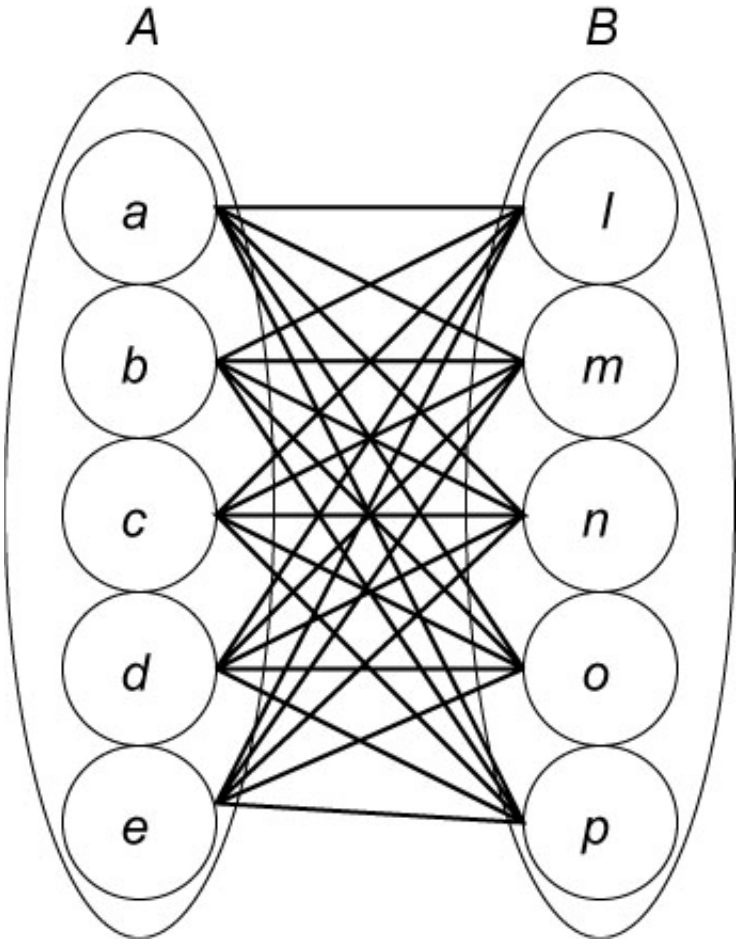
User Manual For The James and JamesLite Distributed Associative Memory Programs

Michael R.W. Dawson and Vanessa Yaremchuk
November 4, 2002
Biological Computation Project
University of Alberta
Edmonton, Alberta, Canada
http://www.bcp.psych.ualberta.ca



INTRODUCTION

James is a program written in Visual Basic 6.0 for the demonstration and exploration of distributed associative memory. 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 in preparation by Michael R.W. Dawson. This manuscript has the working title *Minds and Machines: Connectionism and Psychological Modeling,* and has been accepted for publication by Blackwell Publishing. Michael Dawson and Vanessa Yaremchuk programmed the current version of James. A second program, JamesLite, is identical to James with the exception that it does not include the capability to save network results in Microsoft Excel workbooks. In this document, James will be the only program referred to, as the user interface for it is identical to the interface for JamesLite. Both programs are distributed as freeware from the following website:

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

The purpose of the distributed associative memory is to learn associations between pairs of patterns. During training, a cue pattern is presented to the memory's input units, and a recall pattern is presented at the same time to the memory's output units. A learning rule is then used to modify the network's connection weights to store the association between the two patterns. Later, when training has been completed, a cue pattern is presented to the input units of the memory. This causes signals to be sent through the network's connection weights, which in turn produce activity in the network's output units. If proper learning has occurred, then the reproduced activity in the output units should be similar to, if not identical to, the recall pattern that was originally presented with the cue pattern. This kind of memory is distributed in the sense that one set of connection weights can store the associations between several different pairs of patterns.

INSTALLING THE PROGRAM

James 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 James.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 James.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 James.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.

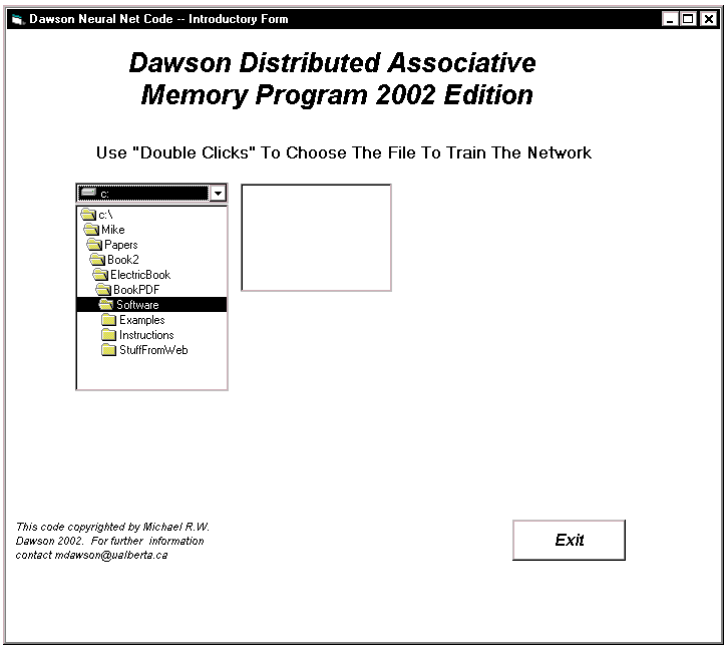
TEACHING A DISTRIBUTED MEMORY

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 "James.exe". Second, one can go to the start button on the computer, choose programs, scroll to the program group BCPNet, and select the program James.exe.

