Welcome to NeuroSolutions

Welcome to NeuroSolutions, the premier neural network simulation environment.

This manual is designed to get you up and running in the least amount of time possible. It is a concise summary of the most important topics covered in the extensive online help. We anticipate you will be able to cover all material in this manual, including examples, in less than six hours total time.

By the time you finish this manual, you will have developed a basic familiarity with the NeuroSolutions environment. To learn more about NeuroSolutions' extensive features, as well as additional tutorials, examples and an introduction to neural computing, please turn to the online help.

Packing List

Before installing, verify that you received the following materials:

- Compact disk (CD)
- Installation instructions page
- Technical support page
- *NeuroSolutions Getting Started Manual*

If any components are missing, please contact NeuroDimension immediately.

To ensure that you are kept informed of the latest product updates and releases, please complete and mail the registration card.

Technical Support

Included with Evaluation/Demo Software

- Toll-free line for bug reports and installation problems

Included with Purchased Software

- Toll-free line for bug reports and installation problems
- 1 year unlimited email, fax and phone support (toll line)

Contact Information

Bug Reports, Installation Problems and Priority Support	**1-800-634-3327 (Option 0)**
All other Technical Support	**(352) 377-1542**
Fax	**(352) 377-9009**
Email	**support@neurosolutions.com**
Calls Outside U.S.	**(352) 377-1542**

Please have your invoice number ready when calling and include your invoice number in all fax, email, or written correspondence.

Note that the technical support described above is for questions regarding the software package. Neural network experts are on staff and available for consulting on an hourly basis. Consulting rates are dependent on the specifics of the problem.

NeuroDimension, Inc.

NeuroDimension, Inc.
1800 N. Main Street, Suite D4
Gainesville, FL 32609
www.nd.com

Sales and Information

Sales	**1-800-634-3327 (Option 3)**
Product Literature and Evaluation Software	**1-800-634-3327 (Option 2)**
Fax	**352-377-9009**
Email	**info@nd.com**
Calls Outside U.S.	**352-377-5144**

Technical Support

Bug Reports, Installation Problems, and Priority Support	**1-800-634-3327 (Option 0)**
All Other Technical Support	**352-377-1542**
Fax	**352-377-9009**
Email	**support@neurosolutions.com**
Calls Outside U.S.	**352-377-1542**

Before contacting technical support, please attempt to answer any questions by first consulting the following resources:

- The printed manual (if applicable)
- The on-line help
- The Frequently Asked Questions (FAQ)

The latest versions of the on-line help and FAQ can always be found at the NeuroDimension web site: www.nd.com.

System Requirements

Before installing NeuroSolutions, you should verify that your system configuration meets the following minimum specifications:

Operating System	Windows 95/98/2000/NT/Me
Memory	16MB RAM (32MB recommended)
Hard Drive	40MB free disk space
Video	640x480 with 256 colors (800x600 with 16M colors recommended)

Installation Instructions

1) Install the Software

Insert the CD-ROM into the drive. The installation program should start automatically. If this does not happen, then Run the program "autorun.exe" from the root CD directory.

The installation program is highly automated and easy to use. Follow the on-screen instructions. It is recommended that first-time users perform the typical installation.

2) Activate the Software

NeuroSolutions is initially in evaluation mode and must be activated in order to remove the evaluation restrictions. Once you purchase a license for NeuroSolutions you will receive an email with instructions for obtaining an activation code that you will use to activate the software. If you purchased and did not receive this email or have misplaced it, please contact NeuroDimension to have another copy sent to you.

Summary of Installed Components

If you have chosen the default installation, NeuroSolutions will install the following items in the Windows *Start* Menu Start, *Start⇒Programs⇒NeuroSolutions 4*:

Item	**Description**
Getting Started Manual	This manual in online form
Interactive Book Preface and TOC	Preface and Table of Contents on an interactive book on neural networks (Chapter 1 is included)
NeuralBuilder	A utility for constructing neural networks based on the neural model desired
NeuralExpert	A utility for constructing neural networks based on the application type
NeuroSolutions for Excel Demos	A demo of an Excel add-in product for NeuroSolutions
NeuroSolutions for Excel Help	Online documentation for NeuroSolutions for Excel
NeuroSolutions for Excel	Launches Microsoft Excel and loads the NeuroSolutions for Excel add-in
NeuroSolutions	The main NeuroSolutions program
NeuroSolutions Help	Online documentation for NeuroSolutions

If you performed the default installation, the installation program created the directory "C:\Program Files\NeuroSolutions 4". Within this directory tree are the sub-directories and files of the NeuroSolutions package. You will not typically need to directly access most of the sub-directories; however, three are of interest to beginning users:

Sub-Directory	**Description**
DLLCust	Contains some ready-to-use Dynamic Link Libraries (DLL's)
Macros	Contains common macros
Tutorials	A set of saved breadboards and related data that correspond to examples in the online help

Uninstalling NeuroSolutions

From the Windows *Start* Menu Start:

§ Go to *Start*⇒*Settings*⇒*Control Panel*.

§ Double-click "Add/Remove Programs".

§ Select "NeuroSolutions 4" in the list and click the "Add/Remove" button.

Sometimes the uninstall program may leave behind a few files. You may delete these manually.

Level Summary

Features common to all levels:

§ Icon-based construction of neural networks
§ Component-level editing of parameters
§ Macro recording and playback
§ OLE-compatible server
§ Online, context-sensitive help
§ NeuralBuilder and NeuralExpert neural network construction utilities

Level-specific features:

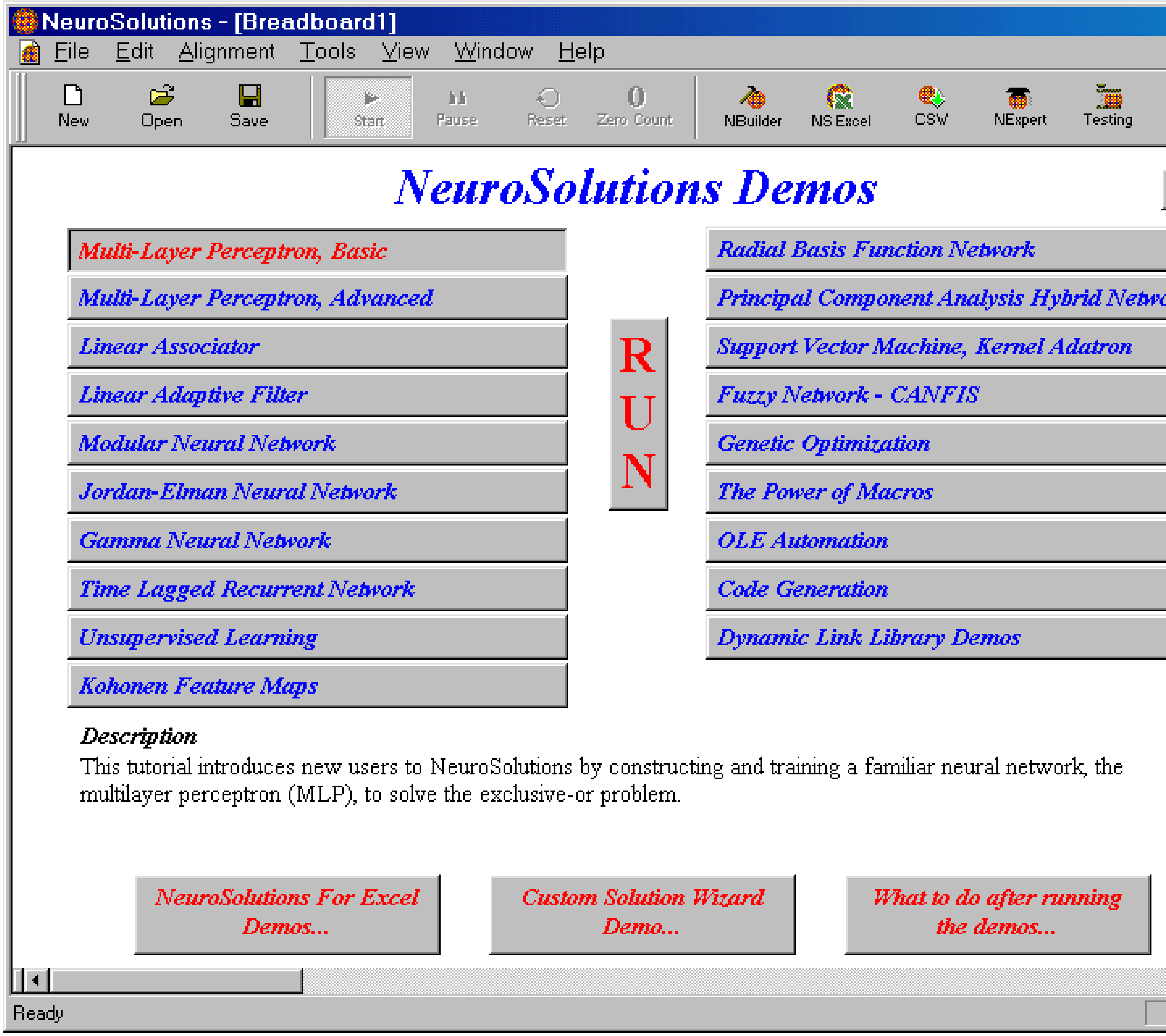
Level	**Topologies**	**Learning Paradigms**	**Hidden Layers**	**Neurons per Hidden Layer**
Educator	§ Multilayer Perceptron (MLP) § Generalized feedforward	§ Backpropagation	2	50
Users	§ Educator + § Modular networks § Hebbian § Principal component analysis (PCA) § Competitive § Kohonen feature maps § Neuro-Fuzzy § Support Vector Machines	§ Educator + § Unsupervised § Genetic § Adatron (SVM)	6	500
Consultants	§ Users + § Hopfield, time delay (TDNN) § Time lagged recurrent (TLRN)	§ Users + § Backpropagation through time (BPTT) § Fixed point recurrent learning	unlimited	unlimited
Professional	§ Consultants + § ANSI C++ Source Code Generation	§ Same as Consultants	unlimited	unlimited
Developers Lite	§ Consultants + § User-defined dynamic link libraries (DLL's)	§ Same as Consultants	unlimited	unlimited
Developers	§ Consultants + § ANSI C++ source code generation § User-defined dynamic link libraries (DLL's)	§ Same as Consultants	unlimited	unlimited

Starting NeuroSolutions

The easiest way to start NeuroSolutions is from the Windows *Start* button Start:

§ Go to *Start⇒Programs⇒NeuroSolutions 4⇒NeuroSolutions*.

