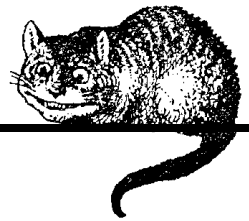
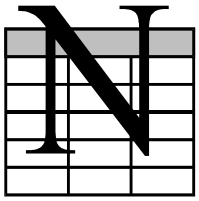
CHESHIRE
ENGINEERING
CORPORATION



Neuralyst™

Version 1.4



User's Guide

(October 1994)

Neural Network Technology

for Microsoft® Excel™

Neuralyst™ User's Guide
by Yin Shih

Editor: Shal Farley

Printing History:
October 1990: v1.1 Edition
March 1991: v1.2 Edition
January 1993: v1.3 Edition
October 1994: v1.4 Edition
March 1995: v1.4 Edition (errata)
March 2001: v1.4 Edition (PDF)
October 2001: v1.4 Edition (errata)

Trademark Notices

Disclaimer Notice

Neuralyst is no substitute for real thinking or common sense. The user should always review and check the results of Neuralyst's processing and evaluate it against known references and standards.

The use of Neuralyst for investment, speculation, gambling, or other similar or related purposes is at the user's risk. Results generated by Neuralyst are dependent on past information and there is no guarantee that future results can be forecast or predicted by Neuralyst. Trading in stocks, commodities and other securities or any form of speculation or gambling is inherently risky and may result in loss.

Warranty and Limitation of Liability

Cheshire Engineering Corporation (*Cheshire*) warrants the diskettes on which the software is distributed and the documentation to be free from defects in materials and workmanship for a period of ninety (90) days from the date of purchase. Cheshire will replace any defective diskette or documentation returned to Cheshire during the warranty period. Replacement is the exclusive remedy for any such defects and Cheshire shall have no liability for any other damage.

Cheshire disclaims all other warranties, expressed or implied, including but not limited to implied warranties of merchantability and fitness for any particular purpose.

In no event shall Cheshire be held liable for any damages whatsoever, including without limitation, damages resulting from financial loss, business interruption, loss of information or data, or any other incidental or consequential damages resulting from the use of Neuralyst, even if Cheshire has been advised of the possibility of such damages.

License Agreement

The computer program(s) described in this document (*Software*) is licensed by Cheshire Engineering Corporation (*Cheshire*) for use only under the terms of this license. Cheshire reserves any rights not expressly granted under this license.

You may make one copy, the *Working* copy, of the Software on any magnetic computer media and you may use the Working copy to execute the Software on one CPU at a time. There are no limits to the number of users, but there should be no possibility that two or more users could execute the Software simultaneously. You may make one additional copy, the *Archival* copy, of the Software for backup purposes only. The Archival copy may only be used to restore the Working copy in the event it is damaged or becomes unreadable.

The Software and accompanying documentation are copyrighted and remain the property of Cheshire. You may not rent, lease, resell for profit, distribute, network, reverse engineer, decompile, disassemble, modify, adapt, translate, or create derivative works based upon the Software, documentation, or any part thereof.

This agreement is governed by the laws of the State of California.

Cheshire Engineering Corporation
650 Sierra Madre Villa Avenue, Suite 201
Pasadena, CA 91107
U.S.A.
+1 626 351 5493 Info@CheshireEng.com

Contents

Chapter 1
Introduction

1.1 Introduction for New Users

Congratulations! You are about to experience a new dimension in spreadsheet processing. Neuralyst adds an unprecedented capability to Excel spreadsheets, providing for open-ended analysis of spreadsheet data and the recognition of associations between data you may have thought to be unrelated.

Up to now spreadsheets (and computers) have provided powerful analysis capabilities, but they were limited by people's ability to envision relationships and express them in spreadsheet models (and computer programs). If a relationship was not realized, then it did not appear on the spreadsheet and resulting tables or charts. Even if a relationship was suspected, it may have been difficult to prove or analyze the effects.

Neuralyst extends the capabilities of spreadsheets in this area using the technology of *neural networks*, also called *neural nets*. Neural networks are simulations of collections of model biological neurons. These are not simulations of real neurons in that they do not model the biology, chemistry, or physics of a real neuron. They do model several aspects of the information combining and pattern recognition behavior of real neurons in a simple yet meaningful way.

This neural modeling has shown incredible capability for emulation, analysis, prediction, and association. Neural networks can be used in a variety of powerful ways: to learn and reproduce rules or operations from given examples; to analyze and generalize from sample facts and

make predictions from these; or to memorize characteristics and features of given data and to match or make associations from new data to the old data. Neural networks can be used to make strict yes-no decisions or used to produce more critical, finely-valued judgments.

In this version of Neuralyst, neural network technology is combined with genetic optimization technology to allow you to develop the optimal neural networks to solve your modeling problems. Genetic optimization uses an evolution-like process to refine and enhance the structure of a neural network until it can model your problem in the most efficient way.

Cheshire has integrated neural network technology and genetic optimization technology with spreadsheet technology, combining a comfortable user interface and powerful data management capabilities with this new approach to data processing. All this is available to you now in Neuralyst.

1.2 Changes and Improvements in V1.4

There are many improvements and new features in Neuralyst V1.4. First, there are some slight changes in the format of the Neuralyst Working Area. However, all old V1.x worksheets will be recognized by V1.4 and they can be automatically updated to the format of the new Working Area without any loss of information.

V1.4 now requires Excel 4.0 at a minimum. It works with Excel 5.0, but that is not a requirement. V1.4 now supports both the U.S. and international versions of Excel with a single collection of macro sheets.

Several improvements have been made in the **Config** menu that enhance your ability to manage Neuralyst's interpretation of data. A new command, **Set Mode Flag Column**, has been added as a replacement to **Set TestFlag Column**. Check Section 8.1.6 for details.

As a companion to **Set Mode Flag Column**, the command **Set Mode Rows** has also been provided. This command provides three options to allow Neuralyst to set certain rows of the Mode Flag

column to control the minimum range of data, maximum range of data, and to define symbolic names for data. Check Section 8.1.7 for details.

The final addition to the **Config** menu is the **Edit Mode Lists** command. This command is also part of the new data management features and allows you to set or edit symbolic names for data and set minimum or maximum range limits. Check Section 8.1.10 for details.

There have been several enhancements in the **Neural** menu. **Set Enhanced Parameters** has been modified with some significant improvements. But, the real star of V1.4 is the addition of two commands to support genetic optimization technology.

In **Set Enhanced Parameters**, the Activation Function control panel has been enhanced, and two new control panels Calculation Method and Scaling Margin have been added. Two new functions are now available in the Activation Function control panel, Hyperbolic Tangent and Augmented Ratio of Squares. The Calculation Method control panel now allows you to select Fixed Point or Floating Point computation of neural networks. This allows you to trade off high speed against high precision. Finally, the Scaling Margin control panel allows you to adjust the headroom available in the data rescaling algorithms used by Neuralyst, thus allowing you more control of how your data is represented. Check Section 8.2.6 for details on all of the above.

Two new **Neural** menu commands are the keys to unlocking the new genetic optimization technology now integrated into Neuralyst, **Set Genetic Parameters** and **Run Genetic Supervisor**. These commands allow initialization and control of the Neuralyst Genetic Supervisor. See Chapter 7 for a better understanding of how genetic technology works and Sections 8.2.7 and 8.2.4 for details on the commands.

Finally, the connection weight limit has been increased from 10,920 to more than 131,000. This will allow neural networks more than ten times as complex to be developed.

Overall, we believe that we have provided a combination of enhancements that should improve the neural network training

process, allow for more sophisticated neural networks to be built, and improved ease of use. We hope you agree.

These changes and additions were the results of many comments received from current users. While not all suggestions were implemented in this version, we believe we have listened carefully and selected a worthwhile set for implementation. Please let us know how we did — and start sending in suggestions for the next version!

Chapter 2
Installing Neuralyst

2.1 About this Manual

This manual is used for both the Microsoft Windows and Apple Macintosh versions of Neuralyst (including Win32 and Power Macintosh). Generally, Neuralyst operations in either version are indistinguishable, except for file names, path versus folder names, and other features that are dependent on the behavior of the respective operating systems.

If you are experienced with Windows or the Macintosh, then the spirit of the directions for usage and operation given in this manual can be taken and applied to either system.

However, to minimize confusion, we have generally provided instructions or filenames for each version. These computer specific items are identified in one of two ways. If there is material of just a few words in length, then the Windows version is in plain text and the Macintosh version is enclosed immediately after in curly braces, i.e. {}. If there is material that is more extensive, then it is presented as alternative sections or subsections (Windows version first, then Macintosh version), as appropriate.

