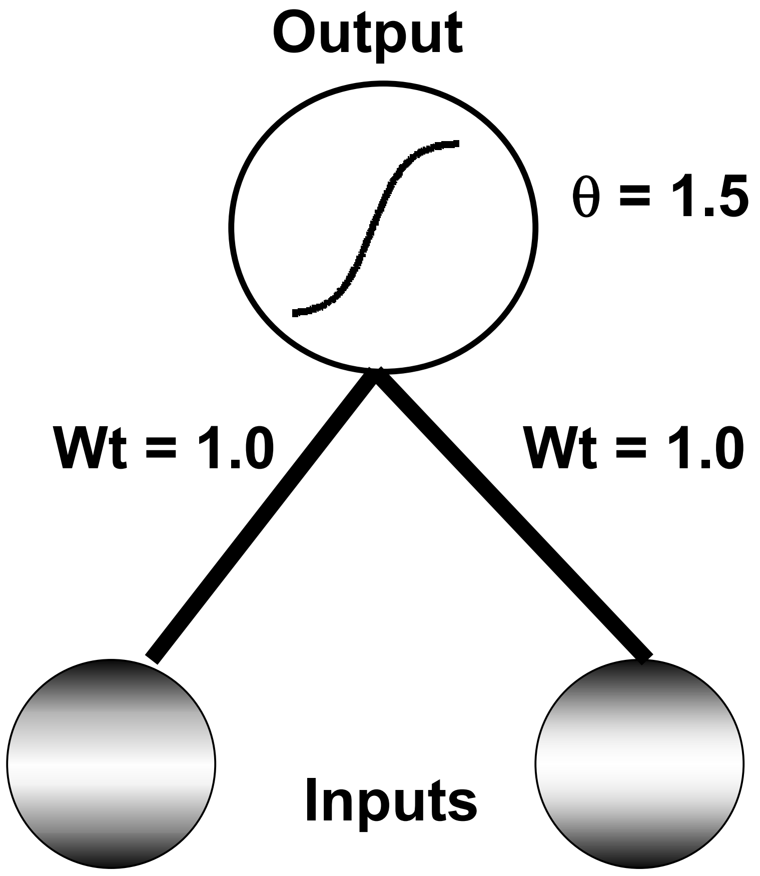
User Manual For The Rosenblatt and RosenblattLite Perceptron Programs

Michael R.W. Dawson and Vanessa Yaremchuk
November 5, 2002
Biological Computation Project
University of Alberta
Edmonton, Alberta, Canada
http://www.bcp.psych.ualberta.ca



INTRODUCTION

Rosenblatt is a program written in Visual Basic 6.0 for the demonstration and exploration of 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 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 Rosenblatt. A second program, RosenblattLite, is identical to Rosenblatt with the exception that it does not include the capability to save network results in Microsoft Excel workbooks. In this document, Rosenblatt will be the only program referred to, as the user interface for it is identical to the interface for RosenblattLite. Both programs are distributed as freeware from the following website:

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

The purpose of the perceptron program is to learn a set of stimulus/response associations, and in this respect it is very similar to distributed associative memories. However, the interpretation of these assocations is usually slightly different. First, unlike a distributed associative memory trained with the Hebb rule or the Delta rule, a perceptron uses a nonlinear activation function in its output units. As a result, the output units are generally trained to turn completely "on" or "off". This means that perceptron responses are usually interpreted as representing names or categories that are applied to stimuli. Thus, a perceptron is usually considered to be a pattern classification system.

The current program explores pattern classification with three different versions of the perceptron. In the first, the Rosenblatt training rule – which is equivalent to the Delta rule – is used to train a perceptron with a threshold activation function. In the second, a gradient descent learning rule is used to train a perceptron that uses a logistic activation function as a continuous approximation of the threshold activation function. In the third, a variation of the gradient descent learning rule is used to train a perceptron that uses a Gaussian activation function in its output units. This last function means that the output units in essence have two different thresholds, instead of one. These variations of the perceptron are described in more detail in Chapter 10 of the book for which this multimedia site has been constructed.

INSTALLING THE PROGRAM

Rosenblatt 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 Rosenblatt.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 Rosenblatt.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 Rosenblatt.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.