The first time you run the program, the ***NeuroSolutions Demo*** panel will appear automatically.


NeuroSolutions - [Breadboard1]
File
Edit
Alignment
Tools
View
Window
Help
New
Open
Save
Start
Pause
Reset
Zero Count
NBuilder
NS Excel
CSW
NExpert
Testing
NeuroSolutions Demos
Multi-Layer Perceptron, Basic
Multi-Layer Perceptron, Advanced
Linear Associator
Linear Adaptive Filter
Modular Neural Network
Jordan-Elman Neural Network
Gamma Neural Network
Time Lagged Recurrent Network
Unsupervised Learning
Kohonen Feature Maps
RUN
Radial Basis Function Network
Principal Component Analysis Hybrid Netwo
Support Vector Machine, Kernel Adatron
Fuzzy Network - CANFIS
Genetic Optimization
The Power of Macros
OLE Automation
Code Generation
Dynamic Link Library Demos
NeuroSolutions For Excel Demos...
Custom Solution Wizard Demo...
What to do after running the demos...
Ready

Description

This tutorial introduces new users to NeuroSolutions by constructing and training a familiar neural network, the multilayer perceptron (MLP), to solve the exclusive-or problem.

Demo selection panel

Running the Demos

The best way to get an overview of the features provided by NeuroSolutions is to run the demos. These demos present a series of examples in neural computing, which illustrate the broad range of capabilities NeuroSolutions has to offer.

To launch the ***NeuroSolutions Demo*** panel:

§ Go to the *Help* menu and select "Demos," or type "Ctrl+D."

To run a demo,

§ Make a selection by clicking one of the demo buttons.

The selected button's text will turn red, and the selected demo will be described.

§ Click the *Run* button.

The two buttons at the bottom are the exception; they do not use the *Run* button.

The demos present a series of panels, each one introducing a particular feature of NeuroSolutions.

§ To advance to the next panel, click the *Forward* button ».

§ To quit the present demo and return to the main demo menu, click the *Cancel* button ✕.

All demos are live! Each one starts with random initial conditions and learns online. These demos are also designed to be interactive. Many of the panels have edit cells that allow you to modify the network parameters and buttons to run the simulation or single-step through an epoch.

These demos are not restrictive. At any time during the presentation, you are able to manipulate the breadboard by adding a component, removing a component, or changing a component's parameters with the Inspector. This can be advantageous in that you can learn a lot about the software by experimenting with the components. However, changing the state of the breadboard may result in an error message later in the demo. If the demo does fail, click the *Cancel* button ✕ to get you back to main demo menu. Or, simply restart the demo.

Kohonen Feature Maps

✕ »

A more difficult 1-D problem

One of the appealing properties of kohonen maps is their ability to "cover" the input space, even where there are no well defined clusters. Here the input consists of a 16 sample sine wave with additive uniform noise. Run the network and observe that, even though the input data density is contiguous, the line of kohonen weights covers the entire sine wave. Notice also that the input data density is slightly higher at the extremum of the sine wave, and that more Kohonen weights relocate there.

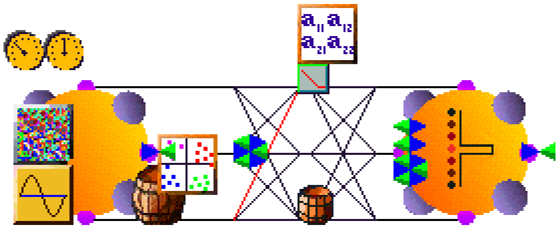
Run

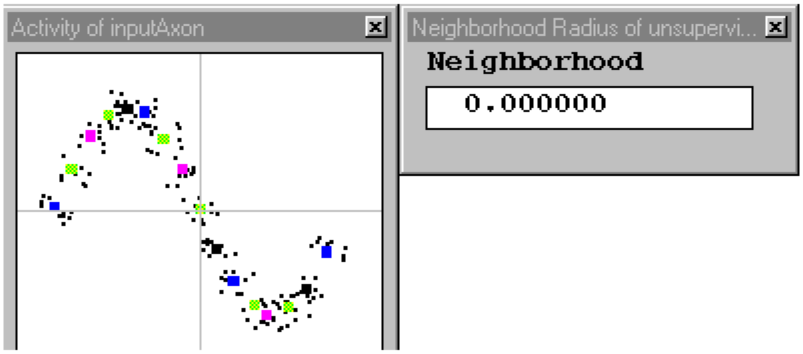
# of PE's

16

Initial neighborhood width

8





Neighborhood

0.000000

A running demo

Using the Online Help

In addition to running the demos, the next best way to learn about NeuroSolutions is to browse the online help. The online help is extensive, with several hundred topics documenting all features of NeuroSolutions. It also has additional examples and tutorials, as well as an introduction to neural networks.

Launching the Online Help

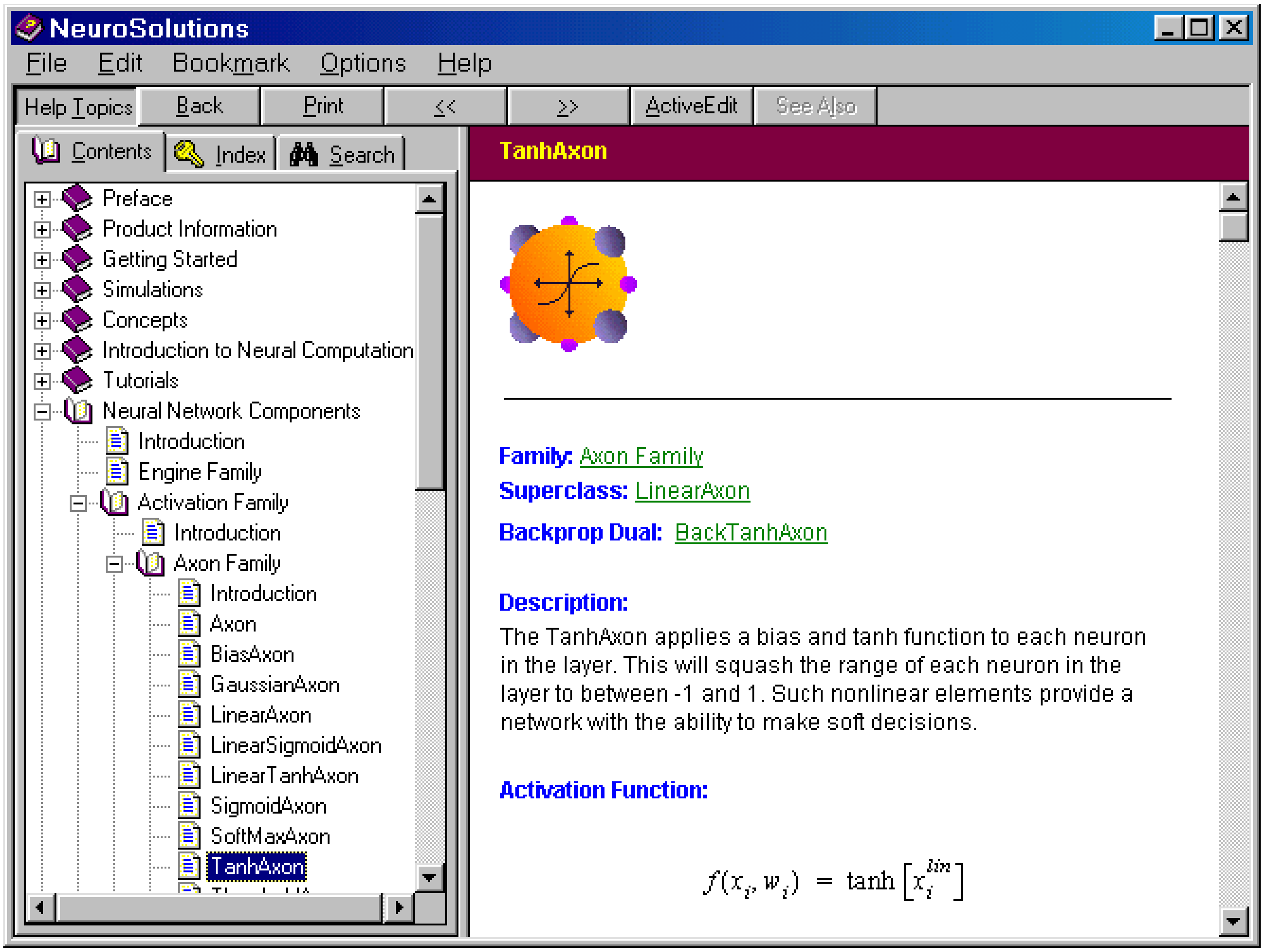
To launch the online help from the Windows *Start* Menu Start:

Go to *Start⇒NeuroSolutions 4⇒NeuroSolutions Help*.

To launch the online help from within NeuroSolutions:

§ Go to the *Help* menu and select "NeuroSolutions Help", or press F1.

You may then get help by topic, index or keyword.



TanhAxon

Family: Axon Family

Superclass: LinearAxon

Backprop Dual: BackTanhAxon

Description:

The TanhAxon applies a bias and tanh function to each neuron in the layer. This will squash the range of each neuron in the layer to between -1 and 1. Such nonlinear elements provide a network with the ability to make soft decisions.

Activation Function:

$$f(x_i, w_i) = \tanh\left[x_i^{lin}\right]$$

NeuroSolutions Help Topics

The *Getting Started Manual* is also available online. Simply launch it as you would the main online help. This manual may be useful when you try the examples.

Context-Sensitive Help

NeuroSolutions also provides context-sensitive help. To access context-sensitive help:

§ Go to the *Help* menu and choose "Context Help", or press "Shift-F1".

Then click on any icon in the palettes, any component on the breadboard, or a property page of the Inspector, and the help file will automatically open to the associated topic.

Printing the Online Help

From the main help topics page, you can print out the entire help file. To print out a single chapter, open that chapter from the main help topics page and then print. Alternatively, you can print out an individual topic from a topic page. The entire documentation is also available in Microsoft Word format from the Download section of the NeuroDimension web site:

http://www.nd.com/download.htm

NeuroSolutions FAQ

Within the online help you will find a special section called the NeuroSolutions Frequently Asked Questions (FAQ), a compendium of the most common questions received by technical support. Beginning users may want to consult it before searching the general help.