The screen snapshots shown in this manual are all taken from the Windows version of Neuralyst. If you have the Macintosh version of Neuralyst, the windows, control boxes, scroll bars, and so on will appear somewhat different, but the worksheet data should appear the same.

2.2 Neuralyst Distribution

The Neuralyst distribution you have received should contain the following items:

1. Neuralyst User's Guide
2. Neuralyst Distribution Disk
3. Software License Agreement
4. User Registration Form

If you do not have all the items listed then please contact Cheshire Customer Service immediately. See Appendix C in this guide for the recommended call-in procedure.

For your convenience, the Neuralyst distribution disk is not copy-protected. You may make an archival copy of the disk, install the software and use it as described in this guide and with the single-use restrictions specified in the Software License Agreement. Please observe these restrictions. Cheshire has made a considerable investment in the development of the Neuralyst program and violations of the copyright and software license are not trivial matters.

Each Neuralyst distribution disk has a label with a unique serial number printed on it. Protect this serial number and do not allow anyone to copy it or the software. The serial number is registered to you as the original purchaser or by your submission of the User Registration Form if you did not receive Neuralyst directly from Cheshire. This serial number will be required to validate customer support access and future software update privileges.

2.3 System Requirements

Neuralyst requires certain hardware and software configurations to run properly.

A math co-processor is not necessary. For most of its calculations Neuralyst defaults to using highly optimized fixed-point arithmetic

operations because in most computer systems they operate faster than the comparable operations using floating-point. If desired, the user can select floating-point operation for Neuralyst instead of fixed-point. A math co-processor will not significantly increase the performance of Neuralyst's computations using fixed-point, but it will increase the performance of Neuralyst's computations if floating-point is selected and it will increase the performance of Excel's computations.

2.3.1 Windows System Requirements

The Windows version of Neuralyst has been written to work on IBM personal computers (PC/XT, PC/AT, and PS/2) and compatibles. Neuralyst also requires Microsoft Windows Version 3.1 and Microsoft Excel Version 4.0 (or higher versions) in order to run. In general any system capable of running Windows and Excel will be able to support Neuralyst.

Neuralyst places a heavy computational load on a system. Therefore, higher performance systems, while not necessary, will reduce the processing (and corresponding waiting) time needed for Neuralyst to analyze a problem. In general, Neuralyst will perform best on 386, 486, and Pentium systems, while high-speed 80286 systems will probably work acceptably.

A math co-processor (80287, 80387 or 80487) is optional.

2.3.2 Macintosh System Requirements

The Macintosh version of Neuralyst has been written to work on Apple Macintosh computers. Neuralyst also requires Microsoft Excel Version 4.0 (or higher versions) in order to run. In general any Macintosh capable of running Excel will be able to support Neuralyst.

Neuralyst places a heavy computational load on a system. Therefore, higher performance systems, while not necessary, will reduce the processing (and corresponding waiting) time needed for Neuralyst to analyze a problem. In general, Neuralyst will perform best on 68030, 68040, or PowerPC systems. A 68020 will probably work acceptably. 68000 systems will work, but the performance will be marginal.

A math co-processor (68881 or 68882) is optional.

2.4 Installation Procedure

Neuralyst is integrated with Excel. This User's Guide assumes you are familiar with Excel and have all the materials and documentation necessary to run Excel. You should be familiar with the operations and commands available in Excel before proceeding to install and use Neuralyst. Cheshire Customer Service will not be able to answer any questions involving Excel set-up or operations.

If you have a PC follow the installation procedure described in Section 2.4.1. If you have a Macintosh follow the installation procedure described in Section 2.4.2.

2.4.1 Windows Installation Procedure

If you are reinstalling Neuralyst or installing a new version over an old version, then you need to make sure that you have exited Windows at least once since the last time you ran Neuralyst, otherwise Windows will try and run the old version of Neuralyst when you start Neuralyst after the install. Once this has been done, then you may proceed with the installation as described.

If you are installing Neuralyst for the first time, make sure that Windows and Excel are installed on your computer. If this is not so, then follow the respective installation procedures for each program as described by the vendor. After this has been done, you might need to make sure that Excel is registered in the Windows `WIN.INI` file so that Windows will be able to find Excel regardless of which directory is currently active. This will have been done automatically by the Excel installation, if it was done normally.

Turn on your PC and allow it to go through its normal boot procedure. If the boot procedure does not start Windows immediately, do so now — you must be in Windows to install Neuralyst correctly. Once your PC has concluded its boot sequence and Windows is loaded, insert the Neuralyst distribution disk in drive `A`.

From the Program Manager's **File** menu choose the **Run** command to run the install program. Enter the text:

```
A:INSTALL.EXE
```

and confirm **OK** to activate the install program. If your floppy is not drive A, then change the drive name to the correct name first, for example, change the text to B:INSTALL.EXE if you use drive B.

The installation program will now run. It will prompt you for the drive and directory that will be the target for the installation. This is defaulted to C:\NEURLYST. If your hard drive is not C, then edit the entry to show the proper drive name. It is recommended that \NEURLYST be used as the destination directory. Confirm **OK** when you have set the destination.

The installation program will now move the required Neuralyst program files and the example Neuralyst worksheet files to the C:\NEURLYST directory and register the Neuralyst program icon with Program Manager. The Neuralyst icon will appear in the Windows Applications Program Group. If there is no Windows Applications Program Group, then the install program will create a new Program Group called Neuralyst.

When the installation is complete, the installation program will exit and return you to Windows. If there are any errors in the installation process the program will exit, but an error report will be generated first and require your acknowledgment before the exit. In the case of error, you will have to correct the problem before trying the installation procedure again. See Appendix A for help with any problems you may encounter during installation.

See Appendix G for instructions on how to install the supplementary Trader's Macro Library. If you have purchased an optional library from Cheshire to integrate with Neuralyst, see the instructions included with each optional library for any special installation instructions.

Once the installation procedure is complete, you may use Neuralyst as described in the tutorial session shown in Chapter 4. But first, let's check to see if Neuralyst is working. Proceed to Section 2.5.

2.4.2 Macintosh Installation Procedure

If you are installing Neuralyst for the first time, make sure that Excel is installed on your computer. If this is not so, then follow the installation procedure for Excel as described by Microsoft. With this done, you are ready to proceed with the Neuralyst installation.

Turn on your Macintosh and allow it to go through its normal boot procedure. Once your Macintosh has concluded its boot sequence, insert the Neuralyst distribution disk in a floppy disk drive. Double-click on the disk icon labeled `Neuralyst Distribution` to show the disk contents, which will be a folder labeled Neuralyst. Select this folder and drag it to your hard disk to copy the contents of the folder. The `Neuralyst` folder may be copied to any area of your hard disk.

See Appendix G for instructions on how to install the supplementary Trader's Macro Library. If you have purchased an optional library from Cheshire to integrate with Neuralyst, see the instructions included with each optional library for any special installation instructions.

Once the installation procedure is complete, you may use Neuralyst as described in the tutorial session shown in Chapter 4. But first, let's check to see if Neuralyst is working. Proceed to the next section.

2.5 The First Neuralyst Session

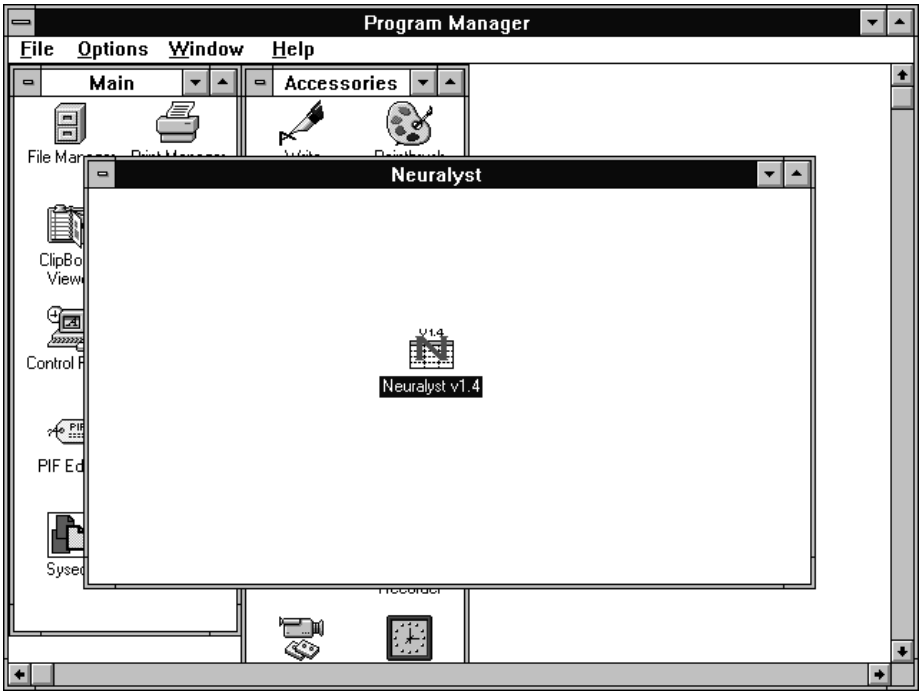


Figure 2-1 Neuralyst Icon *Windows version*

Let's check Neuralyst out. If you are running from a PC, start from the Windows Program Manager and find the Neuralyst icon. Double-click on the Neuralyst icon to start execution. {If you are running from a Macintosh, start from the Neuralyst folder. Double-click on the Excel macro file named Neuralyst to start execution — NOT the Neuralyst Lib icon.}

First Excel and then Neuralyst will be loaded. You will see the title screens for each appear in sequence. When both have appeared, Neuralyst is ready to run. The environment with which you will interact is primarily Excel's, but there will be new commands that relate to Neuralyst operations.