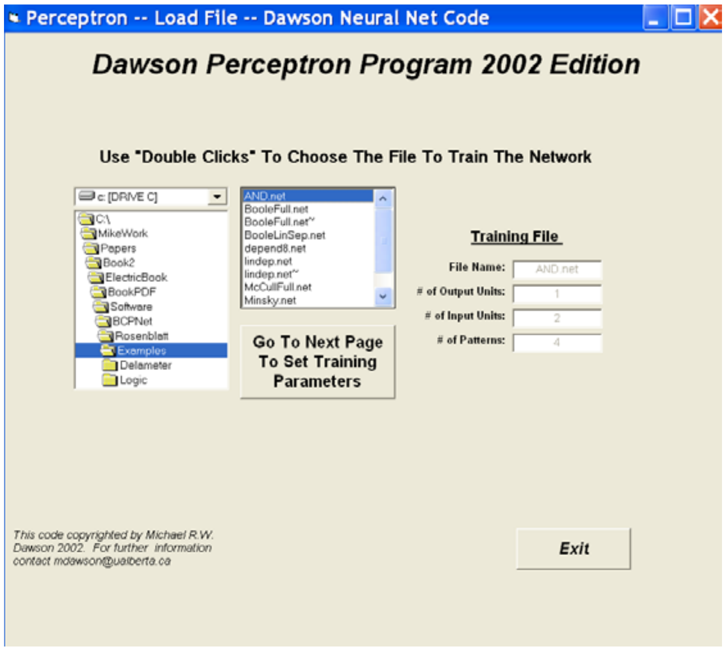
TRAINING A PERCEPTRON

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 "Rosenblatt.exe". Second, one can go to the start button on the computer, choose programs, scroll to the program group BCPNet, and select the program Rosenblatt.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 Rosenblatt program is designed to read, and only files that end in this extension can be used to train the network.


Perceptron -- Load File -- Dawson Neural Net Code

Dawson Perceptron Program 2002 Edition

Use "Double Clicks" To Choose The File To Train The Network

c: [DRIVE C]

C:\
MikeWork
Papers
Book2
ElectricBook
BookPDF
Software
BCPNet
Rosenblatt
Examples
Delameter
Logic

AND.net
BooleFull.net
BooleFull.net~
BooleLinSep.net
depend8.net
lindep.net
lindep.net~
McCullFull.net
Minsky.net

Training File

File Name:	AND.net
# of Output Units:	1
# of Input Units:	2
# of Patterns:	4

This code copyrighted by Michael R.W. Dawson 2002. For further information contact mdawson@ualberta.ca

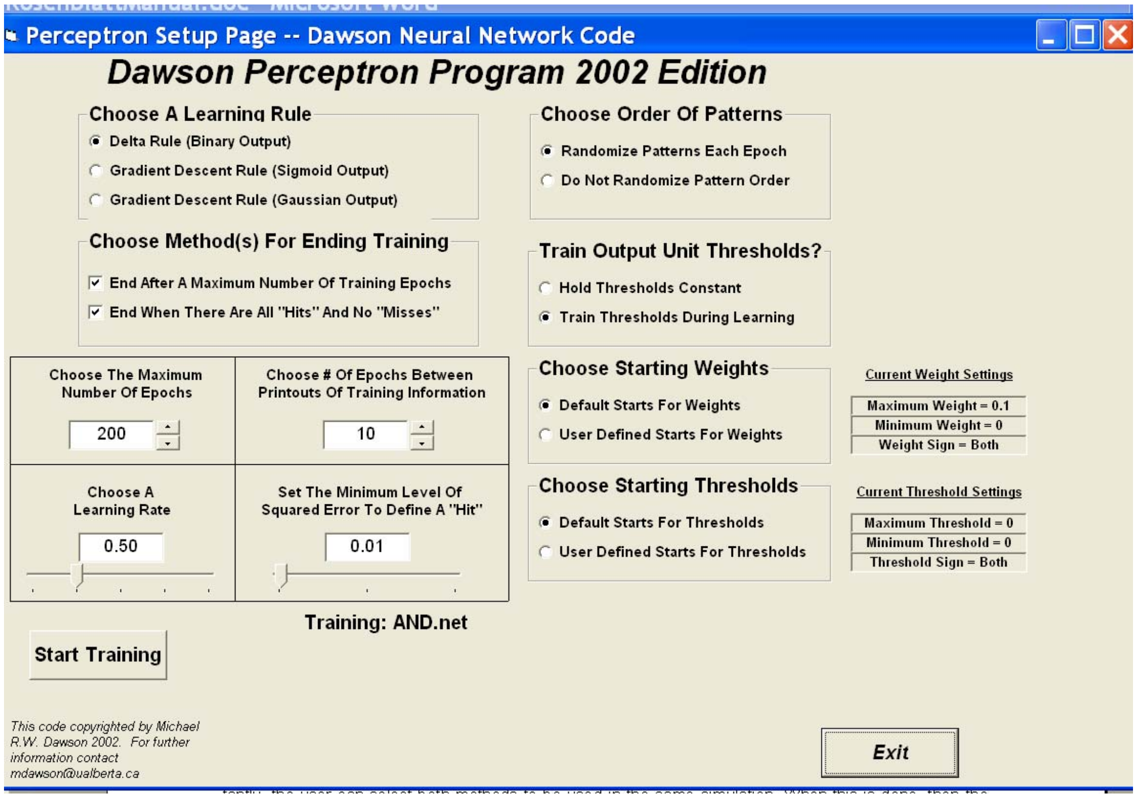
One chooses a .net file by double-clicking one of the file names that is displayed in the file selection box. When this is done, the program reads the desired file, some of the file's properties are displayed, and another button appears on the form. In the figure on the right, the file "AND.net" has 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 "AND.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 10 of *Connectionism And Psychological Modeling*.