Four Ways to Construct a Neural Network

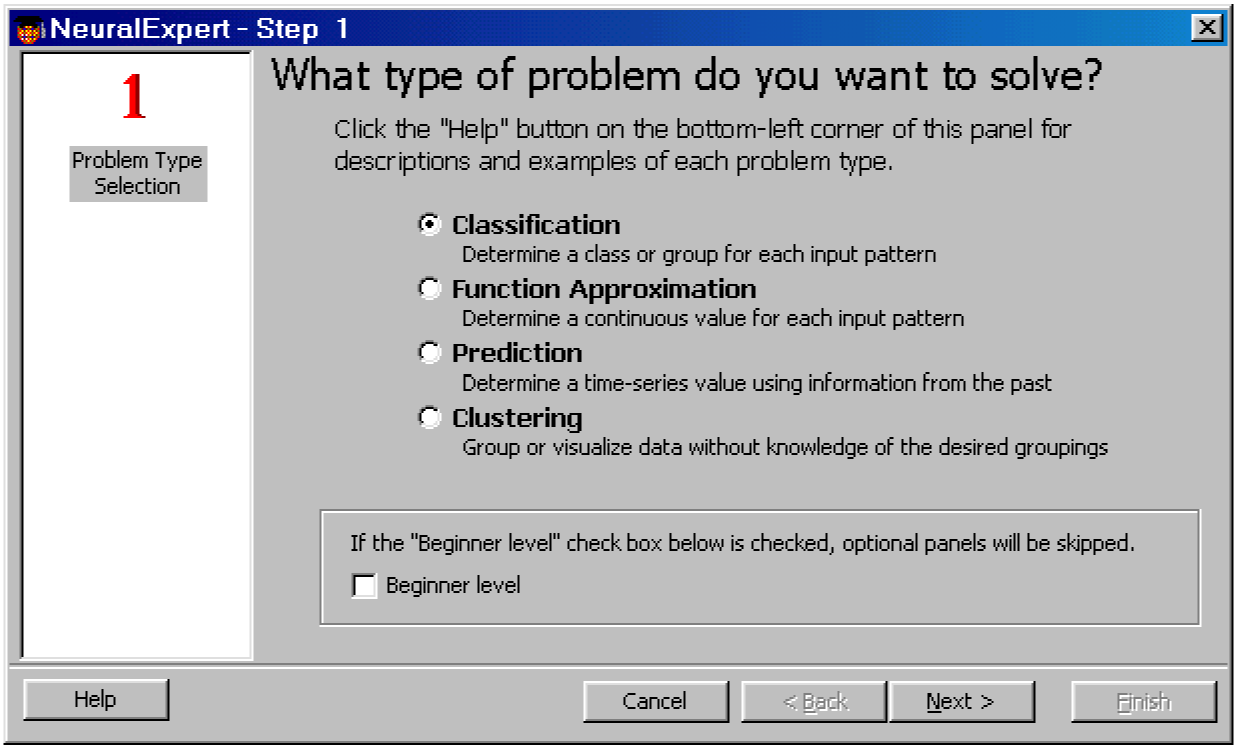
There are four ways to build a neural network:

1) Run the NeuralExpert program.
2) Run the NeuralBuilder program.
3) Run a pre-recorded macro (such as one of the demos) and modify the resulting network.
4) Manually construct a network from a blank breadboard.

For the beginning user the easiest way to build a network is to use the NeuralExpert. More advanced users who want more control of the topology and parameter settings may want to use the NeuralBuilder. It is important to note that breadboards built with either the NeuralExpert or NeuralBuilder can be modified later.

The NeuralExpert

The NeuralExpert asks you questions and intelligently builds a neural network. It configures the parameters and probes based on your description of the problem to be solved. Once you select a problem type you will see all the questions you will need to answer in the panel on the left. You can click on these steps/numbers to navigate through the question and answer session. Once the network is built, you can modify the settings either directly on the breadboard or within the NeuralExpert.



NeuralExpert - Step 1

1

Problem Type Selection

What type of problem do you want to solve?

Click the "Help" button on the bottom-left corner of this panel for descriptions and examples of each problem type.

● Classification
Determine a class or group for each input pattern

○ Function Approximation
Determine a continuous value for each input pattern

○ Prediction
Determine a time-series value using information from the past

○ Clustering
Group or visualize data without knowledge of the desired groupings

If the "Beginner level" check box below is checked, optional panels will be skipped.

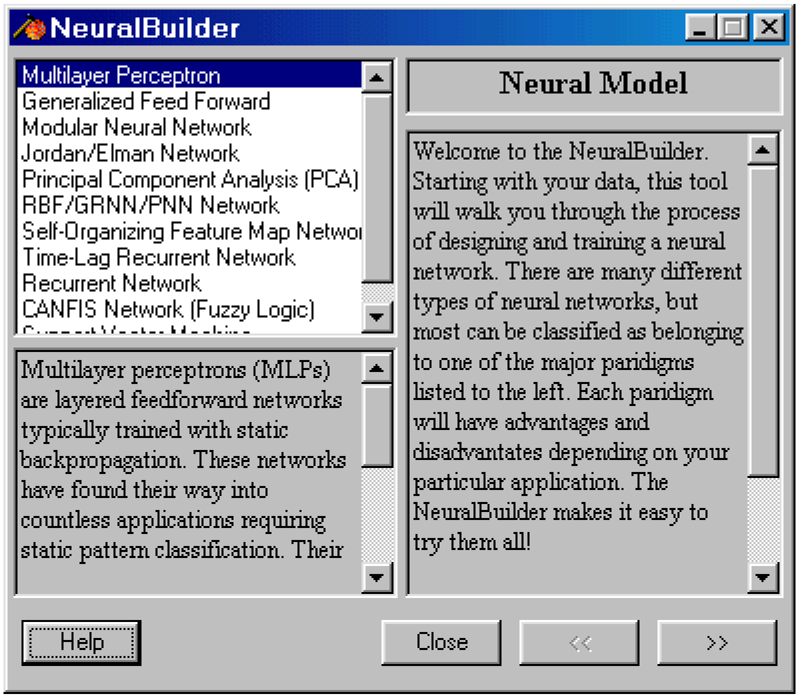
□ Beginner level

NeuralExpert Opening Panel

The NeuralBuilder

The NeuralBuilder is targeted for more advanced users. It presents a series of panels that represent logical steps in the neural network design process. At each panel you make choices and occasionally enter parameters. Instead of asking you questions based on the problem type, as with the NeuralExpert, the NeuralBuilder allows you the specify the neural network based on a particular neural model (topology).

Unlike the NeuralExpert, any modifications to the network must be made directly to the breadboard – the NeuralBuilder does not have an edit capability. However, the NeuralBuilder can be minimized and kept in the background, with all entered data still intact. At any time, you can backtrack, make design changes, and then build another network.



Multilayer perceptrons (MLPs) are layered feedforward networks typically trained with static backpropagation. These networks have found their way into countless applications requiring static pattern classification. Their

Welcome to the NeuralBuilder. Starting with your data, this tool will walk you through the process of designing and training a neural network. There are many different types of neural networks, but most can be classified as belonging to one of the major paridigms listed to the left. Each paridigm will have advantages and disadvantates depending on your particular application. The NeuralBuilder makes it easy to try them all!

NeuralBuilder Opening Panel

Starting the NeuralExpert

These following topics will take you through the various question panels of the NeuralExpert. At each panel, the basic options will be explained, and suggestions offered. You are encouraged to try the examples that are presented at the end of each panel's description. Each example continues throughout the chapter.

To launch the NeuralExpert from the Windows *Start* Menu Start:

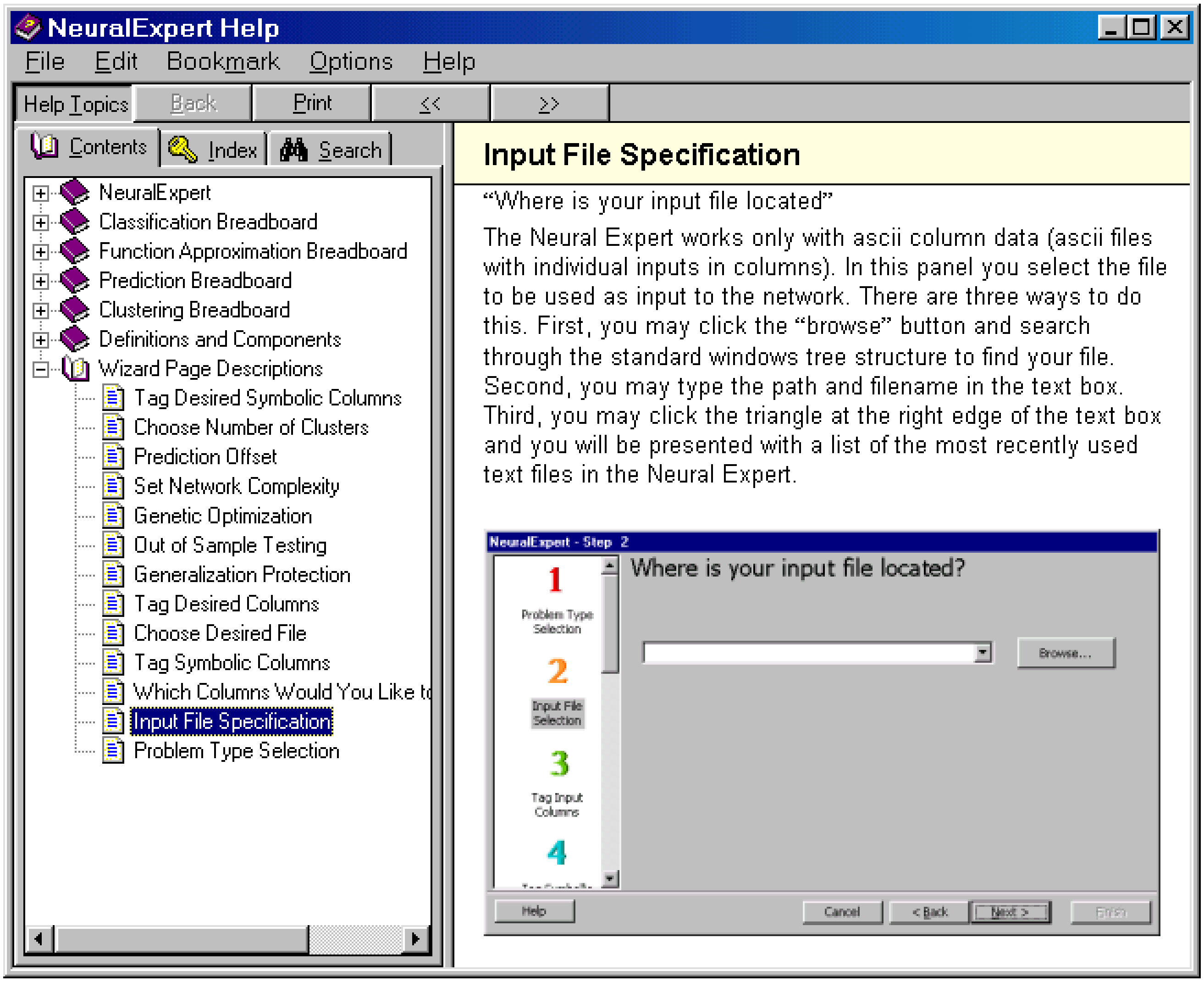
§ Select *Start⇒NeuroSolutions 4⇒NeuralExpert⇒NeuralExpert*.