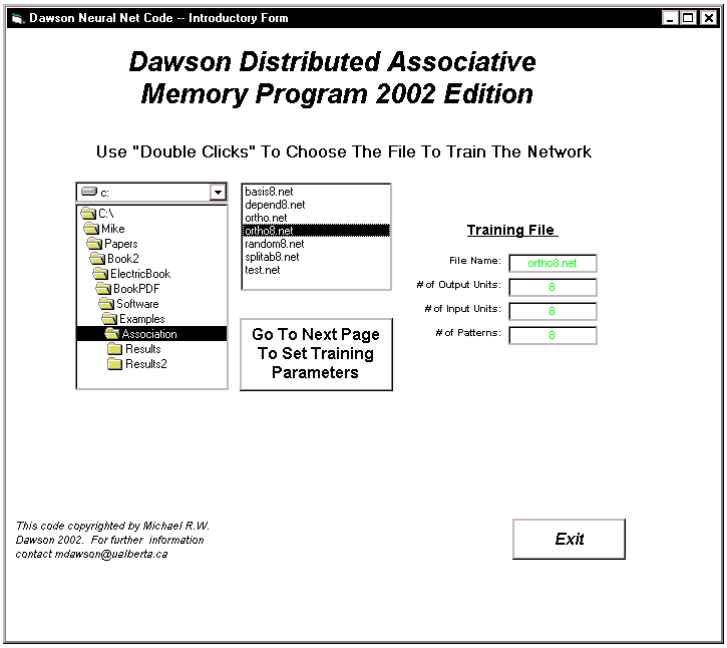
Loading A File To Train A Network

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 James 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 "ortho8.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.

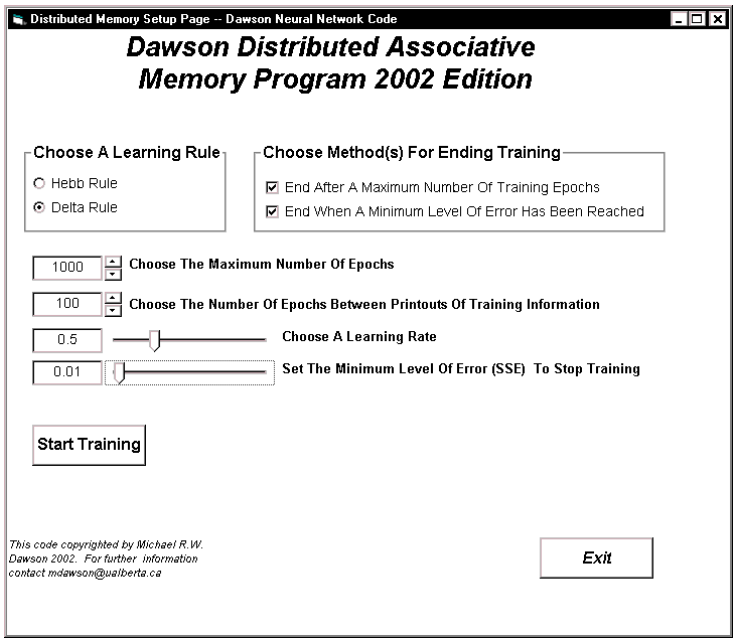


In this example, if "ortho8.net" has been selected, but is not really the file that is desired, one can simply go back to the file selection tools and choose another file. When its file name is double-clicked, the new file will be read in, and will replace the properties of the previous (undesired) file.

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 9 of *Connectionism And Psychological Modeling*. A sample copy of this chapter is available from the website that delivered this manual and the software http://www.bcp.psych.ualberta.ca/~mike/Book2/).



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 distributed associative memory. The first tool is used to choose which of two learning rules, the Hebb rule or the delta rule, will be used to modify the connection weights of the network. The default rule is the Hebb rule. A left-click of the mouse on this tool is all that is required to select the learning rule.

A second tool is used to choose a method for stopping training. In the first method, training stops after a maximum number of epochs (set by the user) has been reached. In the second method, training stops when the sum of squared error (SSE) for the network drops below a minimum level (also specified by the user). 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.

There are two suggestions for setting this aspect of training. First, the user should always set a maximum number of epochs for training to end, just as a precautionary measure. This is why this method is selected as a default. Second, ending processing by using SSE is not recommended when the Hebb rule is being used for training, as this learning rule is not explicitly designed to minimize error.