From the **File** menu choose the **Open** command. Find the Neuralyst directory and move to that, if you are not already in it. There will be

a number of example Neuralyst worksheets listed. Find the one named LOGIC.XLS {Logic} and load it by double-clicking on it.

Excel will now open a window that appears as shown in Figure 2-2.

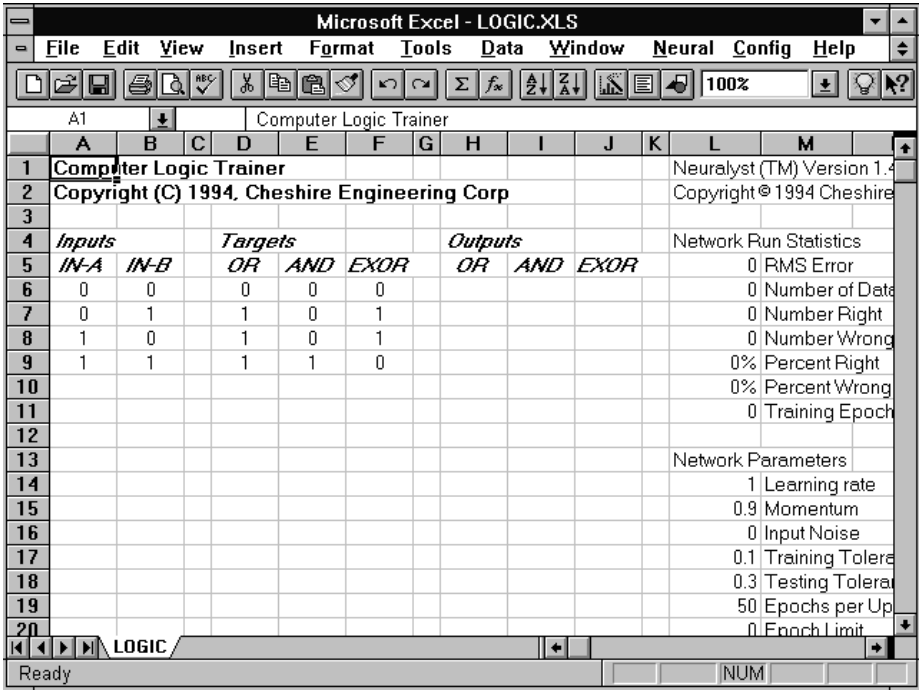


Figure 2-2 LOGIC.XLS {Logic} Example

LOGIC.XLS {Logic} defines three of the most basic operations for computer logic circuits. In computer design, conditions or events are represented by *binary* values, 0 or 1, indicating off/on, false/true, or absence/presence of these conditions or events. Most of the time computer circuits operate with two or more conditions or events as inputs and use these to form a new condition or event that will be used as an input in a succeeding operation.

For example, on row **7** of the example, In-A has the value 0 and In-B has the value 1. Under the Targets area are three columns, **D**, **E**, and **F**. These represent three different rules, OR, AND, and EXOR. OR means "either or both inputs must 1 for the output to be 1", AND means "both inputs must be 1 for the output to be 1" and EXOR means "either input, but not both, must be 1 for the output to be 1". In the case of row **7**,

the OR rule produces 1, the AND rule produces 0 and the EXOR rule produces 1. Each of the four rows, **6**, **7**, **8**, and **9** represent a different set of possible input combinations. With four combinations and three rules, there are a total of 12 possible input-output combinations.

The Inputs and Targets areas represent the information that is known and to which the neural network will be applied. The next area, designated Outputs, shows all 0's. These are the current outputs of the neural network, which have no correlation to the values in the Targets area prior to the application of the neural network.

You will notice two new menu items for Excel in addition to the example worksheet. These are the **Neural** and **Config** menus; which are the operating interface to Neuralyst. These will be discussed in much greater depth later.

*[If you have a Macintosh with a small screen, then Neuralyst will name its menus **N.** and **C.**, rather than **Neural** and **Config**, to save space. If you have a Macintosh, System 7, and a small screen, the normal Neuralyst interface will not work well; see Appendix A for instructions on how to manage this configuration.]*

For now, pull down the **Neural** menu and note the first line, **Reload Network**. Move the pointer to the line and release the mouse button to activate the command. Excel will show an hourglass {watch} cursor for a moment and status messages will flash across the bottom in the Status Display area. The neural network has now been loaded to the initial configuration saved on the worksheet.

[If you normally leave the Excel Calculation mode set to Manual, be sure to set it to Automatic when working with Neuralyst. Some initialization operations and the statistical information Neuralyst reports to you while executing will be incorrect if Automatic Calculation is not set.]

Pull down the **Neural** menu again. This time activate the **Train Network** command. This sets the neural network to work. As time progresses, the cell **L5**, labeled RMS Error, will be highlighted and you will see its value steadily decreasing. The lower this value, the better the neural network has learned the characteristics of the data presented to it. After some time (depending on the speed of your processor) Neuralyst will stop operations and the Outputs area will

be updated. The values now shown represent the outputs of the neural network and will match the corresponding values in the Targets area.

Neuralyst is working! It has learned all three rules for these input combinations and is now able to duplicate the rules presented to it.

Let's take a moment to understand neural networks better and what this example means before we try out more examples. For now, use **Exit** {**Quit**} from the **File** menu to exit Excel, but don't save the changes when asked.

Chapter 3
Basic Concepts for Neural Networks

3.1 Real Neurons

Let's start by taking a look at a biological neuron. Figure 3-1 shows such a neuron.

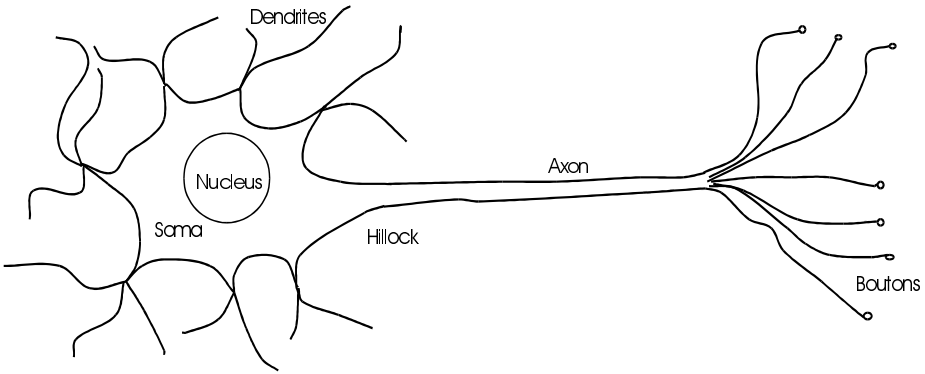


Figure 3-1 A Biological Neuron

A neuron operates by receiving signals from other neurons through connections, called *synapses*. The combination of these signals, in excess of a certain *threshold* or *activation* level, will result in the neuron *firing*, that is sending a signal on to other neurons connected to it. Some signals act as *excitations* and others as *inhibitions* to a neuron firing. *What we call thinking is believed to be the collective effect of the presence or absence of firings in the pattern of synaptic connections between neurons.*

This sounds very simplistic until we recognize that there are approximately one hundred billion (100,000,000,000) neurons each connected to as many as one thousand (1,000) others in the human brain. The massive number of neurons and the complexity of their interconnections results in a “thinking machine”, your brain.