To launch the NeuralExpert from within NeuroSolutions:

§ Go to the *Tools* menu and choose "NeuralExpert" or click the "NExpert" toolbar button .

Getting Help in the NeuralExpert

Online help is available from all NeuralExpert panels. To access help for the current panel, click the *Help* button in the lower left corner of the wizard.



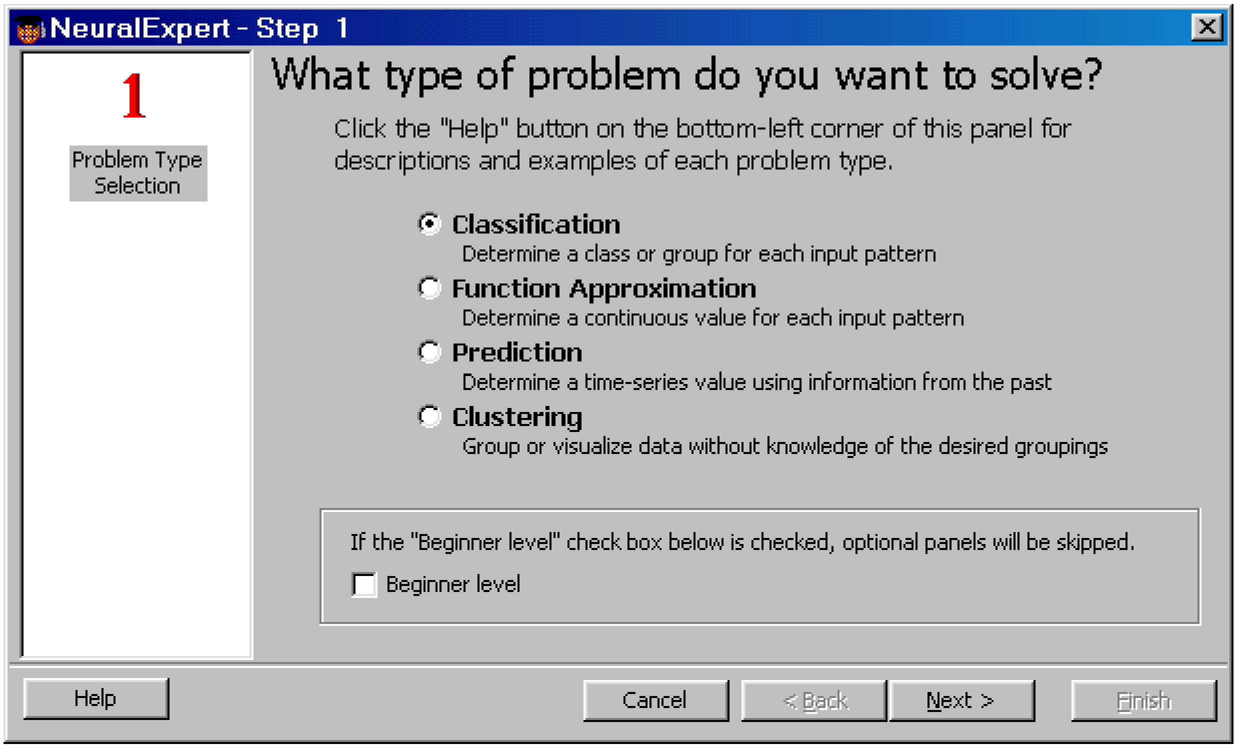
Input File Specification

"Where is your input file located"

The Neural Expert works only with ascii column data (ascii files with individual inputs in columns). In this panel you select the file to be used as input to the network. There are three ways to do this. First, you may click the "browse" button and search through the standard windows tree structure to find your file. Second, you may type the path and filename in the text box. Third, you may click the triangle at the right edge of the text box and you will be presented with a list of the most recently used text files in the Neural Expert.

NeuralExpert Help Topic for the Input File Specification Panel

NeuralExpert Problem Type Selection Panel



What type of problem do you want to solve?

Click the "Help" button on the bottom-left corner of this panel for descriptions and examples of each problem type.

● Classification
Determine a class or group for each input pattern

○ Function Approximation
Determine a continuous value for each input pattern

○ Prediction
Determine a time-series value using information from the past

○ Clustering
Group or visualize data without knowledge of the desired groupings

If the "Beginner level" check box below is checked, optional panels will be skipped.

□ Beginner level

NeuralExpert Problem Type Selection Panel

The first step in building a neural network with the NeuralExpert is the specification of the problem type. The four currently available problem types in the NeuralExpert are Classification, Prediction, Function Approximation, and Clustering. Please see the NeuralExpert help file for detailed descriptions of these four problem types. If your problem does not fit one of these descriptions (or you would like to completely specify the architecture and parameters yourself), then you may want to use the NeuralBuilder instead.

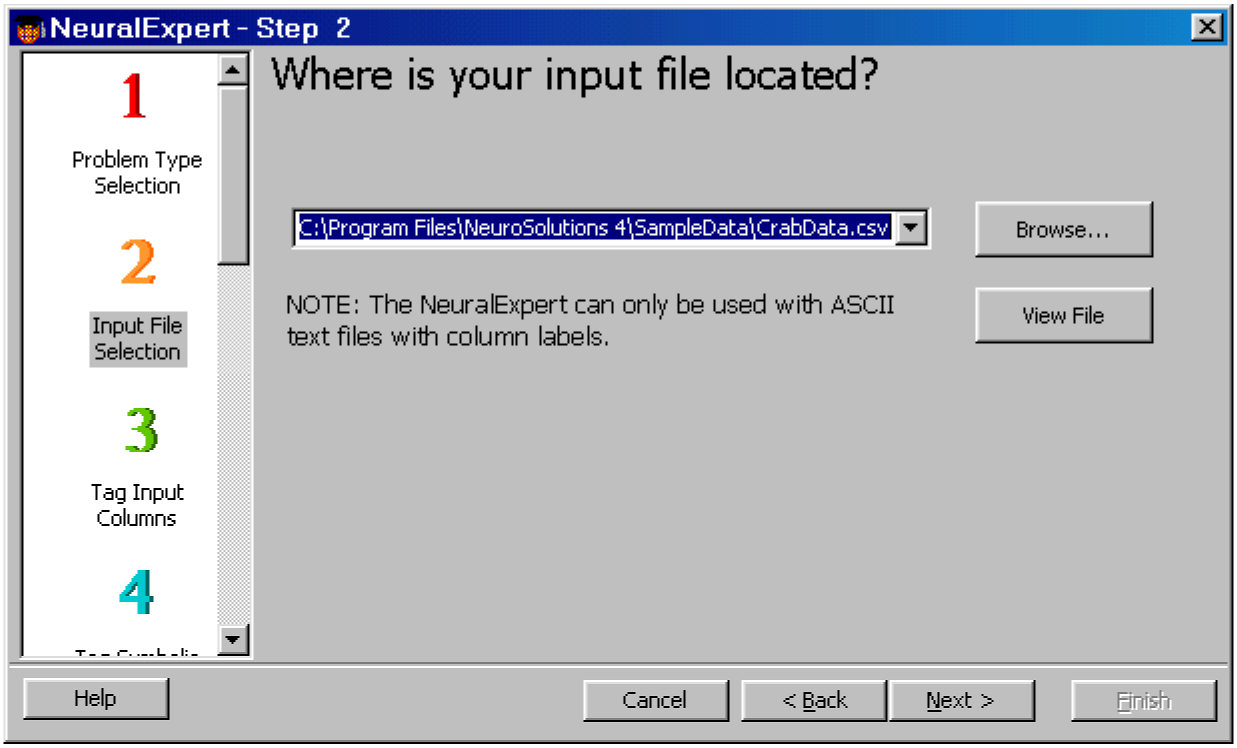
Example

ü Use the default "Classification" in the NeuralExpert ***Problem Type Selection*** panel.

ü For this example, uncheck the "Beginner level" switch so that we can examine some of the advanced panels.

ü Click the *Next* button Next > to advance to the next panel.

NeuralExpert Input File Selection Panel



Where is your input file located?

C:\Program Files\NeuroSolutions 4\SampleData\CrabData.csv

NOTE: The NeuralExpert can only be used with ASCII text files with column labels.