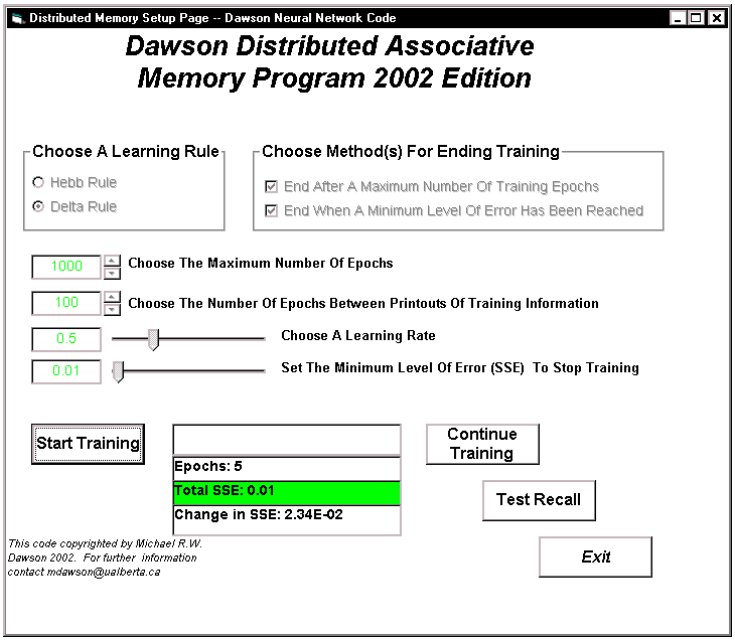
The four remaining tools on the form are used to set numerical values that control training.

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! In other words, just changing this number is not enough to ensure that the program will use it!

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 by either learning rule. More details on the role of learning rate in the equations can be found in Chapter 9 of *Connectionism And Psychological Modeling*. The learning rate is used for either learning rule. 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 network error (that is, SSE) to control termination of learning. This value can be set using the same methods described for the previous tool. The default value for this parameter is 0.5. If this value is set to a smaller value, then the user is requiring the network to generate more accurate responses before learning is ended. If this value is increased, then the user is permitting the network to end training when a larger amount of error is evident in the network's responses. When the delta rule is used to train the network, the smaller this value is set, the longer it will take the network to converge. It is possible to set this value to 0, and to therefore require the network to generate perfect responses. However, in some cases the network will be incapable of achieving this degree of performance (e.g., for linearly dependent training sets), and training will therefore never stop unless a maximum number of training epochs has also been selected. Note that it if the user chooses this parameter, then the "End When A Minimum Level Of Error Has Been Achieved" option 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! In other words, just changing this number is not enough to ensure that the program will use it.

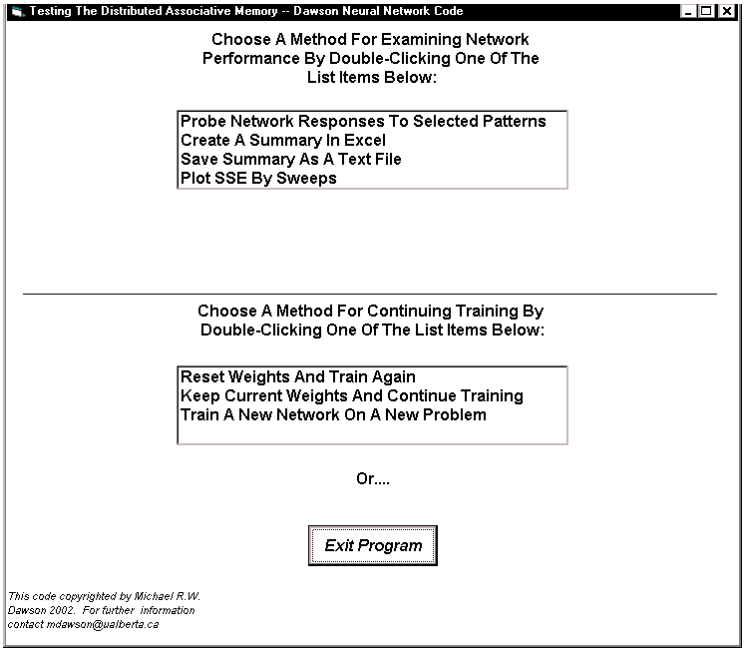
Once these tools have been used to select the desired training parameters, associations (memories) can be stored in the network by pressing the "Start Training" button with a left-click of the mouse. When this is done, new boxes appear on the form to show the user how training is proceeding. When training 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.

