPENDIX 1: MONK1.TXT

The information provided below is a copy of the file monk1.txt. This provides an example of the information that is saved in a text file when some of the checkboxes in the "Save File" form have been selected.

```
Multilayer Perceptron Training Program
==========================================
```

```
Results Of Training With File: monk1.net
Date Of Analysis: 26/02/2003
Time Of Analysis: 11:35:21 AM
==========================================
 Network Type: Value
Learning rate: 0.01
Training completed after 105 epochs
Settings For Initial Random Weights:
Maximum value: 0.1
Minimum value: 0
   Sign value: Both

Settings For Initial Random Biases:
Maximum value: 0
Minimum value: 0
   Sign value: Both

Pattern randomization during an epoch: True
==========================================

Connection weights from hidden units (rows) to output units (columns):
---------------
        Out 1
OutType Value
Bias    +1.55
HID 1   +1.59
HID 2   +1.48

Connection weights from input units (rows) to hidden units (columns):
---------------
        Hid 1   Hid 2
HidType Value   Value
Bias    -.04    +1.04
INP 1   +.25    +.32
INP 2   +.55    +.70
INP 3   -.25    -.32
INP 4   +.29    +.38
INP 5   +.09    -.08
INP 6   +.00    +.00
INP 7   +.00    +.00
INP 8   +.00    +.00
INP 9   -.33    +.35
INP 10  -.33    +.36
INP 11  +.03    -.03
INP 12  +.00    +.00
==========================================
```