NeuralExpert Input File Selection Panel

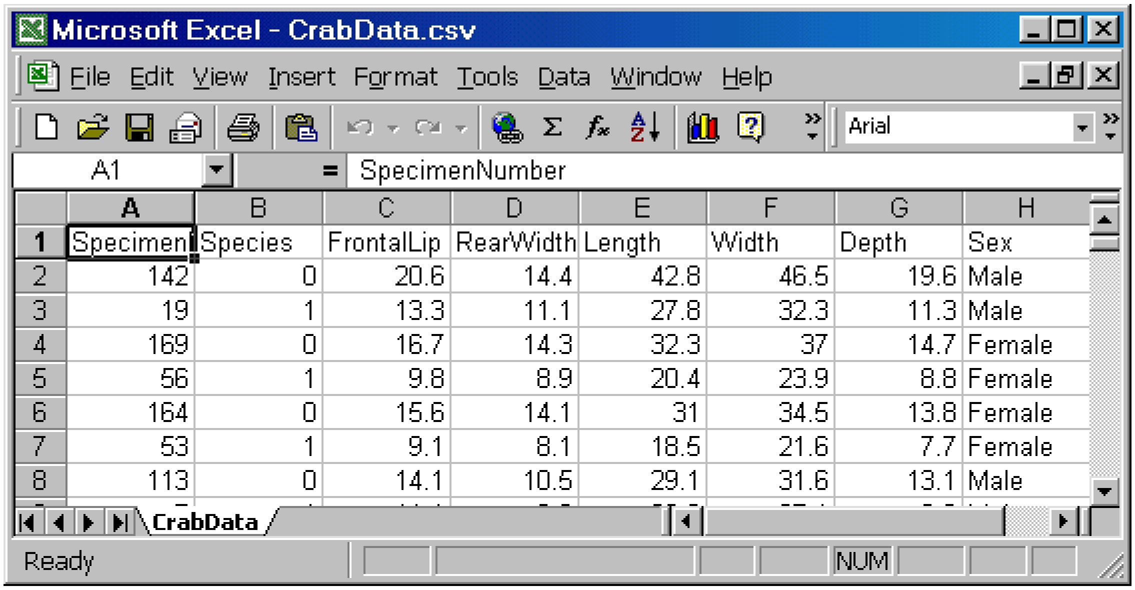
The next step in constructing your neural model is to select the input data. The ***Input File Selection*** panel is where you specify where the input data file is located. There are three ways to do this:

1) Click the "Browse" button and search through the standard windows tree structure to find your file.
2) Type the path and filename in the text box.
3) Click the triangle at the right edge of the text box and you will be presented with a list of the most recently used text files in the NeuralExpert.

Please see the section on File Format Requirements before making this selection.

Example

The sample data we will use contain various attributes of stone crab specimens. There are 50 male and 50 female specimens for each of two species (blue form and orange form) for a total of 200 specimens.   The columns labeled "Species", "Frontal Lip", "Rear Width", "Length", "Width", and "Depth" will serve as inputs to the neural network. The goal is to train a neural network to determine the sex of a specimen (male or female) based on these attributes.


Microsoft Excel - CrabData.csv
File Edit View Insert Format Tools Data Window Help
Arial
A1
=
CrabData
Ready
NUM

A1 = SpecimenNumber

	A	B	C	D	E	F	G	H
1	Specimen	Species	FrontalLip	RearWidth	Length	Width	Depth	Sex
2	142	0	20.6	14.4	42.8	46.5	19.6	Male
3	19	1	13.3	11.1	27.8	32.3	11.3	Male
4	169	0	16.7	14.3	32.3	37	14.7	Female
5	56	1	9.8	8.9	20.4	23.9	8.8	Female
6	164	0	15.6	14.1	31	34.5	13.8	Female
7	53	1	9.1	8.1	18.5	21.6	7.7	Female
8	113	0	14.1	10.5	29.1	31.6	13.1	Male

Crab Classification Data used for the NeuralExpert Example

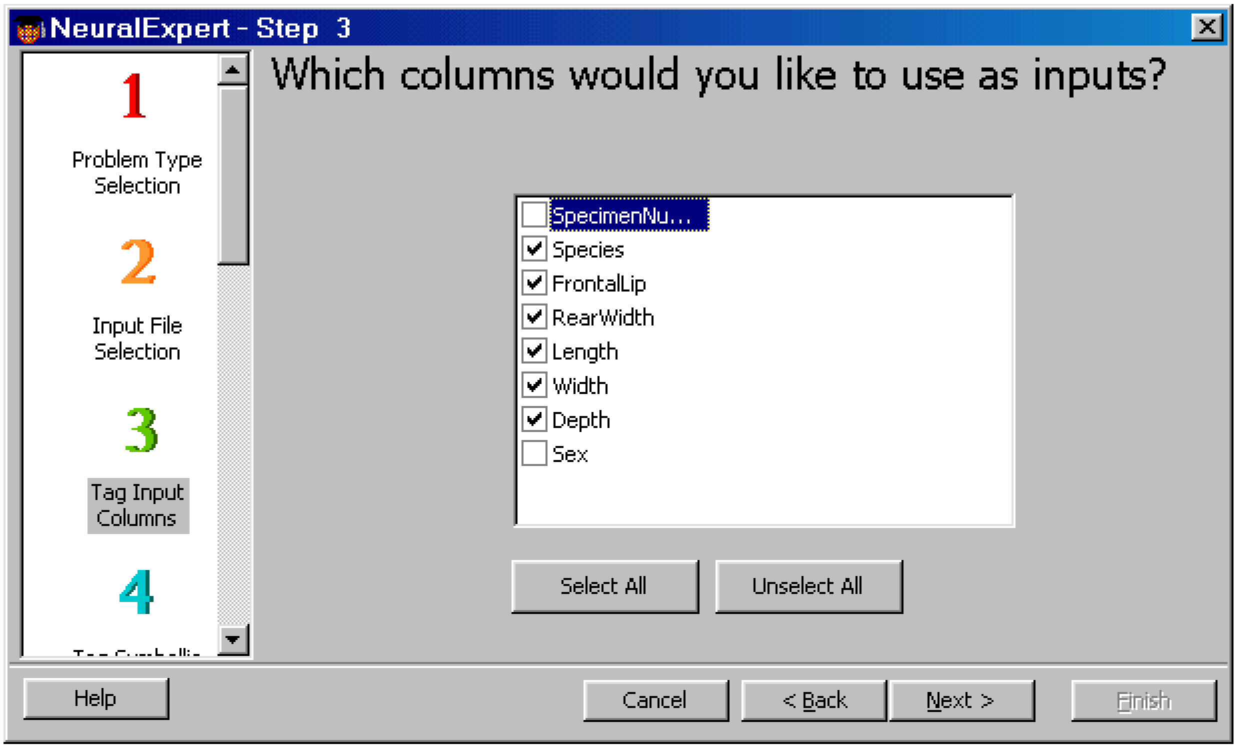
ü Click the *Browse* button Browse... . The ***Open*** panel will display.

ü Navigate to the file "…\NeuroSolutions 4\SampleData\CrabData.csv" and double-click it.

ü Click the *View File* button to view the file's contents (optional).

ü Click the *Next* button Next > to advance to the next panel.

NeuralExpert Tag Input Columns Panel


NeuralExpert - Step 3

Which columns would you like to use as inputs?

☐ SpecimenNu...
☑ Species
☑ FrontalLip
☑ RearWidth
☑ Length
☑ Width
☑ Depth
☐ Sex

NeuralExpert Tag Input Columns Panel

The next step in constructing your neural model is to tag the input columns. The ***Tag Input Columns*** panel is where you specify which data you would like to feed into the neural network. ASCII column data typically has column labels as the first row. If they do not, the wizard will ask you if you would like to add column labels.

Example

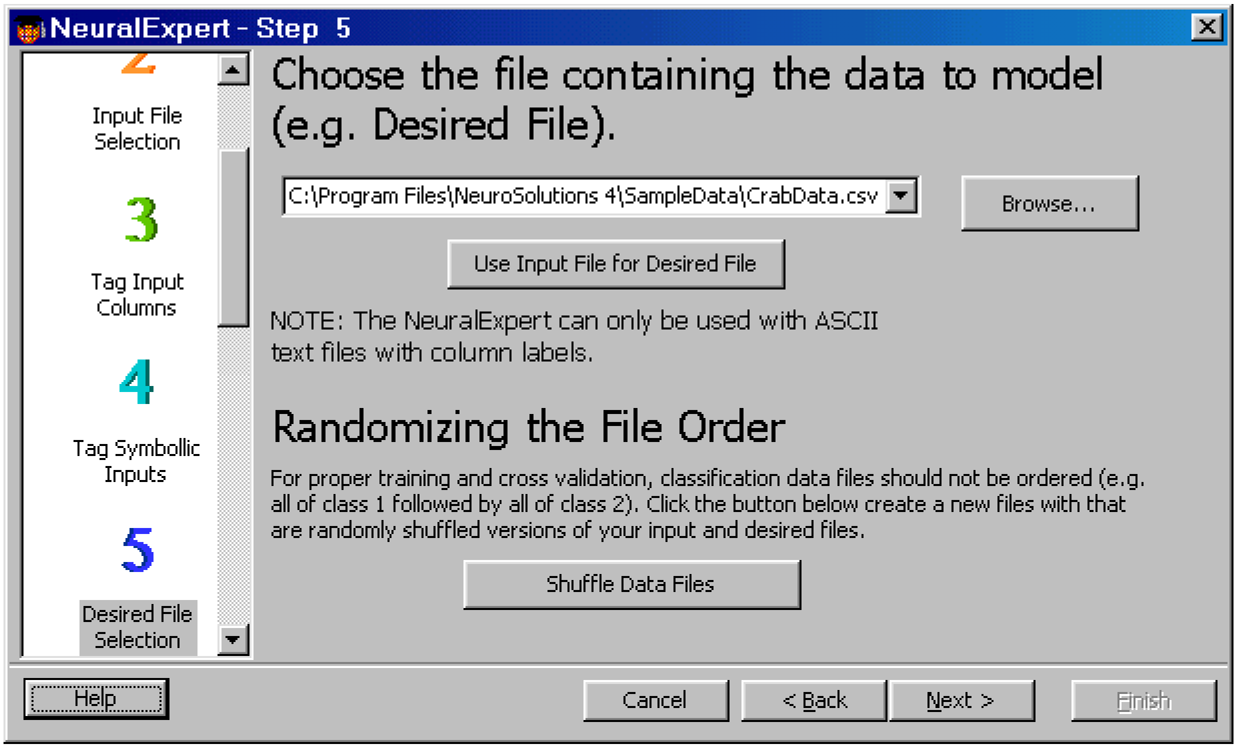
The first column of this data is the specimen number, which is not useful information in classifying the sex. The last column is the desired output ("Male" or "Female"). The rest of the columns will be used as inputs to the network.

ü Uncheck the first item ("SpecimenNumber") and the last item ("Sex").

ü Click the *Next* button Next > to advance to the next panel.

ü All of the input columns have numeric data, so click the *Next* button Next > again to skip the *Tag Symbolic Inputs* panel.

NeuralExpert Desired File Selection Panel



NeuralExpert - Step 5

Choose the file containing the data to model (e.g. Desired File).