Perceptron Setup Page -- Dawson Neural Network Code

Dawson Perceptron Program 2002 Edition

Choose A Learning Rule
- [✓ Delta Rule (Binary Output)]
- Gradient Descent Rule (Sigmoid Output)
- Gradient Descent Rule (Gaussian Output)

Choose Method(s) For Ending Training
- [✓ End After A Maximum Number Of Training Epochs]
- [✓ End When There Are All "Hits" And No "Misses"]

Choose Order Of Patterns
- [✓ Randomize Patterns Each Epoch]
- Do Not Randomize Pattern Order

Train Output Unit Thresholds?
- Hold Thresholds Constant
- [✓ Train Thresholds During Learning]

Choose The Maximum Number Of Epochs: 200

Choose # Of Epochs Between Printouts Of Training Information: 10

Choose A Learning Rate: 0.50

Set The Minimum Level Of Squared Error To Define A "Hit": 0.01

Choose Starting Weights
- [✓ Default Starts For Weights]
- User Defined Starts For Weights

Current Weight Settings
- Maximum Weight = 0.1
- Minimum Weight = 0
- Weight Sign = Both

Choose Starting Thresholds
- [✓ Default Starts For Thresholds]
- User Defined Starts For Thresholds

Current Threshold Settings
- Maximum Threshold = 0
- Minimum Threshold = 0
- Threshold Sign = Both

Training: AND.net

This code copyrighted by Michael R.W. Dawson 2002. For further information contact mdawson@ualberta.ca

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 three learning rules is to be used to train the perceptron. This choice also determines what activation function is being used in the perceptron's output units. The default rule is the Delta rule. When this rule is selected, the activation function is a threshold function. An output unit will generate a response of 1 when its net input is greater than a threshold, and a response of 0 otherwise. The second rule is a gradient descent rule for training output units with a continuous activation function, which in this case is the logistic equation. The derivative of the logistic equation is used to speed learning up by scaling the error term. The third rule is a gradient descent rule for training output units that employ a Gaussian activation function (i.e., value units). This rule is based on Dawson and Schopflocher's modification of gradient descent training, which uses an elaborated error term. It too scales error with the derivative of the activation function to speed learning up. A left-click of the mouse on this tool is all that is required to select which of the three learning rules will be used.

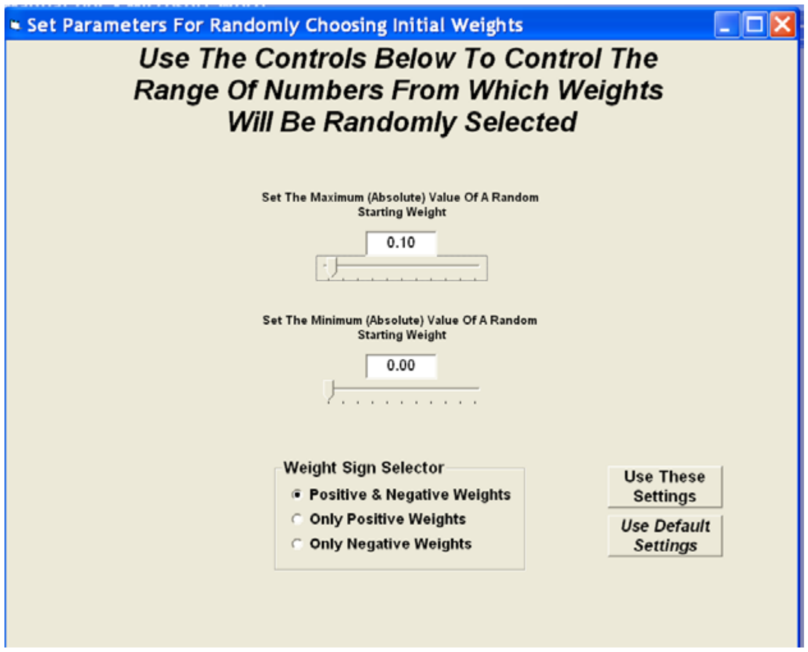
A second tool is used to choose a method for stopping training. In the first method, training stops after a maximum number of epochs (this value is set by the user). In the second method, training stops when there is a "hit" for every pattern and every output unit. This means that when each output is 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 the thresholds of the units (i.e., the threshold for the binary activation function, the bias for the logistic function, or the value of mu for the Gaussian function) 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.