TESTING WHAT THE MEMORY HAS LEARNED

Once training has been completed, the distributed associative memory has stored a set of associations between pairs of 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

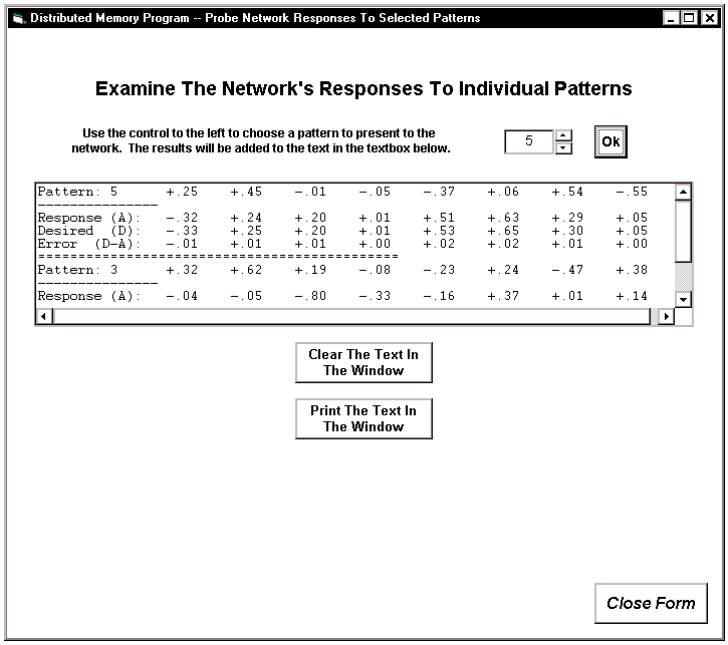
more detail below. The second is the ability to return to previous forms to either continue network training on the same problem, or to read in a new problem for training and study.



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

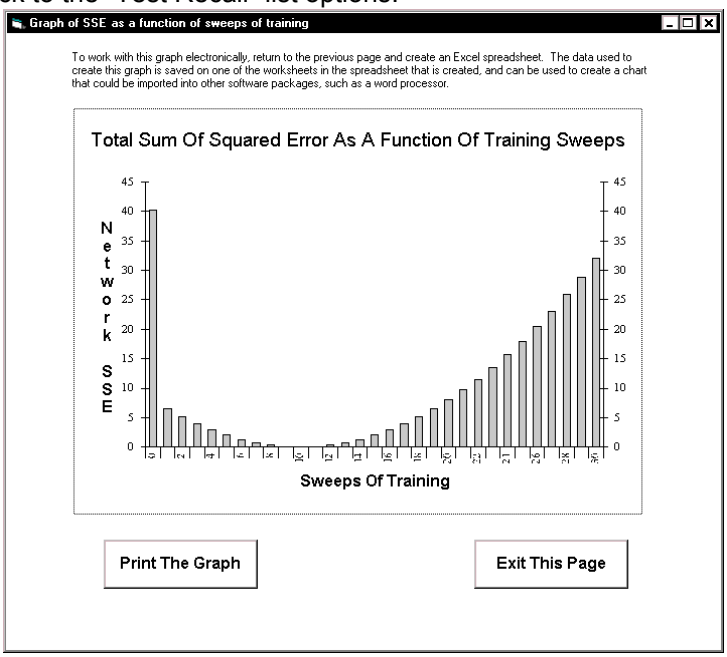
After the network has learned some associations, it may be of interest to the user to examine the particular responses of the network to individual cue patterns in the training set. For instance, in cases where the network is not performing perfectly, it could be that it is responding correctly to some 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 5 has just been selected for presentation to the network.

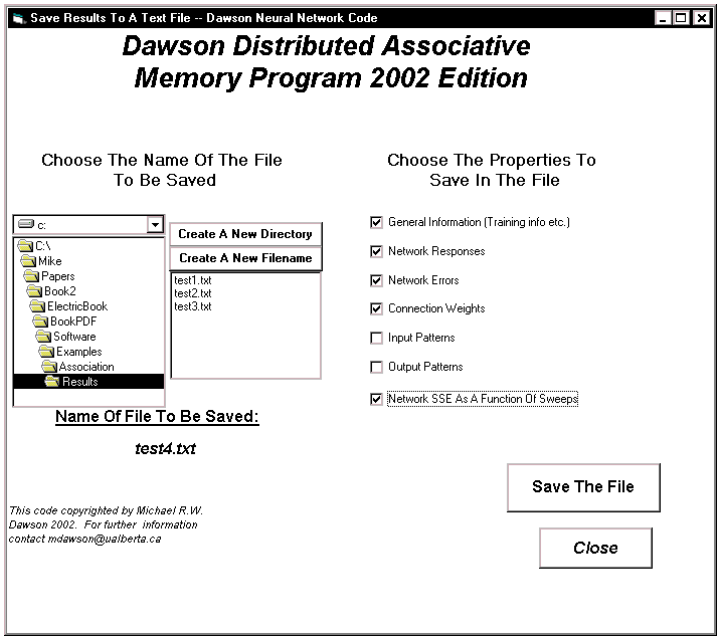
More than one pattern can be tested in this way. The new pattern information is always displayed on top of previous pattern information. For example, in the figure above, the Pattern 5 information was requested immediately after studying the network's responses to Pattern 3. One can use the two scroll 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



In Chapter 9 of *Connectionism and Psychological Modeling*, much of the comparison of the two learning rules for the distributed associative memory depends upon an examination of 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.