C:\Program Files\NeuroSolutions 4\SampleData\CrabData.csv

NOTE: The NeuralExpert can only be used with ASCII text files with column labels.

Randomizing the File Order

For proper training and cross validation, classification data files should not be ordered (e.g. all of class 1 followed by all of class 2). Click the button below create a new files with that are randomly shuffled versions of your input and desired files.

NeuralExpert Desired File Selection Panel

The next step in constructing your neural model is to select the desired output data. The ***Desired File Selection*** panel is where you specify where the desired output data file is located. There are four ways to do this:

1. Click the "Browse" button and search through the standard windows tree structure to find your file.
2. Type the path and filename in the text box.
3. Click the triangle at the right edge of the text box and you will be presented with a list of the most recently used text files in the NeuralExpert.
4. Click the "Use Input File as Desired File" button, which will place the input file name in the desired file text box.

Please see the section on File Format Requirements before making this selection.

For classification problems you will be given the option to randomize the order of your data before presenting it to the network. Neural networks train better if the presentation of the data is not ordered. For instance, if you are classifying between two classes male and female, the network will train much better if the male and female data are intermixed, rather than all the males followed by all the females. If your data is highly ordered, you should randomize the order before training the neural network.

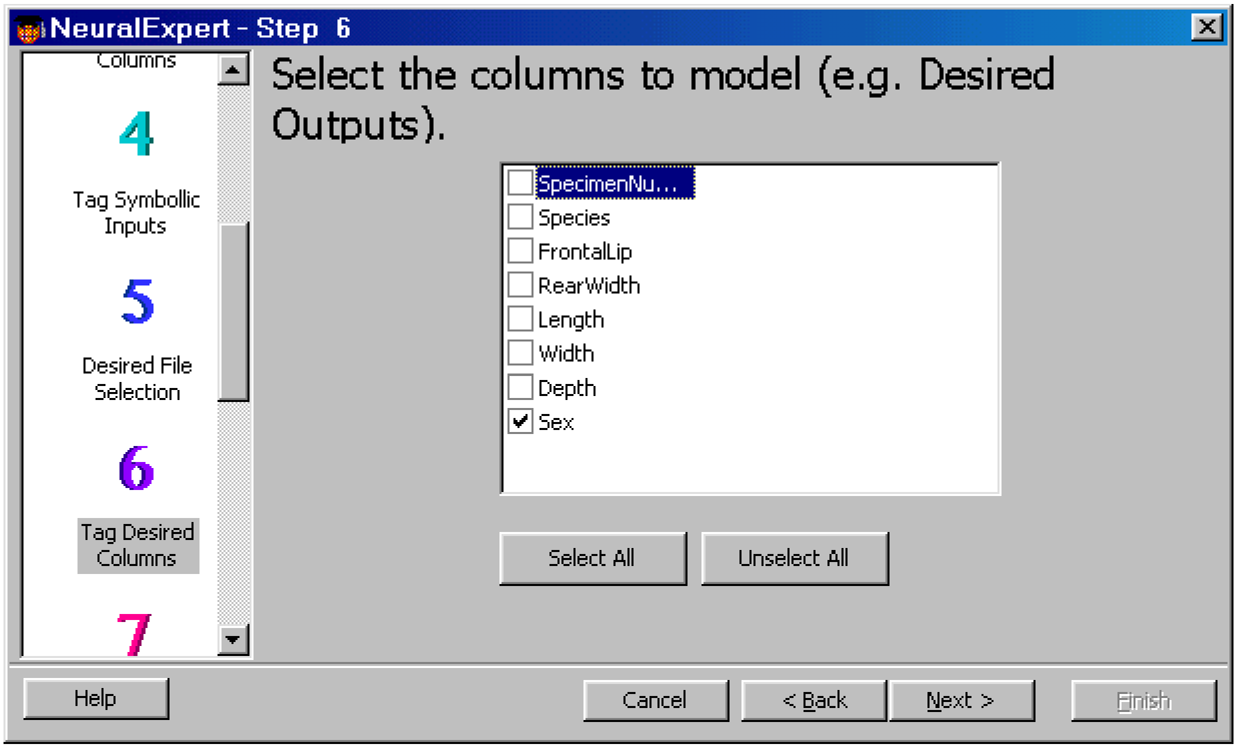
Example

The file we chose for our input columns also contains the desired output column – the sex of the specimen. The rows of this file are already randomized, so there is no need to perform this operation again.

ü Click the *Use Input File as Desired File* button Use Input File for Desired File. The path of the input file should appear in the text box.

ü Click the *Next* button Next > to advance to the next panel.

NeuralExpert Tag Desired Columns Panel



Select the columns to model (e.g. Desired Outputs).

□ SpecimenNu...
□ Species
□ FrontalLip
□ RearWidth
□ Length
□ Width
□ Depth
☑ Sex

NeuralExpert Tag Desired Columns Panel

The next step in constructing your neural model is to tag the desired output columns. The ***Tag Desired Columns*** panel is where you specify which data you would like the neural network to produce.

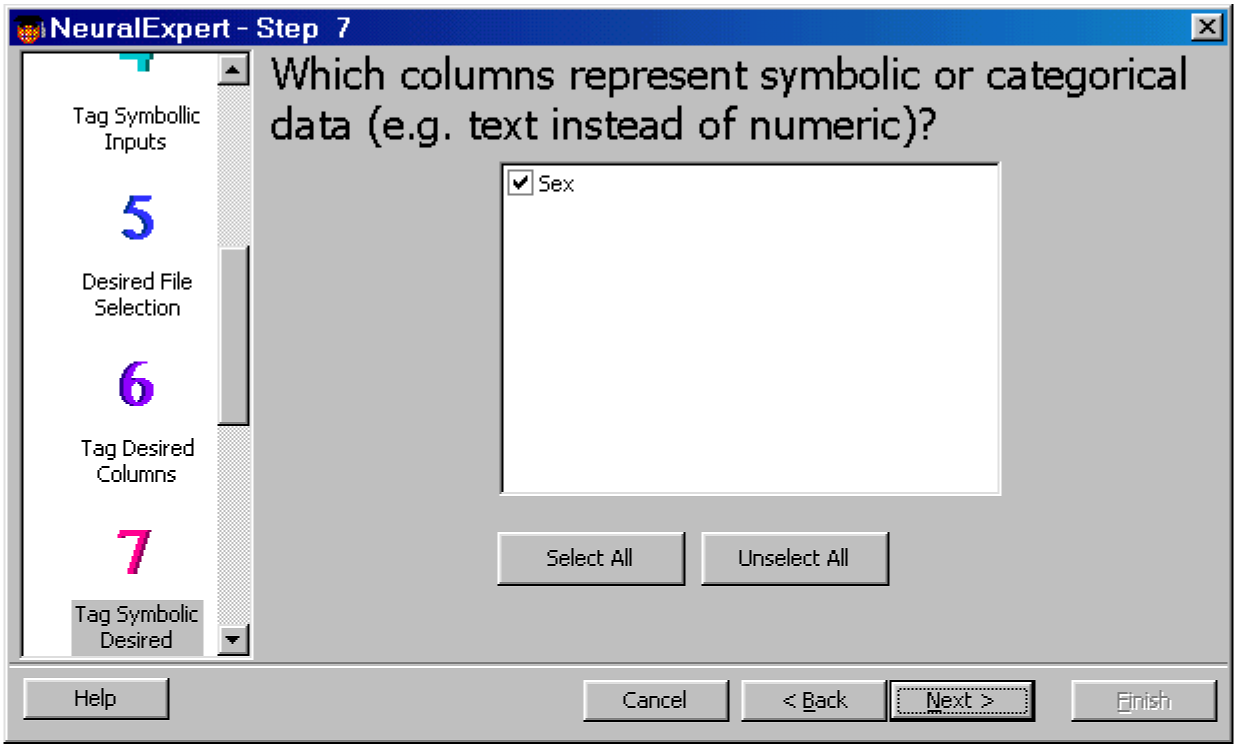
Example

The first column of this data is the specimen number, which is not something that we want to try to classify. The next six columns have already been specified as the input data. The last column is the desired output ("Sex").

ü Uncheck the first item ("SpecimenNumber"), but leave the last item ("Sex") checked.

ü Click the *Next* button Next > to advance to the next panel.

NeuralExpert Tag Symbolic Desired Panel


NeuralExpert - Step 7

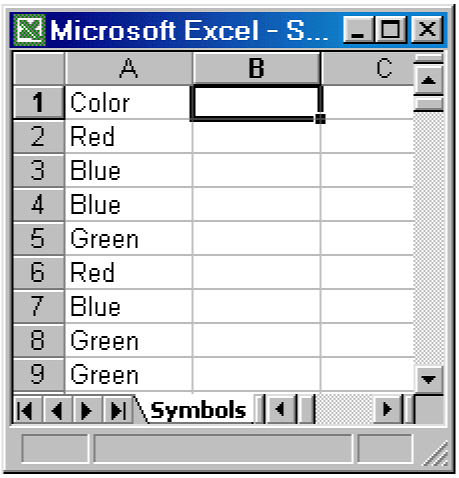
Which columns represent symbolic or categorical data (e.g. text instead of numeric)?

☑ Sex

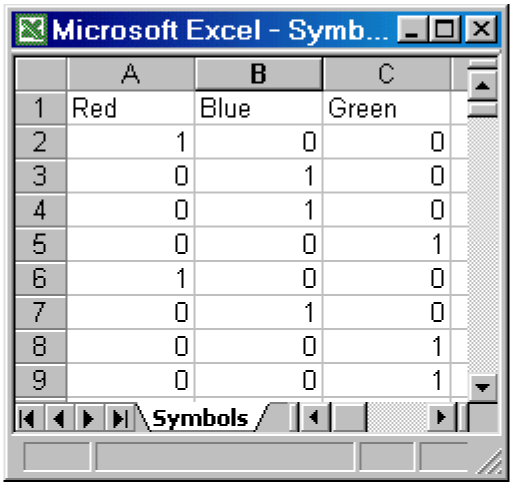
NeuralExpert Tag Symbolic Desired Panel

The ***Tag Desired Columns*** panel is used to specify which (if any) of the desired output columns contain symbolic data. Symbolic columns are those in which each data element is a string of characters (e.g. "yes"/"no"). Most often symbolic strings are non-numeric, but columns containing discrete (non-continuous) numeric values should usually be tagged as symbolic also.