Set Parameters For Randomly Choosing Initial Weights

Use The Controls Below To Control The Range Of Numbers From Which Weights Will Be Randomly Selected

Set The Maximum (Absolute) Value Of A Random Starting Weight: 0.10

Set The Minimum (Absolute) Value Of A Random Starting Weight: 0.00

Weight Sign Selector
- [✓ Positive & Negative Weights]
- Only Positive Weights
- Only Negative Weights

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.

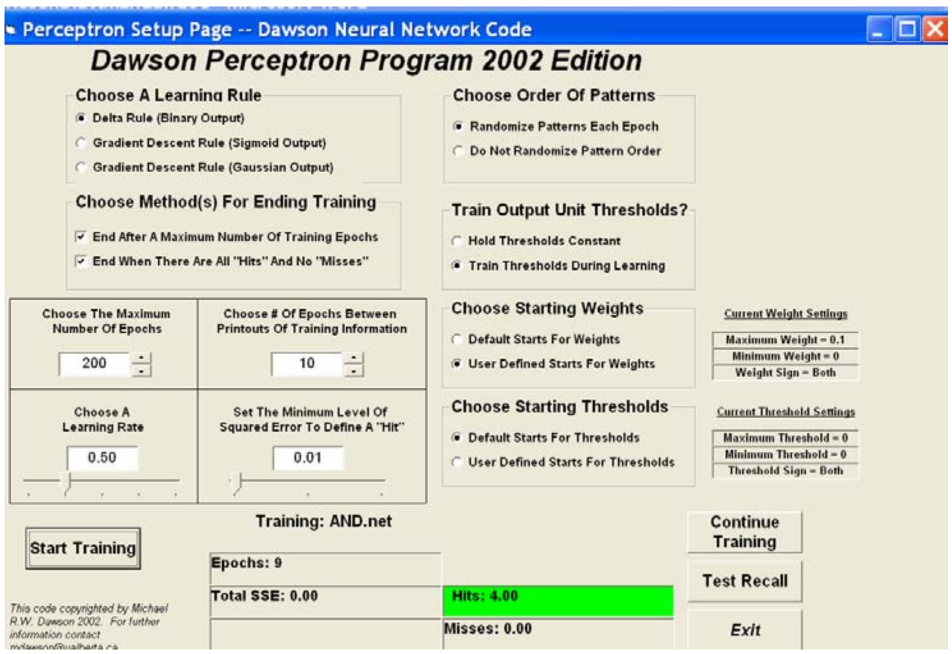
The four remaining tools on the form are used to set numerical values that control training.

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 200.

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 10. 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 10 of *Connectionism And Psychological Modeling*. The learning rate is used for all three learning rules. 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 default learning rate is 0.5. For some problems, when the Gaussian activation function is used, it may be desirable to speed learning up by *decreasing* this value to 0.1 or even to 0.01. For the other activation functions, the speed of learning can usually be increased by increasing the learning rate, provided that the learning rate is kept smaller than 1.0

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 perceptron's responses to stimuli will not be very accurate.



Perceptron Setup Page -- Dawson Neural Network Code

Dawson Perceptron Program 2002 Edition

Choose A Learning Rule
- [✓ Delta Rule (Binary Output)]
- Gradient Descent Rule (Sigmoid Output)
- Gradient Descent Rule (Gaussian Output)

Choose Method(s) For Ending Training
- [✓ End After A Maximum Number Of Training Epochs]
- [✓ End When There Are All "Hits" And No "Misses"]

Choose Order Of Patterns
- [✓ Randomize Patterns Each Epoch]
- Do Not Randomize Pattern Order

Train Output Unit Thresholds?
- Hold Thresholds Constant
- [✓ Train Thresholds During Learning]

Choose The Maximum Number Of Epochs: 200

Choose # Of Epochs Between Printouts Of Training Information: 10

Choose A Learning Rate: 0.50

Set The Minimum Level Of Squared Error To Define A "Hit": 0.01

Choose Starting Weights
- Default Starts For Weights
- [✓ User Defined Starts For Weights]

Current Weight Settings
- Maximum Weight = 0.1
- Minimum Weight = 0
- Weight Sign = Both

Choose Starting Thresholds
- [✓ Default Starts For Thresholds]
- User Defined Starts For Thresholds

Current Threshold Settings
- Maximum Threshold = 0
- Minimum Threshold = 0
- Threshold Sign = Both