Each neuron has a body, called the *soma*. The soma is much like the body of any other cell. It contains the cell nucleus, various bio-chemical factories and other components that support ongoing activity.

Surrounding the soma are *dendrites*. The dendrites are receptors for signals generated by other neurons. These signals may be excitatory or inhibitory. All signals present at the dendrites of a neuron are combined and the result will determine whether or not that neuron will fire.

If a neuron fires, an electrical impulse is generated. This impulse starts at the base, called the *hillock*, of a long cellular extension, called the *axon*, and proceeds down the axon to its ends.

The end of the axon is actually split into multiple ends, called the *boutons*. The boutons are connected to the dendrites of other neurons and the resulting interconnections are the previously discussed synapses. (Actually, the boutons do not touch the dendrites; there is a small gap between them.) If a neuron has fired, the electrical impulse that has been generated stimulates the boutons and results in electrochemical activity which transmits the signal across the synapses to the receiving dendrites.

At rest, the neuron maintains an electrical potential of about 40-60 millivolts. When a neuron fires, an electrical impulse is created which is the result of a change in potential to about 90-100 millivolts. This impulse travels between 0.5 to 100 meters per second and lasts for about 1 millisecond. Once a neuron fires, it must rest for several milliseconds before it can fire again. In some circumstances, the repetition rate may be as fast as 100 times per second, equivalent to 10 milliseconds per firing.

Compare this to a very fast electronic computer whose signals travel at about 200,000,000 meters per second (speed of light in a wire is 2/3 of that in free air), whose impulses last for 10 nanoseconds and may

repeat such an impulse immediately in each succeeding 10 nanoseconds continuously. Electronic computers have at least a 2,000,000 times advantage in signal transmission speed and 1,000,000 times advantage in signal repetition rate.

It is clear that if signal speed or rate were the sole criteria for processing performance, electronic computers would win hands down. What the human brain lacks in these, it makes up in numbers of elements and interconnection complexity between those elements. This difference in structure manifests itself in at least one important way; the human brain is not as quick as an electronic computer at arithmetic, but it is many times faster and hugely more capable at recognition of patterns and perception of relationships.

The human brain differs in another, extremely important, respect beyond speed; it is capable of "self-programming" or adaptation in response to changing external stimuli. In other words, it can learn. The brain has developed ways for neurons to change their response to new stimulus patterns so that similar events may affect future responses. In particular, the sensitivity to new patterns seems more extensive in proportion to their importance to survival or if they are reinforced by repetition.

3.2 Neural Network Structure

Neural networks are models of biological neural structures. The starting point for most neural networks is a model neuron, as in Figure 3-2. This neuron consists of multiple inputs and a single output. Each input is modified by a *weight*, which multiplies with the input value. The neuron will combine these weighted inputs and, with reference to a threshold value and activation function, use these to determine its output. This behavior follows closely our understanding of how real neurons work.

While there is a fair understanding of how an individual neuron works, there is still a great deal of research and mostly conjecture regarding the way neurons organize themselves and the mechanisms used by arrays of neurons to adapt their behavior to external stimuli.

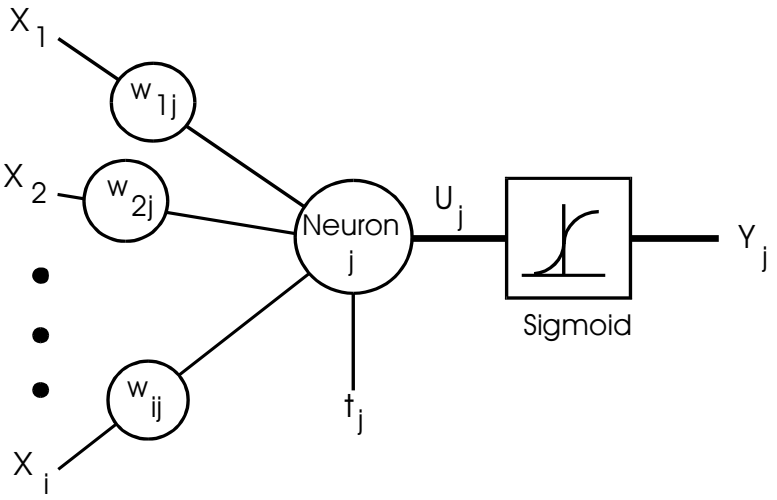


Figure 3-2 A Model Neuron

There are a large number of experimental neural network structures currently in use reflecting this state of continuing research.

In our case, we will only describe the structure, mathematics and behavior of that structure known as the *backpropagation network*. This is the most prevalent and generalized neural network currently in use. If the reader is interested in finding out more about neural networks or other networks, please refer to the material listed in Appendix F.

To build a backpropagation network, proceed in the following fashion. First, take a number of neurons and array them to form a *layer*. A layer has all its inputs connected to either a preceding layer or the inputs from the external world, but not both within the same layer. A layer has all its outputs connected to either a succeeding layer or the outputs to the external world, but not both within the same layer.

Next, multiple layers are then arrayed one succeeding the other so that there is an input layer, multiple intermediate layers and finally an output layer, as in Figure 3-3. Intermediate layers, that is those that have no inputs or outputs to the external world, are called *hidden layers*. Backpropagation neural networks are usually

fully connected. This means that each neuron is connected to every output from the preceding layer or one input from the external world

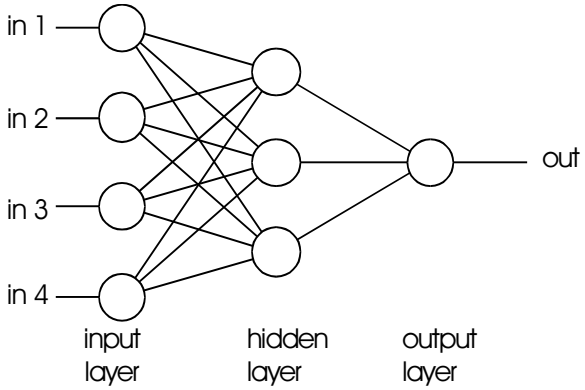


Figure 3-3 Backpropagation Network

if the neuron is in the first layer and, correspondingly, each neuron has its output connected to every neuron in the succeeding layer.

Generally, the input layer is considered a distributor of the signals from the external world. Hidden layers are considered to be categorizers or feature detectors of such signals. The output layer is considered a collector of the features detected and producer of the response. While this view of the neural network may be helpful in conceptualizing the functions of the layers, you should not take this model too literally as the functions described may not be so specific or localized.

With this picture of how a neural network is constructed, we can now proceed to describe the operation of the network in a meaningful fashion.

3.3 Neural Network Operation

The output of each neuron is a function of its inputs. In particular, the output of the ***j***th neuron in any layer is described by two sets of equations:

$$U_j = \sum (X_i * w_{ij}) \qquad \text{[Eqn 3-1]}$$

and

$$Y_j = F_{th} (U_j + t_j) \qquad \text{[Eqn 3-2]}$$

For every neuron, ***j***, in a layer, each of the ***i*** inputs, $\boldsymbol{X}_i$, to that layer is multiplied by a previously established weight, $\boldsymbol{w}_{ij}$. These are all summed together, resulting in the internal value of this operation, $\boldsymbol{U}_j$. This value is then biased by a previously established threshold value, $\boldsymbol{t}_j$, and sent through an activation function, $\boldsymbol{F}_{th}$. This activation function is usually the sigmoid function, which has an input to output mapping as shown in Figure 3-4. The resulting output, $\boldsymbol{Y}_j$, is an input to the next layer or it is a response of the neural network if it is the last layer. Neuralyst allows other threshold functions to be used in place of the sigmoid described here. See Section 6.6 for details.

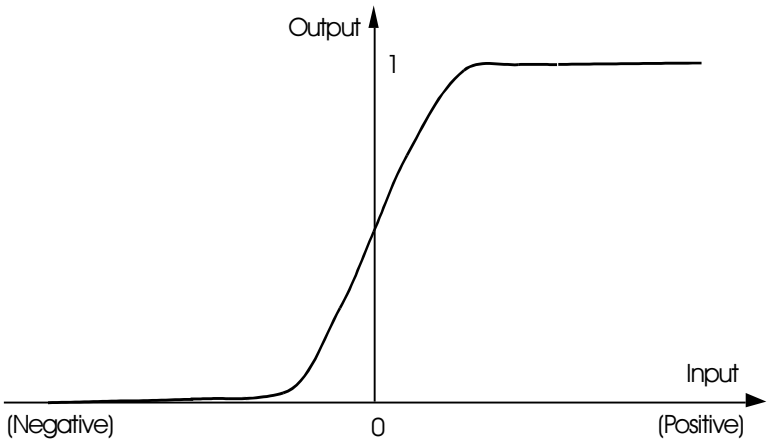


Figure 3-4 Sigmoid Function

In essence, Equation 3-1 implements the combination operation of the neuron and Equation 3-2 implements the firing of the neuron.