Saving Results In A Text File

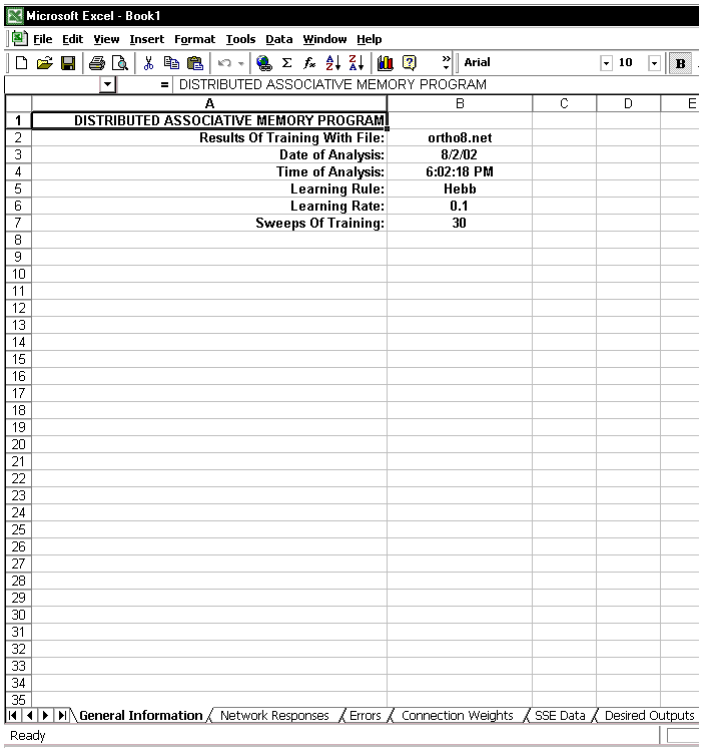
One of the options for storing information about network performance is to save network results as a text file. The form that permits this to be done, illustrated on the right, is accessed by choosing the list item "Save Summary As A Text File" from the "Test Network" page.

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

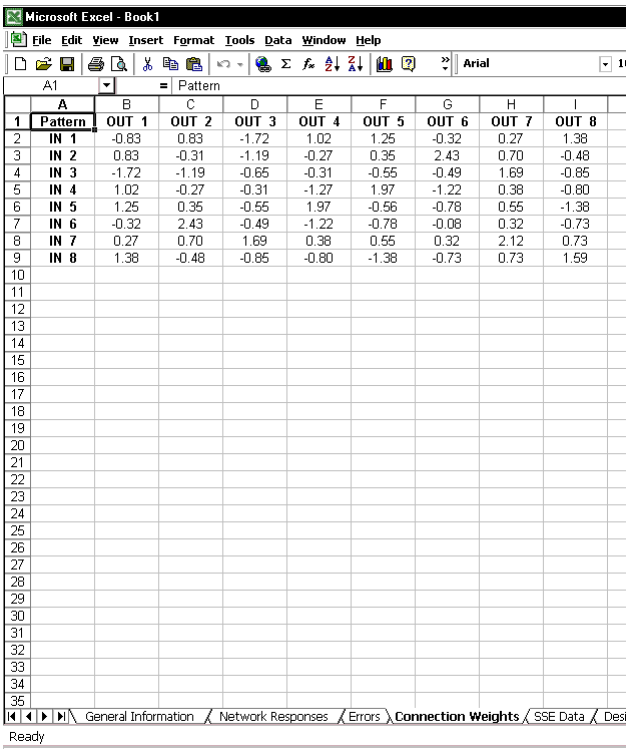
A second method for saving network performance is to save it in a structured Microsoft Excel workbook. This option is only available in the James program, and has been removed from JamesLite. It should obviously only be selected by users who also have Microsoft Excel installed on their computer. It is 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 James program's windows. If the worksheet has been created successfully, then the user should see something similar to the screen shot that is presented on the right.

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 above) 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 James software. One can select different tabs in the worksheet to examine network properties. For example, in the figure on the right, the "Connection 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.



Pattern	OUT 1	OUT 2	OUT 3	OUT 4	OUT 5	OUT 6	OUT 7	OUT 8
IN 1	-0.83	0.83	-1.72	1.02	1.25	-0.32	0.27	1.38
IN 2	0.83	-0.31	-1.19	-0.27	0.35	2.43	0.70	-0.48
IN 3	-1.72	-1.19	-0.65	-0.31	-0.55	-0.49	1.69	-0.85
IN 4	1.02	-0.27	-0.31	-1.27	1.97	-1.22	0.38	-0.80
IN 5	1.25	0.35	-0.55	1.97	-0.56	-0.78	0.55	-1.38
IN 6	-0.32	2.43	-0.49	-1.22	-0.78	-0.08	0.32	-0.73
IN 7	0.27	0.70	1.69	0.38	0.55	0.32	2.12	0.73
IN 8	1.38	-0.48	-0.85	-0.80	-1.38	-0.73	0.73	1.59

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 James program. This practice is not recommended. Instead, potential system crashes are likely to be avoided by closing the Excel workbook before returning to James. When James is returned to, the "Test Network" form will still be displayed.