Training: AND.net

Epochs: 9

Total SSE: 0.00

Hits: 4.00

Misses: 0.00

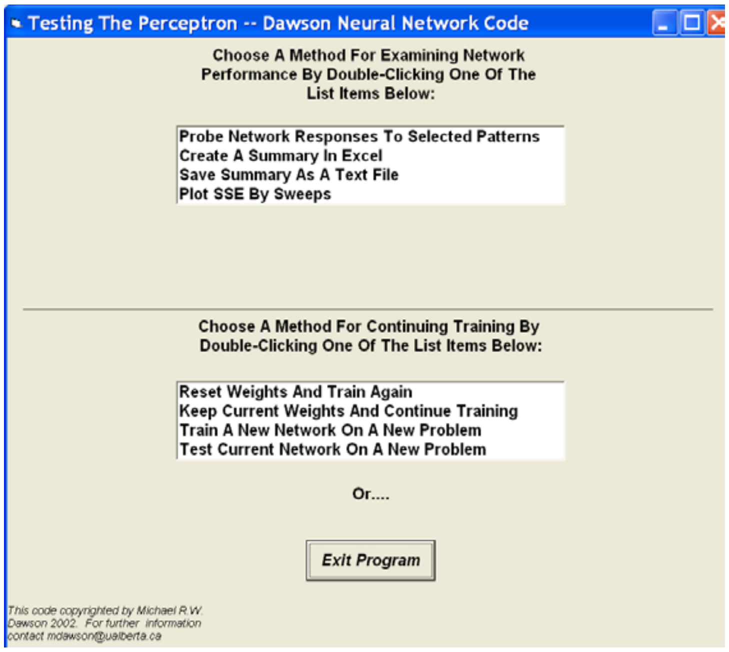
This code copyrighted by Michael R.W. Dawson 2002. For further information contact mdawson@ualberta.ca

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 9 epochs because SSE had dropped to zero, and there were 4 hits and 0 misses on the training patterns.

TESTING WHAT THE MEMORY 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.



Choose A Method For Examining Network Performance By Double-Clicking One Of The List Items Below:

Probe Network Responses To Selected Patterns
Create A Summary In Excel
Save Summary As A Text File
Plot SSE By Sweeps

Choose A Method For Continuing Training By Double-Clicking One Of The List Items Below:

Reset Weights And Train Again
Keep Current Weights And Continue Training
Train A New Network On A New Problem
Test Current Network On A New Problem

Or....

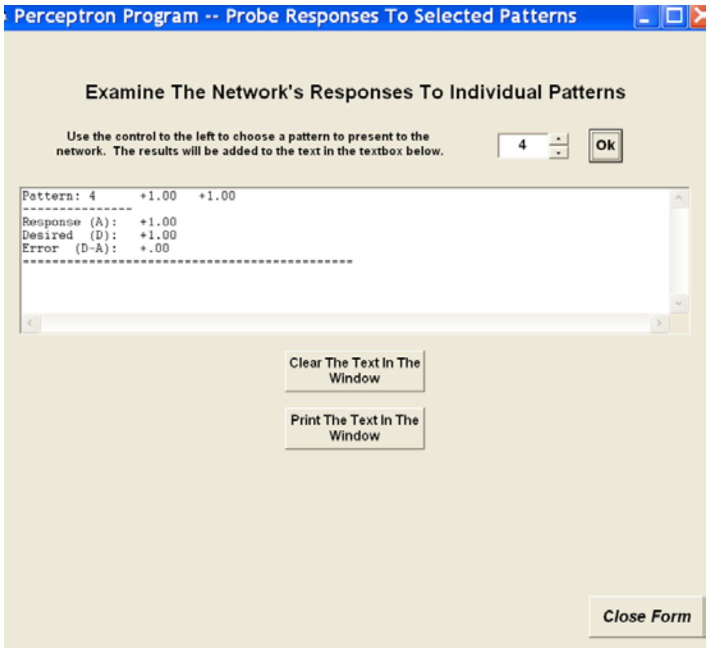
This code copyrighted by Michael R.W. Dawson 2002. For further information contact mdawson@ualberta.ca

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 4 has just been presented to the network.


Perceptron Program -- Probe Responses To Selected Patterns
Examine The Network's Responses To Individual Patterns
4
Ok
Clear The Text In The Window
Print The Text In The Window
Close Form