From these equations, a predetermined set of weights, a predetermined set of threshold values and a description of the network structure (that is the number of layers and the number of neurons in each layer), it is possible to compute the response of the neural network to any set of inputs. And this is just how Neuralyst goes about producing the response. But how does it learn?

3.4 Neural Network Learning

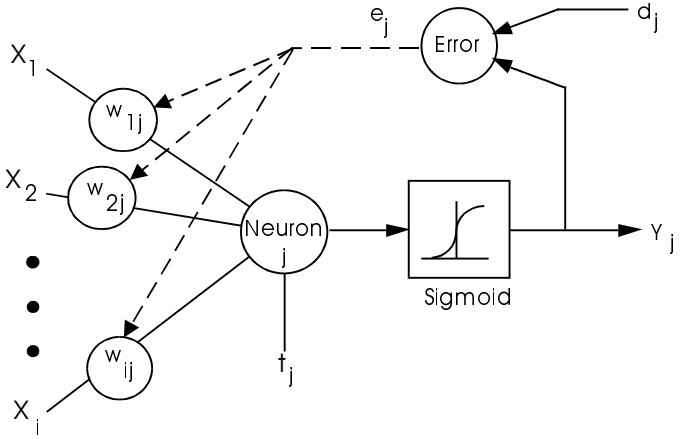


Figure 3-5 Neuron Weight Adjustment

Learning in a neural network is called *training*. Like training in athletics, training in a neural network requires a coach, someone that describes to the neural network what it should have produced as a response. From the difference between the desired response and the actual response, the *error* is determined and a portion of it is propagated backward through the network. At each neuron in the network the error is used to adjust the weights and threshold values of the neuron, so that the next time, the error in the network response will be less for the same inputs.

This corrective procedure is called *backpropagation* (hence the name of the neural network) and it is applied continuously and repetitively

for each set of inputs and corresponding set of outputs produced in response to the inputs. This procedure continues so long as the individual or total errors in the responses exceed a specified level or until there are no measurable errors. At this point, the neural network has learned the training material and you can stop the training process and use the neural network to produce responses to new input data.

[There is some heavier going in the next few paragraphs. Skip ahead if you don't need to understand all the details of neural network learning.]

Backpropagation starts at the output layer with the following equations:

$$w_{ij} = w'_{ij} + LR * e_j * X_i \qquad \text{[Eqn 3-3]}$$

and

$$e_j = Y_j * (1 - Y_j) * (d_j - Y_j) \qquad \text{[Eqn 3-4]}$$

For the ***i***th input of the ***j***th neuron in the output layer, the weight $\boldsymbol{w}_{ij}$ is adjusted by adding to the previous weight value, $\boldsymbol{w'}_{ij}$, a term determined by the product of a *learning rate*, ***LR***, an error term, $\boldsymbol{e}_j$, and the value of the ***i***th input, $\boldsymbol{X}_i$. The error term, $\boldsymbol{e}_j$, for the jth neuron is determined by the product of the actual output, $\boldsymbol{Y}_j$, its complement, $\boldsymbol{1 - Y}_j$, and the difference between the desired output, $\boldsymbol{d}_j$, and the actual output.

Once the error terms are computed and weights are adjusted for the output layer, the values are recorded and the next layer back is adjusted. The same weight adjustment process, determined by Equation 3-3, is followed, but the error term is generated by a slightly modified version of Equation 3-4. This modification is:

$$e_j = Y_j * (1 - Y_j) * \sum (e_k * w'_{jk}) \qquad \text{[Eqn 3-5]}$$

In this version, the difference between the desired output and the actual output is replaced by the sum of the error terms for each neuron, ***k***, in the layer immediately succeeding the layer being processed (remember, we are going backwards through the layers so

these terms have already been computed) times the respective pre-adjustment weights.

The learning rate, ***LR***, applies a greater or lesser portion of the respective adjustment to the old weight. If the factor is set to a large value, then the neural network may learn more quickly, but if there is a large variability in the input set then the network may not learn very well or at all. In real terms, setting the learning rate to a large value is analogous to giving a child a spanking, but that is inappropriate and counter-productive to learning if the offense is so simple as forgetting to tie their shoelaces. Usually, it is better to set the factor to a small value and edge it upward if the learning rate seems slow.

In many cases, it is useful to use a revised weight adjustment process. This is described by the equation:

$$w_{ij} = w'_{ij} + (1-M)*LR*e_j*X_i + M*(w'_{ij} - w''_{ij}) \qquad \text{[Eqn 3-6]}$$

This is similar to Equation 3-3, with a *momentum* factor, ***M***, the previous weight, $\boldsymbol{w'_{ij}}$, and the next to previous weight, $\boldsymbol{w''_{ij}}$, included in the last term. This extra term allows for momentum in weight adjustment. Momentum basically allows a change to the weights to persist for a number of adjustment cycles. The magnitude of the persistence is controlled by the momentum factor. If the momentum factor is set to 0, then the equation reduces to that of Equation 3-3. If the momentum factor is increased from 0, then increasingly greater persistence of previous adjustments is allowed in modifying the current adjustment. This can improve the learning rate in some situations, by helping to smooth out unusual conditions in the training set.

[Okay, that's the end of the equations. You can relax again.]

As you train the network, the total error, that is the sum of the errors over all the training sets, will become smaller and smaller. Once the network reduces the total error to the limit set, training may stop. You may then apply the network, using the weights and thresholds as trained.

It is a good idea to set aside some subset of all the inputs available and reserve them for *testing* the trained network. By comparing the

output of a trained network on these test sets to the outputs you know to be correct, you can gain greater confidence in the validity of the training. If you are satisfied at this point, then the neural network is ready for *running*.

Usually, no backpropagation takes place in this running mode as was done in the training mode. This is because there is often no way to be immediately certain of the desired response. If there were, there would be no need for the processing capabilities of the neural network! Instead, as the validity of the neural network outputs or predictions are verified or contradicted over time, you will either be satisfied with the existing performance or determine a need for new training. In this case, the additional input sets collected since the last training session may be used to extend and improve the training data.

Now that we have an understanding of how a neural network functions and what it may accomplish, let's get into a more detailed Neuralyst session.

Chapter 4
A Neuralyst Tutorial

4.1 Starting Neuralyst

To start Neuralyst, the PC should first be running Windows. Once you have Windows active, Neuralyst may be started in one of two ways. First, you may start by double-clicking on the Neuralyst icon in Windows Program Manager. This will cause Excel to run followed immediately by the loading of the Neuralyst package. Second, you may start from within Excel: from the **File** menu choose the **Open** command, move to the Neuralyst directory and select NEURLYST.XLM to cause the Neuralyst package to load.

[For Windows, only one instance of Excel with Neuralyst should be running at one time. If you start Neuralyst twice, the two instances may interfere with each other. See Appendix A for more information.]

{Neuralyst may be started in one of two ways on the Macintosh. First, you may start by double-clicking on the Excel macro file named Neuralyst. This will cause Excel to run followed immediately by the loading of the Neuralyst package. Second, you may start from within Excel: from the **File** menu choose the **Open** command, move to the Neuralyst folder, and select Neuralyst to cause the Neuralyst package to load.}

Once Excel is running with the Neuralyst package loaded, from the **File** menu you may choose the **New** command to create a new file or the **Open** command to select the file to use.

Let's go ahead and open the file EXPLODE.XLS {Explode} in the Neuralyst directory. You will see a window similar to that shown in Figure 4-1.

Note the two new menu items that have been added with the loading of the Neuralyst package. These two new menus are **Neural** and **Config**. These menus interface to Neuralyst and allow you to define the problem and control the operations of Neuralyst.