NeuroSolutions translates a symbolic column by expanding it to N columns, where N is the number of unique strings in the column. Each expanded column represents a particular string. A "1" in an expanded column indicates the occurrence of the column's corresponding string and a "0" indicates a non-occurrence. The figure below illustrates a symbolic column before and after the symbolic translation process.


Microsoft Excel - S...

	A	B	C
1	Color		
2	Red		
3	Blue		
4	Blue		
5	Green		
6	Red		
7	Blue		
8	Green		
9	Green		



	A	B	C
1	Red	Blue	Green
2	1	0	0
3	0	1	0
4	0	1	0
5	0	0	1
6	1	0	0
7	0	1	0
8	0	0	1
9	0	0	1

Symbolic Column Before (left) and After (right) Symbolic Translation

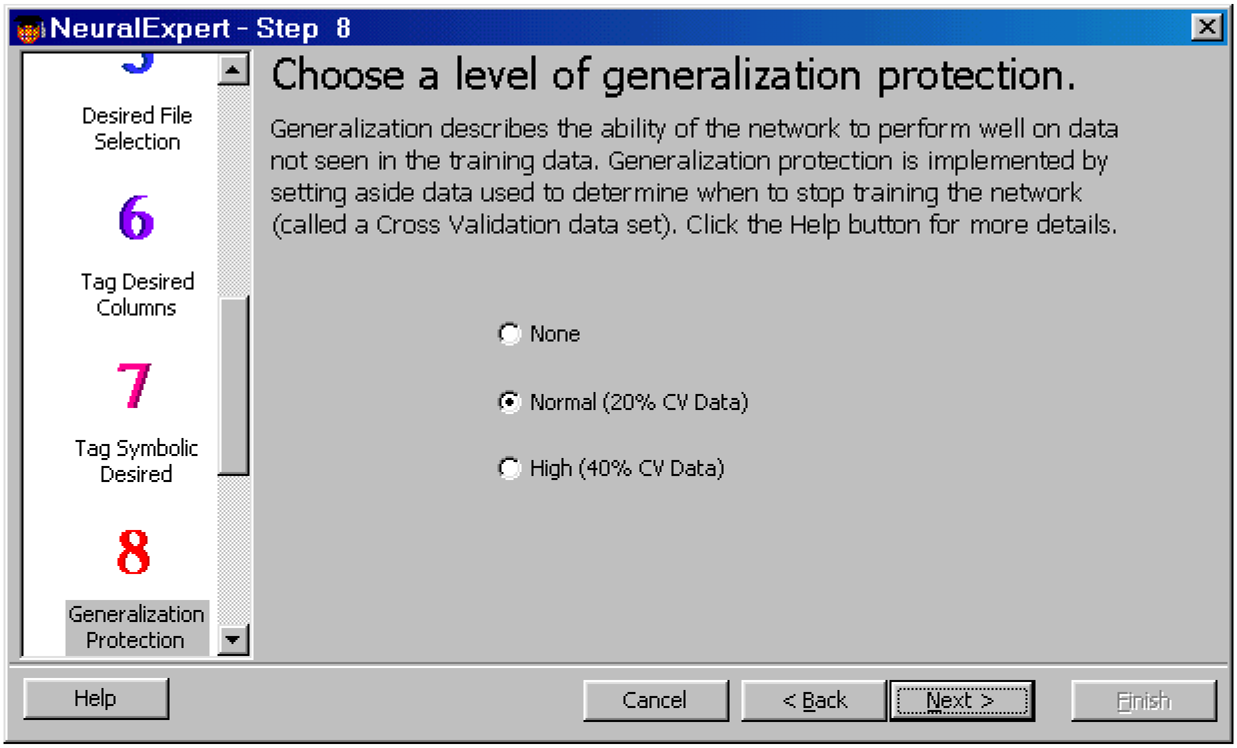
Example

The desired output column contains non-numeric data – instances of "Male" or "Female" specimens. Therefore, this column should be tagged as symbolic.

ü Leave the only item ("Sex") checked.

ü Click the *Next* button Next > to advance to the next panel.

NeuralExpert Generalization Protection Panel



Choose a level of generalization protection.

Generalization describes the ability of the network to perform well on data not seen in the training data. Generalization protection is implemented by setting aside data used to determine when to stop training the network (called a Cross Validation data set). Click the Help button for more details.

○ None

● Normal (20% CV Data)

○ High (40% CV Data)

NeuralExpert Generalization Protection Panel

One of the primary goals in training neural networks is to ensure that the network performs well on data that it has not been trained on (called "generalization"). The standard method of ensuring good generalization is to divide your training data into multiple data sets. The most common data sets are the training, cross validation, and testing data sets.

The cross validation data set is used by the network during training. Periodically, while training on the training data set, the network is tested for performance on the cross validation set. During this testing, the weights are not trained, but the performance of the network on the cross validation set is saved and compared to past values. If the network is starting to overtrain on the training data, the cross validation performance will begin to degrade. Thus, the cross validation data set is used to determine when the network has been trained as well as possible without overtraining (i.e., maximum generalization).

The ***Generalization Protection*** panel is used to specify the amount of data to set aside for cross validation. "None" indicates that all of the data in the input and desired files will be used for training. This option is generally only used when you have very little data to work with (e.g., less than 100 rows). "Normal" generalization protection specifies that 20% of your data will be set aside for cross validation. "High" generalization protection will set aside 40% of your data for cross validation. This option should only be used when you have a great deal of data (e.g. 10,000 rows or more).

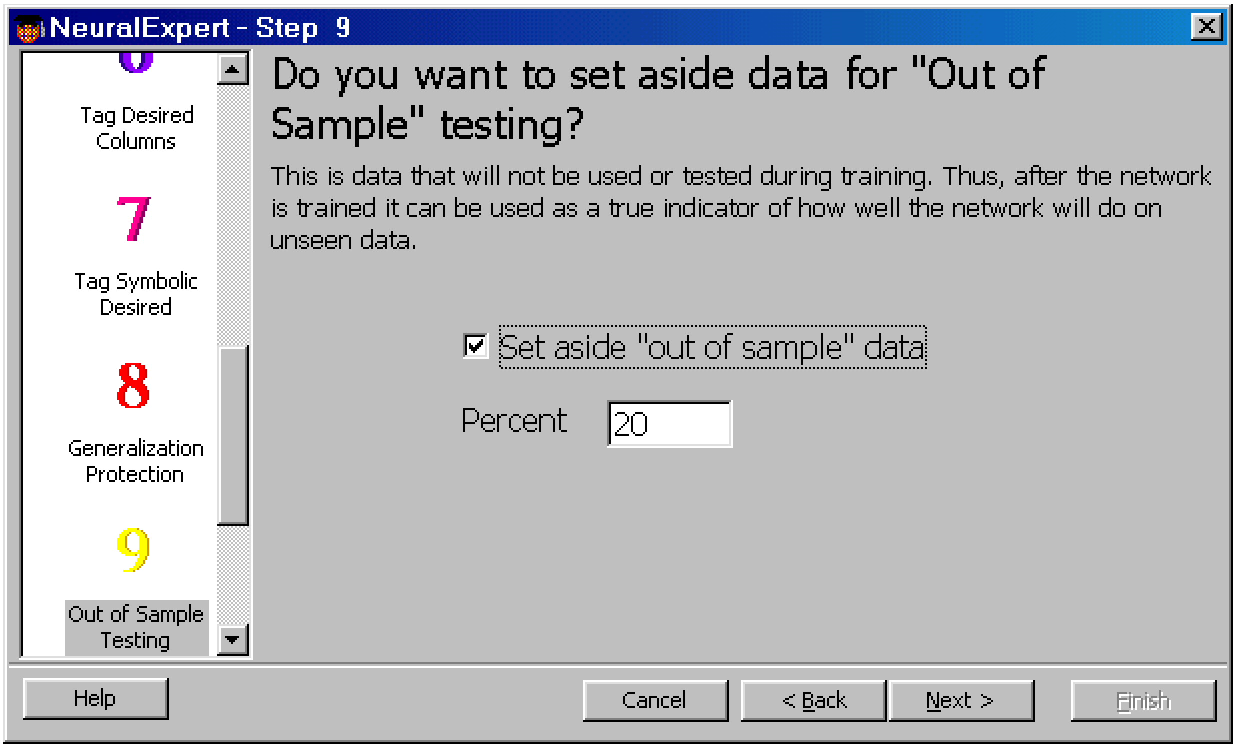
Example

The data file for this example has 200 rows, so we will want to use "Normal" generalization protection.

ü Check the "Normal" radio button.

ü Click the *Next* button Next > to advance to the next panel.

NeuralExpert Out Of Sample Testing Panel


NeuralExpert - Step 9

Do you want to set aside data for "Out of Sample" testing?

This is data that will not be used or tested during training. Thus, after the network is trained it can be used as a true indicator of how well the network will do on unseen data.

☑ Set aside "out of sample" data

Percent 20

NeuralExpert Out Of Sample Testing Panel

Although the network is not trained with the cross validation set, it uses the cross validation set to choose a "best" set of weights. Therefore, it is not truly an out-of-sample test of the network. For a true test of the performance of the network, an independent (i.e., out of sample) testing set is used. This provides a true indication of how the network will perform on new data.

The ***Out Of Sample Testing*** panel is used to specify the amount of data to set aside for the testing set. The percentage will vary depending on the amount of data you have and how rigorously you wish to test the network on out of sample data. The default value is 20% when this option is enabled.