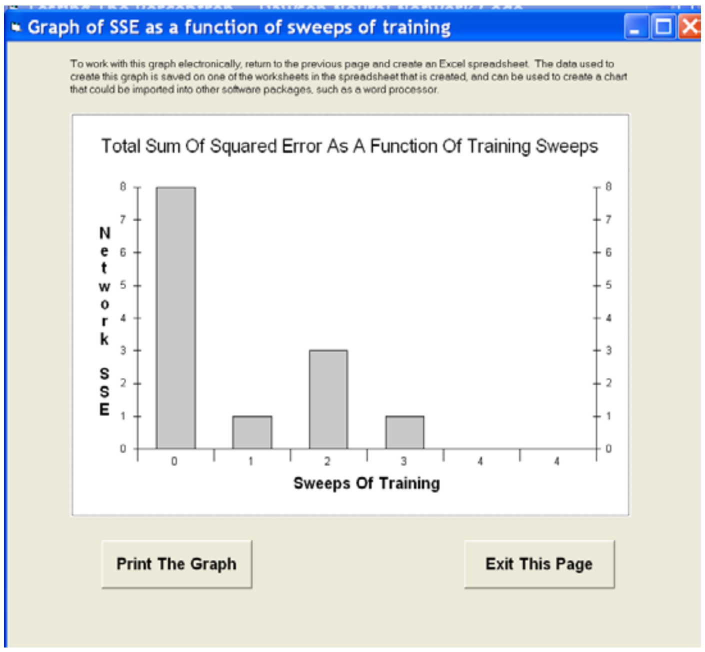
Use the control to the left to choose a pattern to present to the network. The results will be added to the text in the textbox below.

```
Pattern: 4      +1.00   +1.00
---------------
Response (A):   +1.00
Desired  (D):   +1.00
Error  (D-A):   +.00
---------------------------------------------
```

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.

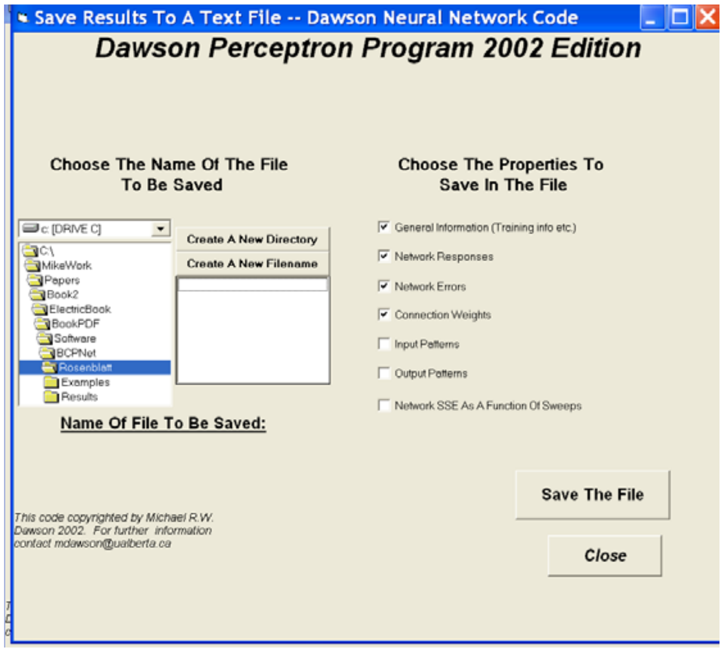

Graph of SSE as a function of sweeps of training
Total Sum Of Squared Error As A Function Of Training Sweeps
Network SSE
0
1
2
3
4
5
6
7
8
0
1
2
3
4
4
Sweeps Of Training
Print The Graph
Exit This Page

To work with this graph electronically, return to the previous page and create an Excel spreadsheet. The data used to create this graph is saved on one of the worksheets in the spreadsheet that is created, and can be used to create a chart that could be imported into other software packages, such as a word processor.

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.


Save Results To A Text File -- Dawson Neural Network Code
Dawson Perceptron Program 2002 Edition
Choose The Name Of The File To Be Saved
Choose The Properties To Save In The File
c: [DRIVE C]
C:\
MikeWork
Papers
Book2
ElectricBook
BookPDF
Software
BCPNet
Rosenblatt
Examples
Results
Create A New Directory
Create A New Filename
General Information (Training info etc.)
Network Responses
Network Errors
Connection Weights
Input Patterns
Output Patterns
Network SSE As A Function Of Sweeps
Name Of File To Be Saved:
Save The File
Close
This code copyrighted by Michael R.W. Dawson 2002. For further information contact mdawson@ualberta.ca

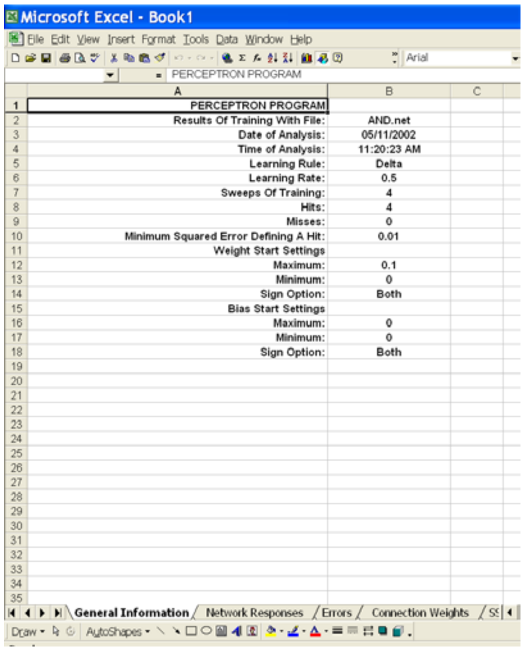
There are two sets of controls on this form. The first is a set of drive, directory, and file control boxes that are very similar to those found on the very first form seen when the program starts to run. One uses the drive and directory controls to navigate to a folder in which network data is to be 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 Rosenblatt program, and has been removed from RosenblattLite. 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.