If saving Excel files from James causes system crashes, it is likely because of memory resource conflicts. The Excel options were built into James because they provide a convenient format for working with network data after training has been accomplished. For instance, many of the tables that are provided in Chapter 9 of *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 JamesLite instead, and save network performance as text files only.

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 set to zero, 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 none of these options are desired, then the program can be closed by pressing the "Exit Program" button with a left-mouse click.

CREATING NEW TRAINING FILES

When James 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 9 of 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

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 ortho8.net via the Hebb rule. The first step of training this network is to read in the file ortho8.net, which contains the following information:

```
8
0
8
8
2.726E-01 3.901E-01 -3.082E-01 -1.565E-01 6.364E-01 -4.847E-01 9.673E-02 6.853E-02
3.505E-01 -2.166E-01 6.276E-02 -2.762E-01 2.920E-01 2.763E-01 -4.666E-01 -6.088E-01
3.171E-01 6.190E-01 1.921E-01 -8.109E-02 -2.313E-01 2.361E-01 -4.711E-01 3.762E-01
-3.283E-01 2.514E-01 2.031E-01 1.363E-02 5.287E-01 6.454E-01 2.975E-01 5.478E-02
2.504E-01 4.471E-01 -8.069E-03 -5.367E-02 -3.729E-01 6.236E-02 5.353E-01 -5.522E-01
-4.451E-02 -5.132E-02 -8.284E-01 -3.408E-01 -1.624E-01 3.804E-01 1.176E-02 1.475E-01
-5.174E-01 3.655E-01 -2.902E-01 4.328E-01 -5.472E-03 -9.601E-02 -4.129E-01 -3.839E-01
5.119E-01 -1.354E-01 -2.290E-01 7.656E-01 1.062E-01 2.421E-01 7.246E-02 7.728E-02
5.119E-01 -1.354E-01 -2.290E-01 7.656E-01 1.062E-01 2.421E-01 7.246E-02 7.728E-02
-5.174E-01 3.655E-01 -2.902E-01 4.328E-01 -5.472E-03 -9.601E-02 -4.129E-01 -3.839E-01
-4.451E-02 -5.132E-02 -8.284E-01 -3.408E-01 -1.624E-01 3.804E-01 1.176E-02 1.475E-01
2.504E-01 4.471E-01 -8.069E-03 -5.367E-02 -3.729E-01 6.236E-02 5.353E-01 -5.522E-01
-3.283E-01 2.514E-01 2.031E-01 1.363E-02 5.287E-01 6.454E-01 2.975E-01 5.478E-02
3.171E-01 6.190E-01 1.921E-01 -8.109E-02 -2.313E-01 2.361E-01 -4.711E-01 3.762E-01
3.505E-01 -2.166E-01 6.276E-02 -2.762E-01 2.920E-01 2.763E-01 -4.666E-01 -6.088E-01
2.726E-01 3.901E-01 -3.082E-01 -1.565E-01 6.364E-01 -4.847E-01 9.673E-02 6.853E-02
```

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 (8 in this case). The second number indicates the number of hidden units (0 in this case). The third number provides the number of input units (8). The fourth number provides the number of training patterns (again, 8). Note that even though there are no hidden units in this network, a digit specifying the number exists in this file. This is to make the files read by the James program compatible with other software packages that we are developing.

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 8 different training patterns in this training set, there are eight different rows in this section of the file. Each row provides the value that will be input, as a cue, to each of the 8 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 James 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 James program will only read in files that have the .net extension.
8. Use the James program to explore associative learning of the training set that you have created.

SOME EXERCISES FOR STUDYING DISTRIBUTED ASSOCIATIVE MEMORY

As was noted earlier, one of the primary purposes of the James and JamesLite programs is to provide students with a system that can be used to explore some of the properties of a distributed associative memory. This section of the manual provides some example exercises that can be performed with a small set of sample .net files that are provided along with the software when it is installed. In all of the examples below, it is assumed that the James program is being used. However, all of the examples can be performed with JamesLite – provided the user checks some of the results by examining the text files that are saved when the network has performed the desired tasks.