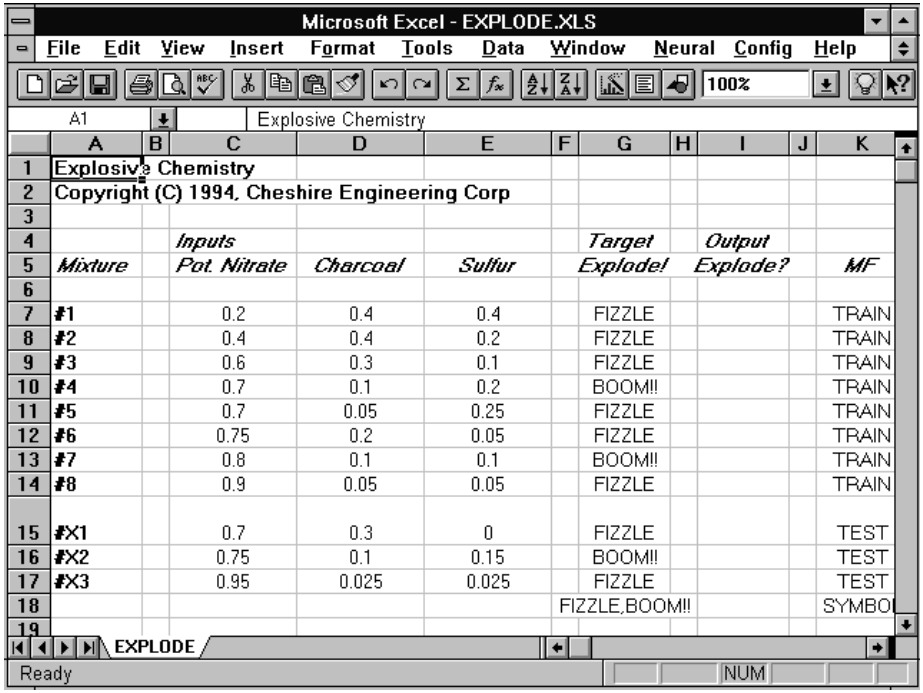


Figure 4-1 EXPLODE.XLS {Explode}

Now, let's take a look at the worksheet. EXPLODE.XLS {Explode} is a representation of a series of chemistry experiments. A young chemist has compounded mixtures #1-8 using the proportions of Potassium Nitrate, Charcoal, and Sulfur shown. For each of these mixtures he has determined whether or not that mixture will explode. This is summarized in the first eight rows of the Explode! column. For example, row **9** shows that a mixture of 60% Potassium Nitrate, 30% Charcoal, and 10% Sulfur did not explode (the value in the column Explode! is FIZZLE), while row **10** shows that a mixture of 70% Potassium Nitrate, 10% Charcoal, and 20% Sulfur did explode (the value in the Explode! column is BOOM!!).

[The chemistry rules in the world of EXPLODE.XLS {Explode} do not correspond to the real world. If you really want to learn how to make gunpowder, you will have to find the formula elsewhere!]

However, his lab manager has gotten a little tired of paying for all the broken glassware and has asked him to cut the experiments short, before he has had a chance to test compounds #X1-X3. Does he have enough information from the previous experiments to make an analysis of whether the proportions of the three chemicals in these three mixtures will result in an explosion?

Neuralyst can be used to help this young chemist analyze his results.

4.2 Configuring the Neural Network

Neuralyst cannot work with a completely unorganized conglomeration of facts. For a problem to be presented to Neuralyst, it should be organized as *instances*, *examples* or *cases*, consisting of related data or facts including known or expected results and goals.

Within this context, Neuralyst expects that a certain number of rows, of all the rows of the worksheet, will represent each instance of a problem. If that number is 1, then each row constitutes a separate instance. If that number is greater than 1, then that many rows are taken together to constitute each separate instance. In the case of EXPLODE.XLS {Explode}, the data has been organized into individual rows. Each of rows **7** through **17** is used to describe a different mixture in this series of experiments. The first eight, rows **7** through **14**, represent instances whose results are known; the next three, rows **15** through **17**, represent instances that we wish to predict after neural network training on the first eight. The last row with data, row **18**, is a special row which is used to define the valid symbols, <FIZZLE, BOOM!!>, for the Explode! column.

Neuralyst also expects that the columns of a worksheet individually represent different facts, goals, or predictions for each instance of a problem. In this worksheet the first column, column **A**, is descriptive and identifies each instance. The next three columns, **C**, **D**, and **E** (blank columns are OK), represent facts that describe this particular

instance. The next column, **G**, also represents a fact, whether or not the mixture exploded, but in this case it also represents the known result. The next column, **I**, will be used by Neuralyst to present its outputs or predictions. The final column, **K**, is used primarily to distinguish between those instances with known results to be used in training the neural network and those instances which have been saved as tests or those instances without known results for which a prediction is desired. It also identifies those rows which are used for special purposes.

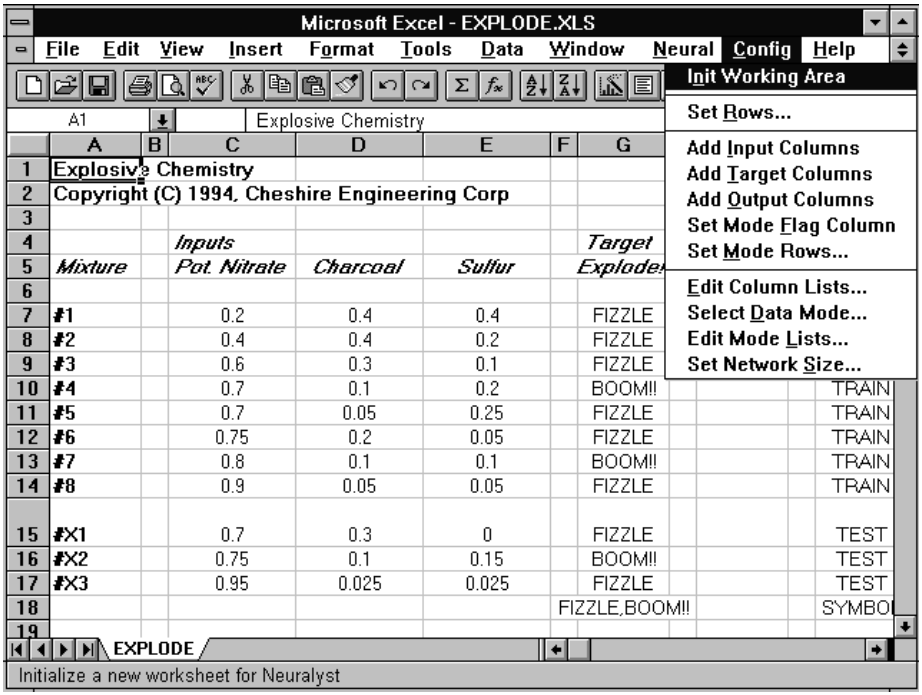


Figure 4-2 The Neuralyst Config Menu

The **Config** menu contains the interface to the commands that allow you to define a problem. Figure 4-2 shows the commands available from the menu.

The **Config** menu is organized in the same sequence as a problem should generally be defined for Neuralyst. The first item is **Init Working Area**. This will establish the area on the worksheet that is reserved for Neuralyst's parameters and data. This area should

be the first cell that is to the right or below all the other data in the worksheet. No other data should be entered in or beyond the defined area as it may result in the incorrect operation of Neuralyst or it may be deleted as a result of one of Neuralyst's internal actions. In this example, move your cursor to cell **M1**, select it, pull the **Config** menu down and select the **Init Working Area** command. Neuralyst will ask you to confirm the initialization request; go ahead and click **OK**. The cursor will now show as an hourglass {watch} for a moment as Neuralyst initializes the Working Area. The results should appear as in Figure 4-3.

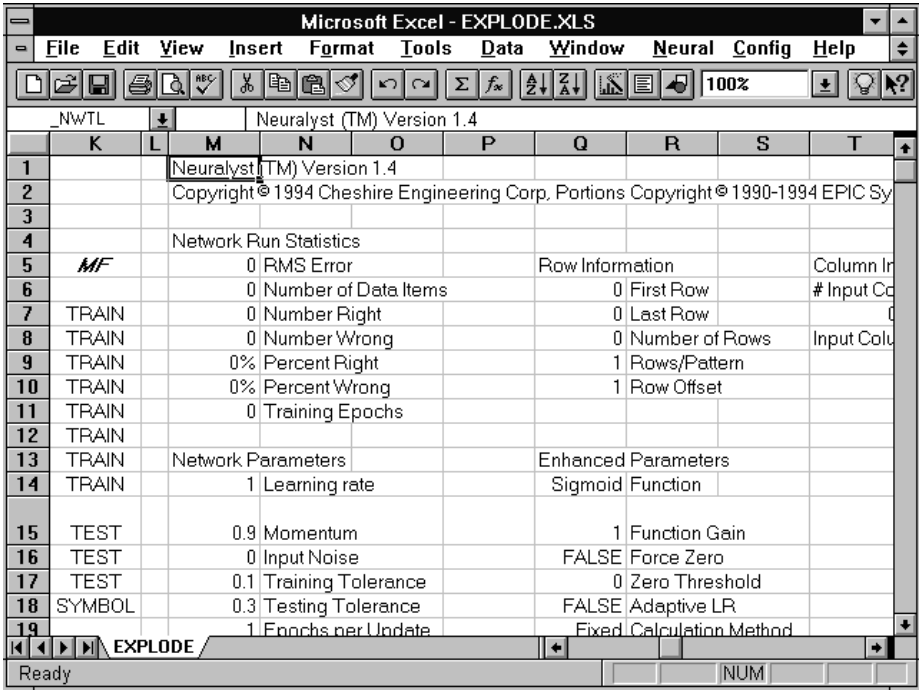


Figure 4-3 Portion of Neuralyst Working Area

Now the range of the input instances needs to be defined. This is the purpose of the **Set Rows** command. Scroll the window to show column **A** and select the cells **A7** through **A18**. Now pull down the **Config** menu and select the **Set Rows** command. This will tell Neuralyst which rows it is to use as inputs to the neural network. The column used for this purpose is not critical: it may or may not be one of the Input columns to be defined next. A dialog box will appear after