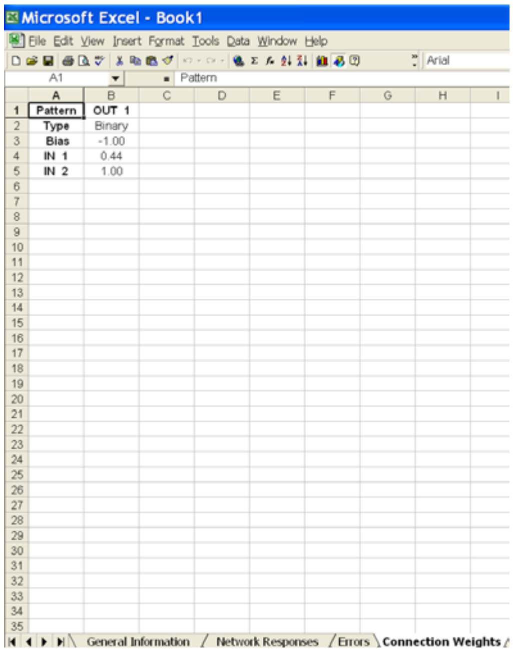
When this item is selected, a patience-requesting message is displayed on the “Test Network” form, and a number of different programming steps are taken to build an Excel Worksheet. When this is completed, the Worksheet is displayed as its own window, which will open on the user’s computer in front of any of the Rosenblatt 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.



	A	B
1	PERCEPTRON PROGRAM	
2	Results Of Training With File:	AND.net
3	Date of Analysis:	05/11/2002
4	Time of Analysis:	11:20:23 AM
5	Learning Rule:	Delta
6	Learning Rate:	0.5
7	Sweeps Of Training:	4
8	Hits:	4
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

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 Rosenblatt software. One can select different tabs in the worksheet to examine network properties. For example, in the figure on the bottom, 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.