1. The purpose of this exercise is to demonstrate associative learning using the Hebb rule. Run the James program, and load the file "basis8.net". This file consists of 8 basis vectors (each pattern has one 1 in it, and all the rest of the pattern's units are equal to 0. This kind of pattern set is orthonormal. On the setup page, choose the Hebb rule, choose to end after a maximum number of sweeps, and set this maximum number of sweeps to 10. Have the program print out every epoch by setting the sweeps between printouts value to 1. Use a learning rate of 0.1; the minimum SSE value need not be changed, because it will not be used. Start training by pressing the appropriate button. **What is the total SSE when training ends?** Press the test recall button. Examine learning over time by plotting the SSEs. **What is the appearance of this graph?** Exit the graph, and then create an Excel spreadsheet. When the spreadsheet appears, examine the network error to each pattern. **What can be said about these errors?** Examine the connection weights that are displayed in the spreadsheet. **What is the relationship between the set of connection weights, and the input and output patterns whose associations are stored in the weights?** (To answer this question, remember you can take a look at the set of input patterns and the set of output patterns, because this information is stored in the spreadsheet as well.) Close the Excel spreadsheet without saving it. Then, to proceed to Exercise 2, tell the program to train a new network on a new problem.
2. The purpose of the second exercise is to demonstrate associative learning, accomplished via Hebb learning, with a training set that seems more complicated. Load the file ortho8.net. This file consists of 8 pairs of vectors that are mutually orthonormal, and thus is very similar to the training set used in Exercise 1. However, each processing unit has a negative or positive fractional value, so the patterns seem more complicated. **Use the same settings that were used in Exercise 1. What is the total SSE when training ends? How does this compare to the previous exercise? What is the appearance of the SSE by epochs graph?** Use the Excel spreadsheet to examine network errors, and to examine the connection weights. **How do these compare to those observed in Exercise 1? How might one analyze the connection weight matrix to**

tease out the associations that are stored distributively within it? Close the spreadsheet, and then ask the network to train a new network on a new problem.

3. Load in basis8.net, which was used in Exercise 1. Use exactly the same settings to train this network, with the exception that the maximum number of sweeps should be set to 30 instead of to 10. Examine SSE over time, and examine the network properties using Excel. **Compare and contrast the performance in this simulation to the performance observed in Exercise 1. What are the implications of this simulation to Hebb rule learning?** Close the spreadsheet, and then ask the network to train a new network on a new problem.
4. Load in ortho.net, which was used in Exercise 2. Use exactly the same settings to train this network, with the exception that the maximum number of sweeps should be set to 30 instead of to 10. Examine SSE over time, and examine the network properties using Excel. **Compare and contrast the performance in this simulation to the performance observed in Exercise 2. What are the implications of this simulation to Hebb rule learning?** Close the spreadsheet, and then ask the network to reset itself for continued training on this problem. Then proceed to Exercise 5.
5. The purpose of this exercise is to compare the performance of the Delta rule to the performance that has been observed with the Hebb rule. It still involves training ortho8.net. Set the learning rule to the Delta rule, and make sure that check marks are set beside both methods for ending training. Set the maximum number of sweeps to 100, and set the number of sweeps between printouts to 2. The learning rate can remain at 0.10, and the minimum SSE for ending training can remain at 0.05. Train the network. **At what epoch does training stop? What is the total SSE when this occurs?** Plot SSE as a function of epochs. **What is the appearance of this graph, and how does it differ from the graphs that have been examined in the previous exercises?** Summarize the network properties by using Excel. **What mistakes, if any, is the network making to the stimuli? What do the connection weights look like?** Close the spreadsheet, and ask the program to read in a new problem. Then proceed to Exercise 6.
6. Read in the file random8.net. First, using the settings from Exercise 1, train the memory with the Hebb rule on this problem. **Can it learn the problem? If so, how many sweeps are required?** When you decide to end training, examine the performance of the network using the SSE by epochs graph and Excel. **Does this information provide any insights into the kind of problems that the network is having, if it is having any?** Second, close the graph and the spreadsheet, and ask the program to reset the weights. Train the network on the same problem using the Delta rule, using the same settings that were used in Exercise 5. **Can it learn the problem? If so, how many sweeps are required?** When you decide to end training, examine the performance of the network using the SSE by epochs graph and Excel. **Does this information provide any insights into the kind of problems that the network is having, if it is having any?** Close the graph and spreadsheet, and ask the program to read in a new problem. Then proceed to Exercise 7.
7. Read in the file depend8.net. First, using the settings from Exercise 1, train the memory with the Hebb rule on this problem. **Can it learn the problem? If so, how many sweeps are required?** When you decide to end training, examine the performance of the network using the SSE by epochs graph and Excel. **Does this information provide any insights into the kind of problems that the network is having, if it is having any?** Second, close the graph and the spreadsheet, and ask the program to reset the weights. Train the network on the same problem using the Delta rule, using the same settings that were used in Exercise 5. **Can it learn the problem? If so, how many sweeps are required?** When you decide to end training, examine the performance of the network using the SSE by epochs graph and Excel. **Does this information provide any insights into the kind of problems that the network is having, if it is having any?** Close the graph and spreadsheet, and exit the program.