the selection has been accepted to let you set the number of rows per pattern and the number of rows to offset between patterns. Leave both at their default values of 1 for this example and confirm with OK.

The Input columns containing the facts for each instance are defined through the **Add Input Columns** command. The columns to be set as Input columns may either be selected one at a time or several columns may be selected as an extended selection. To do this, point to the column headings and select the three columns, **C** through **E**. Now pull down the **Config** menu again and select the **Add Input Columns** command. This will tell Neuralyst that these columns contain the input facts. Each row of these columns, as previously defined by **Set Rows**, being another instance to be presented to the neural network.

The known result or consequence of these facts is defined for the neural network through the **Add Target Columns** command. Select column **G** and then select this command on the **Config** menu. Similarly, the neural network's output or prediction for these facts after processing is defined through the **Add Output Columns** command. Select column **I** and then select this command on the **Config** menu. Multiple columns may be set as Target columns or Output columns, but this feature is not necessary for this example.

The final column to be defined is the Mode Flag column. When presenting a problem to Neuralyst, there needs to be some way to distinguish between those facts which are to be used for training purposes and either those facts which have known results but which are reserved to test the success of the training or those facts for which no results are known and for which the neural network prediction is desired. The Mode Flag column makes that distinction. Rows that have this column set to TRAIN will be treated as training sets. Rows that have this column set to TEST will be treated as "testing" sets. In the cases where the Target column values are defined (results are known), then these may be considered test rows. In the cases where the Target column values are not defined (results are not known), then these may be considered as facts requiring neural network prediction. To define the Mode Flag column, select column **K** and select the command **Set Mode Flag Column** from the **Config** menu.

The Mode Flag column is also used to indicate a number of special rows that are useful to modify the behavior of Neuralyst. There are three additional rows or row types that can be designated, these are: SYMBOL, MIN, and MAX. A row designated by SYMBOL is interpreted by Neuralyst to contain definitions of symbols for Input or Target columns. A row designated by MIN is interpreted by Neuralyst to contain definitions of the minimum range limit for Input, Target, or Output, columns. A row designated by MAX is interpreted by Neuralyst to contain definitions of the maximum range limit for Input or Target columns. If MIN or MAX rows are set, the limits entered for each target column are applied to the corresponding Output column. To enter row **18** as a SYMBOL row, select row **18** and select the command **Set Mode Rows** from the **Config** menu. When the dialog box appears, select the **Set Symbol Row** option and confirm **OK**.

The **Edit Column Lists** command allows you to view the column labels entered under each column type and to change them if desired. Select **Edit Column Lists** from the **Config** menu to see what this looks like. You can try making some changes, but be sure to restore the labels to their original values and confirm **OK** when you are done.

The **Select Data Mode** command allows you to classify the data for your problem between a training set and a testing set. This is done through a number of options which will set the Mode Flag column as TRAIN or TEST for you to indicate training or testing.

The **Edit Mode Lists** command allows you to view the settings of SYMBOL, MIN, and MAX fields for each designated column and to change them if desired. Select column **G** and then select **Edit Mode Lists** from the **Config** menu to see what this looks like. You can try making some changes, but be sure to restore the labels to their original values and confirm **OK** when you are done.

Once the problem data has been defined, the neural network structure needs to be defined. To do this, select the **Set Network Size** command from the **Config** menu. A dialog box will appear requesting you to input the number of layers for the neural network. The default is 2, but in this instance you should enter 3 and then confirm **OK**. Another dialog box will now appear. This dialog box shows the input and output layers with a fixed number of neurons determined by the number of Input Columns and Target columns specified. In this case

the input layer is layer 1 which has three neurons and the output layer is layer 3 which has one neuron. There is an additional layer, layer 2, which starts with a default of one neuron. Change this to 3 and confirm **OK**. The hourglass {watch} cursor will appear for a moment as Neuralyst builds the neural network, then Neuralyst will display the Network Weights, as they have been initialized in the Working Area.

This completes the configuration process, but before you proceed to running Neuralyst, you should take a moment to step back and review the worksheet as it appears now. This will give you a general sense of how a Neuralyst worksheet is organized. First there is a Descriptive Area, which is usually located above or to the left of the Problem Definition Area. This includes column titles, information about the worksheet and other descriptive text. Then there is the Problem Definition Area, within which there will be a set of Input columns which represent facts to be presented, a set of Target or goal columns which represent known results, a set of Output columns for the neural network to present its outputs or predictions and a Mode Flag column which allows you to specify training instances, testing or prediction instances, and special definition rows used by Neuralyst. Finally, there is a Neuralyst Working Area which Neuralyst uses for its operational purposes. The Working Area will usually be the rightmost or bottommost part of the worksheet.

4.3 Running the Neural Network

The Neural menu provides the commands that allows you to control the operation of the neural network.

The first command in the menu, **Reload Network**, allows a saved Neuralyst worksheet to be reloaded. The next three, **Train Network**, **Run/Predict with Network**, and **Run Genetic Supervisor**, determine the operating mode of the neural network. The next, **Set Network Parameters**, allows you to set parameters which control the operation of the neural network. **Set Enhanced Parameters** allows you to enhance the behavior of the neural network from the standard backpropagation neural

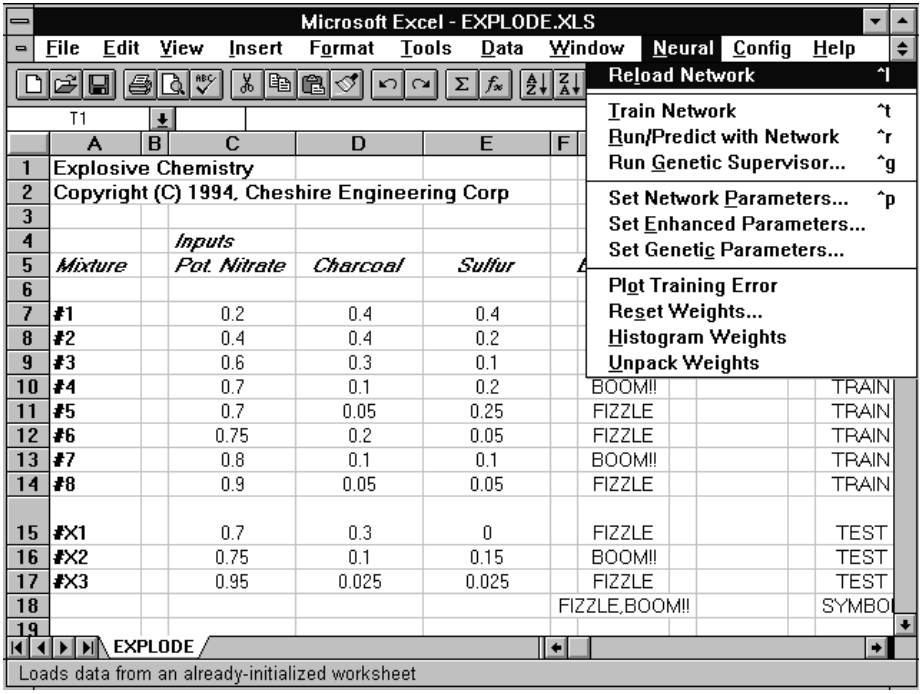


Figure 4-4 The Neuralyst Neural Menu

network described in Chapter 3. **Set Genetic Parameters** allows you to set parameters which control the Genetic training supervisor which can be used to optimize the definition and configuration of the neural network. The **Plot Training Error** command allows you to view the progress of training. The next three, **Reset Weights**, **Histogram Weights**, and **Unpack Weights**, allow you to set and view the neural network weights, providing access to the representations of learning the neural network has undergone.