	A	B
1	Pattern	OUT 1
2	Type	Binary
3	Bias	-1.00
4	IN 1	0.44
5	IN 2	1.00

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

If saving Excel files from Rosenblatt causes system crashes, it is likely because of memory resource conflicts. The Excel options were built into Rosenblatt 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 10 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 RosenblattLite 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 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. 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 Rosenblatt 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 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 AND.net via the Delta rule. The first step of training this network is to read in the file ortho8.net, which contains the following information:

```
1
0
2
4
0 0
0 1
1 0
1 1
0
0
0
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 (0 in this case). The third number provides the number of input units (2). The fourth number provides the number of training patterns (4). 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 Rosenblatt 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 4 different training patterns in this training set, there are four different rows in this section of the file. Each row provides the value that will be input, as a cue, to each of the 2 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 Rosenblatt 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 Rosenblatt program will only read in files that have the .net extension.
8. Use the Rosenblatt program to explore associative learning of the training set that you have created.

SOME EXERCISES FOR STUDYING THE PERCEPTRON

As was noted earlier, one of the primary purposes of the Rosenblatt and RosenblattLite programs is to provide students with a system that can be used to explore some of the properties of perceptrons. 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 Rosenblatt program is being used. However, all of the examples can be performed with RosenblattLite – 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 the first exercise is to explore the training of a perceptron on a small, simple problem. Furthermore, it is designed to permit a comparison amongst all three learning rules. Start the Rosenblatt program, and read in the AND.net file. Train the perceptron on this problem using the Delta rule. To start, the default settings are probably appropriate. How long does it take for the network to learn this problem? If you reset the network, does it always take the same amount of time to converge? How do changes in the learning rate affect the network's performance? Change the settings in order to generate a decent plot of network error as a function of epochs of training. What is the appearance of this graph? Use Excel to display network properties. Does the network converge to the same structure every time that it is trained? Repeat this investigation by using the gradient descent method of training, first using the logistic activation function, then using the Gaussian activation function. For each learning rule, make sure that you explore a variety of learning rates, etcetera. At the end of this exploration, you should be in a position to compare and contrast the three different learning rules.
2. The purpose of the second exercise is to explore some of the limitations of perceptrons, as well as one attempt to circumvent these limitations. Repeat Exercise 1, but use the file XOR.net. You should find that the perceptron has difficulty learning this problem when the first two learning rules

are used. Describe the difficulty that the perceptron is having -- is there a particular problem that it cannot solve, or easy generating areas for all of the patterns? Then train the perceptron on this problem using the third learning rule. You should find that learning in this case is possible. Why is it that this learning rule succeeds, while the other two failed? What does this imply for expanding perceptrons to solve problems that are not linearly separable?

3. The purpose of this exercise is to explore perceptron training with multiple output units. Use the file McCullfull.net. This file has four input patterns ([0,0], [1,0], [0,1], [1,1]) but has 16 output units. Each output unit responds is a particular kind of logic gate. Train the perceptron on this training set using each of the three rules. Can you get the perceptron to converge to a correct response to every pattern? If not, explain why. If one of the rules appears to work better than the others, then also provide an explanation of this. You might find your exploration of the perceptron in Exercise 2 useful in answering this question.
4. The purpose of the fourth exercise is to explore the utility of the perceptron in studying animal learning. Delameter, Sosa, and Koch (1999) were interested in studying positive and negative patterning in animals. In patterning, an animal learns to respond one way to indvidual stimuli, and the opposite way when combinations of stimuli are presented at the same time. In their study, they used 6 input units, for stimuli A, B, X, C, D, and Y. A and B were both of type X, and C and D were both of type Y. So whenever A or B was activated, so was X. Similarly, whenever C or D was activated, so was Y. In the first stage of their experiment, they trained a network to convergence on a pretraining regimen. The network was trained to turn on to AX and CY, and was trained to turn off to BX and DY. This is represented as (AX+, BX-, CY+, DY-). Then, without changing connection weights, a new problem was loaded in, as is indicated in the table below:

	File	Condition	File	Condition
+ve patterning	pretrain.net	AX+, BX-, CY+, DY-	cell1.net	AX-, CY-, AXCY+
+ve patterning	pretrain.net	AX+, BX-, CY+, DY-	cell2.net	BX-, DY-, BXDY+
-ve patterning	pretrain.net	AX+, BX-, CY+, DY-	cell3.net	AX+, CY+, AXCY-
-ve patterning	pretrain.net	AX+, BX-, CY+, DY-	cell4.net	BX+, DY+, BXDY-

One of the issues that they were interested in was the effect of pre-training on subsequent learning. In this exercise, you can replicate the Delameter, Sosa, and Koch experiment. Start the Rosenblatt program. Load in the file pretrain.net from the Examples\Delameter directory. Choose one of the learning rules, and train the network on this problem until it converges. Go to the "Test Recall" form. If you like, examine the properties of the trained network. Once you are done, tell the program to read in a new problem *without changing the current weights*. Read in one of the other files in the directory (cell1.net, cell2.net, cell3.net, or cell4.net). Train the network on this new file until it converges. Record the SSE at the start of training, the number of epochs to converge, and the SSE at the end of training. Repeat this sequence of events (train new network on pretraining.net, then train trained network on one of the cell?.net files) until all four of the post-training conditions have been examined. Did the network learn all of the problems that you presented? Were any of the problems harder to learn than the others? You could repeat this experiment with each of the learning rules. Does the network behave any differently depending on which learning rule has been selected? Why might this be the case? Finally, what if I told you that Delameter, Sosa, and Koch used a network that had four hidden units? Given this information, what are the implications of your results to their experiment?

APPENDIX 1: AND.TXT

The information provided below is a copy of the file AND.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.

```
Perceptron Training Program
=============================================
Results Of Training With File: AND.net
Date Of Analysis: 05/11/2002
Time Of Analysis: 11:20:15 AM
```

```
============================================
Learning rule: Delta
Learning rate: 0.5
Training completed after 4 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
=============================================
Network Responses To Each Input Pattern:
---------------
Pattern 1      +.00
Pattern 2      +.00
Pattern 3      +.00
Pattern 4      +1.00
=============================================
After training, sum of squared error was: 0
Network Response Errors To Each Input Pattern:
---------------
Pattern 1      +.00
Pattern 2      +.00
Pattern 3      +.00
Pattern 4      +.00
=============================================
Connection weights from input units (rows) to output units (columns):
---------------
       Out 1
OutType Binary
Bias    -1.00
INP 1   +.44
INP 2   +1.00
=============================================
The set of input patterns was:
---------------
Pattern 1      +.00    +.00
Pattern 2      +.00    +1.00
Pattern 3      +1.00   +.00
Pattern 4      +1.00   +1.00
=============================================
The set of desired outputs was:
---------------
Pattern 1      +.00
Pattern 2      +.00
Pattern 3      +.00
Pattern 4      +1.00
=============================================
Network SSE as a function of sweeps of training was:
---------------
Sweeps  Network SSE
0.00    8.00E+00
1.00    1.00E+00
2.00    3.00E+00
3.00    1.00E+00
4.00    0.00E+00
4.00    0.00E+00
=============================================
```