PROGRAM HISTORY, ISSUES, ETCETERA

Date	Comments
August 6, 2002	• James and JamesLite .zip files installed on website

	• User manual installed on website • Call for beta testing of software and software delivery made to select group of e-mail addresses (psychology, bcp, some individuals at U. of A. and Carleton)
August 7, 2002	• .zip files changed so that programs are stored in BCPNet program group, instead of James program group • minor revisions to user manual

APPENDIX 1: TEST4.TXT

The information provided below is a copy of the file test4.txt. This provides an example of the information that is saved in a text file when all of the checkboxes in the "Save File" form have been selected.

```
Distributed Associative Memory Program
==========================================
Results Of Training With File: ortho8.net
Date Of Analysis: 8/4/02
Time Of Analysis: 7:32:26 PM
==========================================
Learning rule: Hebb
Learning rate: 0.1
Training completed after 10 epochs
==========================================
Network Responses To Each Input Pattern:
---------------
Pattern 1      +.51    -.14    -.23    +.77    +.11    +.24    +.07    +.08
Pattern 2      -.52    +.37    -.29    +.43    -.01    -.10    -.41    -.38
Pattern 3      -.04    -.05    -.83    -.34    -.16    +.38    +.01    +.15
Pattern 4      +.25    +.45    -.01    -.05    -.37    +.06    +.54    -.55
Pattern 5      -.33    +.25    +.20    +.01    +.53    +.65    +.30    +.05
Pattern 6      +.32    +.62    +.19    -.08    -.23    +.24    -.47    +.38
Pattern 7      +.35    -.22    +.06    -.28    +.29    +.28    -.47    -.61
Pattern 8      +.27    +.39    -.31    -.16    +.64    -.48    +.10    +.07
==========================================
After training, sum of squared error was: 8.96756674041477E-08

Network Response Errors To Each Input Pattern:
---------------
Pattern 1      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
Pattern 2      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
Pattern 3      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
Pattern 4      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
Pattern 5      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
Pattern 6      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
Pattern 7      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
Pattern 8      +.00    +.00    +.00    +.00    +.00    +.00    +.00    +.00
==========================================
Connection weights from input units (rows) to output units (columns):
---------------
       Out 1   Out 2   Out 3   Out 4   Out 5   Out 6   Out 7   Out 8
INP 1   -.28    +.28    -.57    +.34    +.42    -.11    +.09    +.46
INP 2   +.28    -.10    -.40    -.09    +.12    +.81    +.23    -.16
INP 3   -.57    -.40    -.22    -.10    -.18    -.16    +.56    -.28
INP 4   +.34    -.09    -.10    -.42    +.66    -.41    +.13    -.27
INP 5   +.42    +.12    -.18    +.66    -.19    -.26    +.18    -.46
INP 6   -.11    +.81    -.16    -.41    -.26    -.03    +.11    -.24
INP 7   +.09    +.23    +.56    +.13    +.18    +.11    +.71    +.24
INP 8   +.46    -.16    -.28    -.27    -.46    -.24    +.24    +.53
==========================================
The set of input patterns was:
---------------
Pattern 1      +.27    +.39    -.31    -.16    +.64    -.48    +.10    +.07
Pattern 2      +.35    -.22    +.06    -.28    +.29    +.28    -.47    -.61
Pattern 3      +.32    +.62    +.19    -.08    -.23    +.24    -.47    +.38
Pattern 4      -.33    +.25    +.20    +.01    +.53    +.65    +.30    +.05
```

```
Pattern 5      +.25    +.45    -.01    -.05    -.37    +.06    +.54    -.55
Pattern 6      -.04    -.05    -.83    -.34    -.16    +.38    +.01    +.15
Pattern 7      -.52    +.37    -.29    +.43    -.01    -.10    -.41    -.38
Pattern 8      +.51    -.14    -.23    +.77    +.11    +.24    +.07    +.08
============================================
The set of desired outputs was:
---------------
Pattern 1      +.51    -.14    -.23    +.77    +.11    +.24    +.07    +.08
Pattern 2      -.52    +.37    -.29    +.43    -.01    -.10    -.41    -.38
Pattern 3      -.04    -.05    -.83    -.34    -.16    +.38    +.01    +.15
Pattern 4      +.25    +.45    -.01    -.05    -.37    +.06    +.54    -.55
Pattern 5      -.33    +.25    +.20    +.01    +.53    +.65    +.30    +.05
Pattern 6      +.32    +.62    +.19    -.08    -.23    +.24    -.47    +.38
Pattern 7      +.35    -.22    +.06    -.28    +.29    +.28    -.47    -.61
Pattern 8      +.27    +.39    -.31    -.16    +.64    -.48    +.10    +.07
============================================
Network SSE as a function of sweeps of training was:
---------------
Sweeps  Network SSE
0.00    8.00E+00
1.00    6.48E+00
2.00    5.12E+00
3.00    3.92E+00
4.00    2.88E+00
5.00    2.00E+00
6.00    1.28E+00
7.00    7.20E-01
8.00    3.20E-01
9.00    8.00E-02
10.00   8.97E-08
============================================
```