[If you have installed the Trader's Macro Library or other supplementary libraries, they will appear as additional commands at the end of the Neural menu.]

For now, we can start by setting a parameter and then proceed to train the network. Select the **Set Network Parameters** command from the **Neural** menu. A dialog box will appear showing several parameters, **Learning Rate**, **Momentum**, **Input Noise**, **Training Tolerance**, **Testing Tolerance**, **Epochs per Update**,

Epoch Limit, **Time Limit**, and **Error Limit**. Each of these will have a default value.

Learning Rate determines the magnitude of the correction term applied to adjust each neuron's weights when training. Learning Rate must be positive, is adjusted in the range of 0 to 1 and has a default value of 1. Large values of Learning Rate will cause the network to train more quickly, but too large a value may cause the training to be unstable and no learning will occur.

Momentum determines the "lifetime" of a correction term as the training process takes place. Momentum must be greater than or equal to 0 but less than 1 and has a default of 0.9. Values of Momentum closer to 1 will cause the neural network to retain more of the impact of previous corrections to the current corrections. Values of Momentum close to 0 will allow mostly or only the current corrective term to have an effect. Momentum helps to smooth out the training process so that no single aberrant instance can force learning in an undesirable direction.

Input Noise provides a slight random variation to each input value for every training epoch. As training occurs, this has the effect of preventing the neural network from learning the exact input values. Ideally, this will prevent overtraining and improve the generalization process. Input Noise has a range of 0 to 1, but small values of Input Noise are generally the most useful. Input Noise represents a percentage of the range for an input, for example a value of 0.1, or 10%, means that a noise level of up to 10% of the input range will be applied. Input Noise is set to 0 as a default.

Training Tolerance defines the percentage error allowed in comparing the neural network output to the target value to be scored as "Right" during the training process. The Training Tolerance should be between 0 and 1 and has 0.1, or 10%, as a default. The Training Tolerance value has no effect on the learning algorithm. However, when Neuralyst finds 100% Right, as defined by Training Tolerance, it will automatically stop training.

Testing Tolerance is similar to Training Tolerance, but it is applied to the neural network outputs and the target values only for the test data (as defined by the value of the Mode Flag column). Neural network output and test target values are scored as "Right" if they are

within the Testing Tolerance. Otherwise they are scored as “Wrong”. The Testing Tolerance should be between 0 and 1 and has 0.3, or 30%, as a default. Testing Tolerance may be set to the same limit as Training Tolerance, but is often set to a less restrictive value since prediction is usually less exact than training.

Epochs per Update allows you to control the number of epochs between updates of the neural network results which are displayed in the Network Run Statistics block in the worksheet. An epoch is one complete processing run through all defined training cases. Epochs per Update has a default value of 1. However larger values will mean less frequent communication between the neural network and the worksheet, reducing overall training time.

Epoch Limit sets a maximum number of training epochs the neural network will undergo in those situations where you wish to control the number of training epochs rather than setting a Training Tolerance. Epoch Limit has a default value of 0, which means that there is no limit set.

Time Limit sets a maximum amount of time that the training of the neural network will undergo. This is useful if Neuralyst may be left unattended during the training process. Time Limit has a default value of 0, which means that no limit is set.

Error Limit sets a limit for an increase in the training error. Generally a neural network will steadily reduce the training error. If too much training occurs or if the neural network has insufficient capacity or an inappropriate configuration for the problem, then it is possible for the training error to increase. Error Limit allows these conditions to terminate training. Error Limit has a default value of 0, which means that no limit is set.

When one or more limits are set, the first limit that occurs, or if no limit occurs, the achievement of training within the Training Tolerance will terminate training.

Only Epochs per Update should be changed in this example. Enter 50, indicating 50 epochs between worksheet updates, for Epochs per Update. Confirm **OK**.

Finally, select the **Train Network** command from the **Neural** menu. If the configuration steps have been followed properly, the window will shift so that the Network Run Statistics block is visible and the first cell in that block, RMS Error, will be changing quickly. As training progresses, the RMS Error will be decreasing and the scores in Right and Percent Right will be increasing.

While the neural network is training, you can cause training to pause by typing the **Esc** {**Esc** or **cmd-.**} key. Try it. When the **Esc** {**Esc** or **cmd-.**} key is typed, Neuralyst will stop at the next update point and save its current work and after a moment allow you to access the worksheet. Training can be resumed by selecting the **Train Network** command from the Neural menu again. Do that now. After a few minutes (depending on the speed of your computer), the Right and Percent Right scores will be 8 and 100%, respectively. The neural network will have been trained.

When Neuralyst concludes training the neural network, it will place the current values of the neural network output in the Output column, **I**. The values in this column should match those in column **G**, the targets used to train the neural network. When the Target column is symbolic, then the Output column is filled with symbols that correspond most closely to the actual numeric values that are processed by the neural network. When the Target column is numeric, then the Output column is filled with the actual numeric values. In the numeric case, the effect of Training Tolerance is more visible as the variation from the target values can be calculated.

With the neural network trained, Neuralyst is now ready to run the neural network to predict the outcomes of experiments X1 through X3. The "known" results (based on a fictitious formula that models the behavior of all the experiments in the EXPLODE.XLS {Explode} world) have been entered in the last three lines of the Explode! column, but when Neuralyst runs the neural network in predictive mode it will not look at these in making its prediction. These results will only be used in scoring the success of the neural network.

To test the neural network, select **Run/Predict with Network** from the **Neural** menu. The results of the run will be placed in the last three rows of the Explode? column, corresponding to X1 through X3, and the scoring will now be updated to reflect the results of this test

run rather than the previous training runs. The Right and Percent Right scores will be 3 and 100%, reflecting the testing of these three additional experiments and the comparison of the test outputs against the known values.

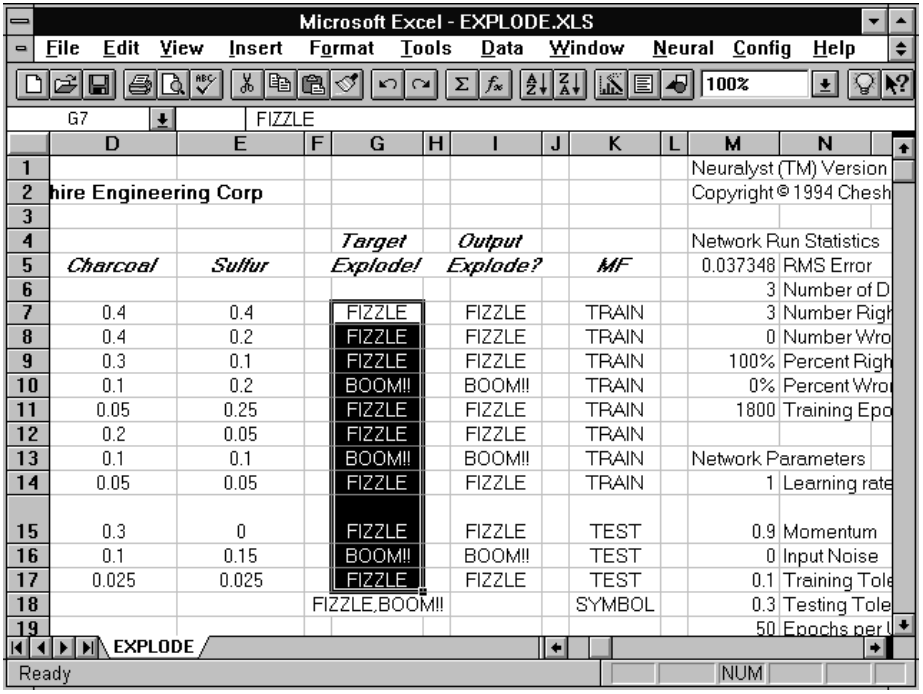


Figure 4-5 Final Screen for EXPLODE.XLS {Explode}

4.4 Finishing Up

That's it! We've just configured and run a neural network, had it learn some of the rules of chemistry of the EXPLODE.XLS {Explode} world, and been able to use it to predict the outcome of additional experiments that the neural network had not seen before!

To save this work, this Neuralyst worksheet can be saved like any other Excel worksheet: from the **File** menu choose the **Save** or **Save As** command. The **Exit** {**Quit**} command (also in the **File** menu) can be used to exit; Neuralyst will unload along with Excel.