Example

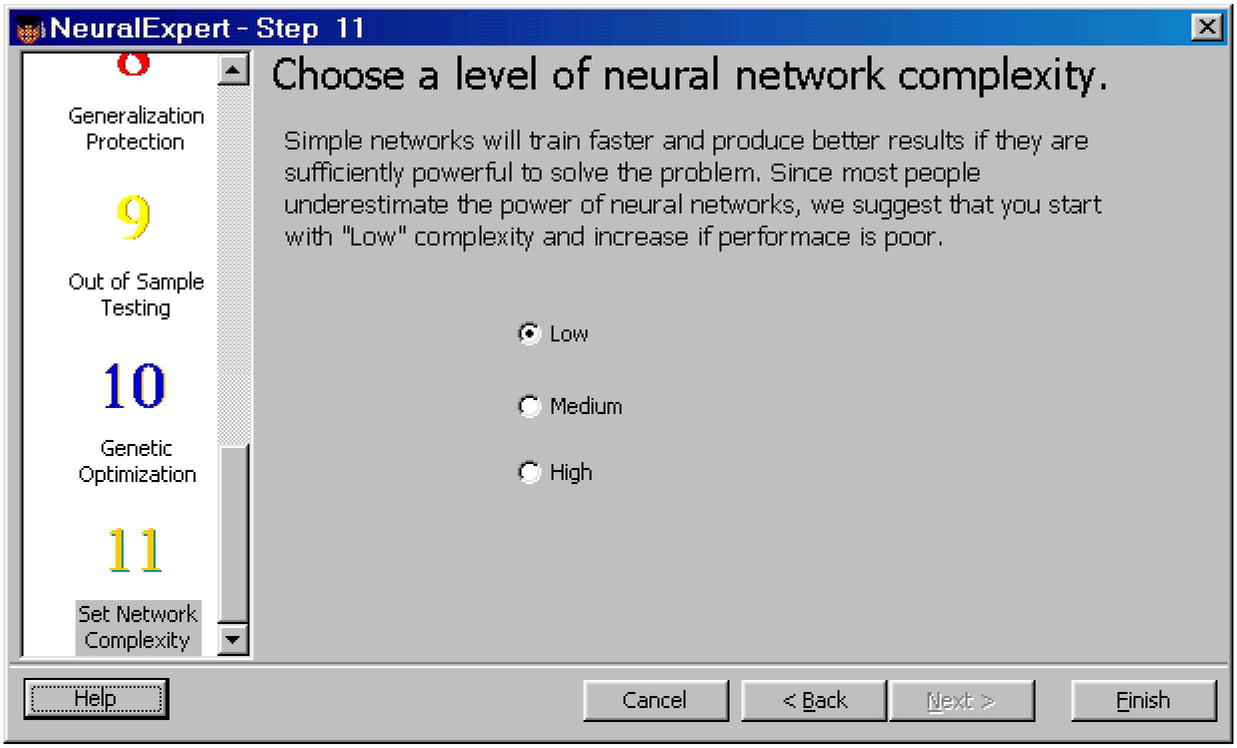
We would like to set aside 40 of the 200 rows of data to test the performance of the network after training.

ü Check the "Set aside out of sample data" check box. Leave the "Percent" text box at the default of 20.

ü Click the *Next* button Next > to advance to the next panel.

ü This problem is not complex enough to warrant the additional processing time needed to perform parameter optimization, so click the *Next* button Next > again to skip the *Genetic Optimization* panel.

NeuralExpert Network Complexity Panel



Choose a level of neural network complexity.

Simple networks will train faster and produce better results if they are sufficiently powerful to solve the problem. Since most people underestimate the power of neural networks, we suggest that you start with "Low" complexity and increase if performace is poor.

● Low

○ Medium

○ High

NeuralExpert Network Complexity Panel

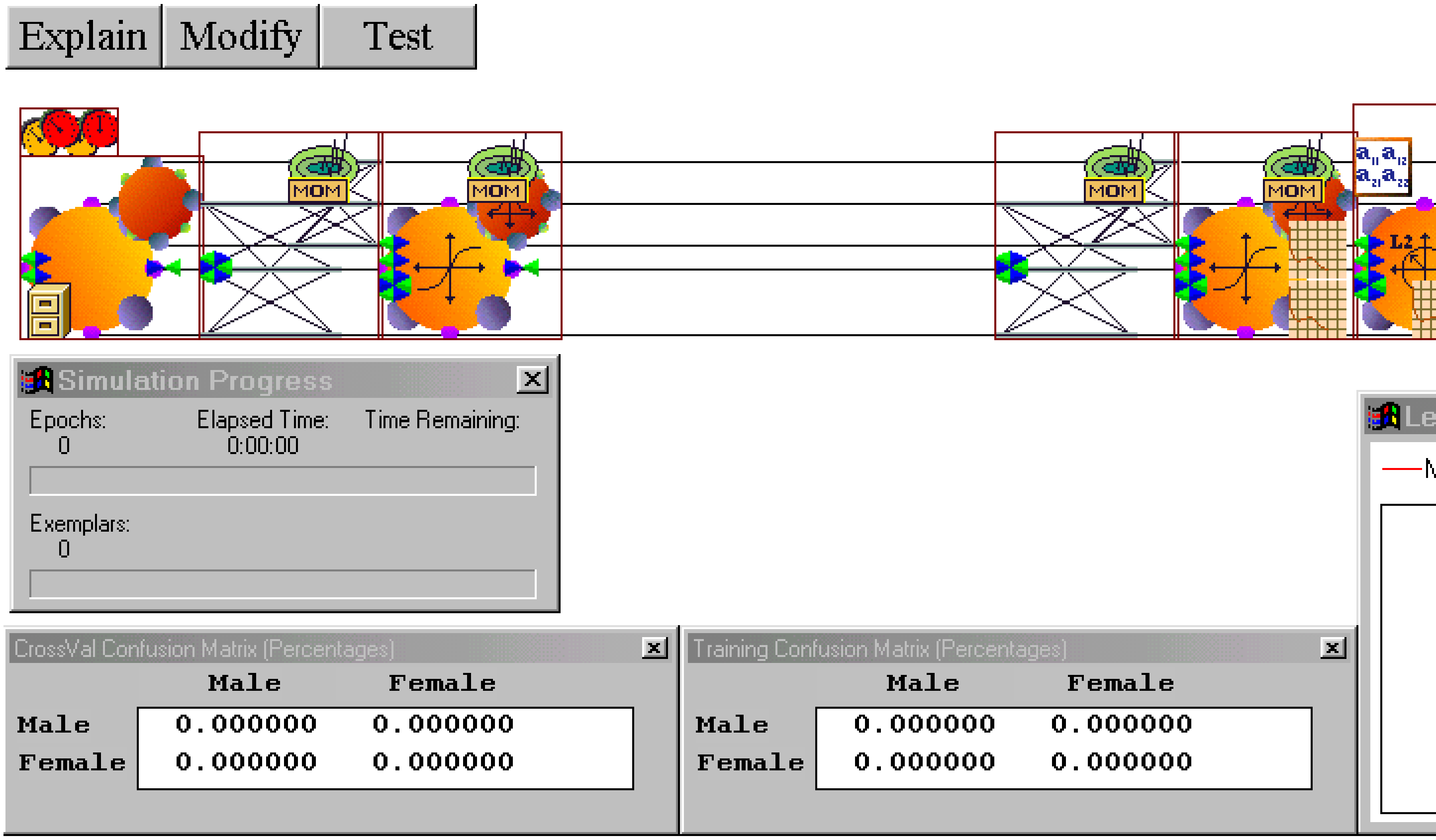
The ***Network Complexity*** panel is used to specify the size of the neural network in terms of hidden layers and processing elements (neurons). In general, smaller neural networks are preferable over large ones. If a small one can solve your problem sufficiently (you would be surprised how powerful small networks are), then a large one will not only require more training and testing time but also may perform worse on new data. This is the generalization problem -- the larger the neural network, the more free parameters it has to solve the problem. Excessive free parameters may over fit the data, causing the network to overspecialize or memorize the training data. When this happens, the performance of the training data will be much better than the performance of the cross validation or testing data sets.

It is strongly recommended that you start with a "low complexity" network. After using a low complexity network, you can move to a "medium" or "high" complexity network and see if the performance is significantly better. Be warned that "medium" or "high" complexity networks generally require a large amount of data (i.e., a thousand or more training rows) to adequately train.

Example

We only have 120 rows of training data, so we should only need a low complexity network.

ü Leave the default setting of "Low" and click the Finish button Finish to build the neural network in NeuroSolutions.



Simulation Progress

Epochs: 0

Elapsed Time: 0:00:00

Time Remaining:

Exemplars: 0

CrossVal Confusion Matrix (Percentages)

	Male	Female
Male	0.000000	0.000000
Female	0.000000	0.000000

Training Confusion Matrix (Percentages)

	Male	Female
Male	0.000000	0.000000
Female	0.000000	0.000000

NeuroSolutions Breadboard built with the NeuralExpert

After the network is built, a warning panel will be displayed indicating that there may not be enough training data to adequately train the neural network. This is because this sample data set has a small number of training exemplars (120) for the number of inputs we have (6). This is not a problem since we are just using this data set for demonstration purposes. If you get this panel when using a real data set and your network does not perform well on the testing set, then you may need to either reduce the number of inputs or increase the amount of training data.

Breadboard Icons

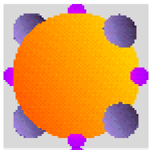
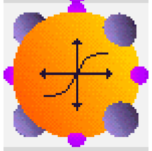
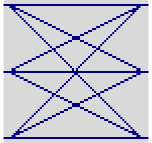
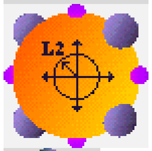
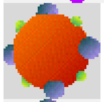
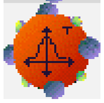
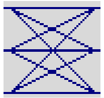
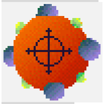
Although you are no doubt anxious to try out your first simulation, a short digression on understanding the NeuroSolutions breadboard is in order.

The Graphical User Interface and Simulations

NeuroSolutions adheres to the so-called local additive model. Under this model, each component can activate and learn using only its own weights and the activations of its neighbors. This lends itself very well to the object-oriented modeling, since each component can be a separate object that sends and receives messages. This in turn allows for a graphical user interface (GUI) with icon-based construction of networks.

When you see a group of components on the breadboard, it is important to understand that it is much more than a picture; it is an actual representation of the underlying neural network behind the user interface. What this means to the user is that there is a one-to-one correspondence between the icons on the breadboard and the simulation that is going on behind the GUI. Any network that you can construct on the breadboard can be simulated. This is the key to the power of NeuroSolutions. In addition to all the standard neural architectures, it is very easy to build and simulate novel architectures that are state of the art in neurocomputing.

Here are some of the most common components:

Icon	Name	Description	Primary Usage
	Axon	Layer of PE's (processing elements) with identity transfer function.	Can act as a placeholder for the File component at the input layer, or as a linear output layer.
	TanhAxon	Layer of PE's with hyperbolic transfer function (output range –1 to 1).	Used as hidden or output layer.
	FullSynapse	Full matrix multiplication.	Connects two axon layers.
	L2Criterion	Square error criterion.	Computes the error between the output and desired signal, and passes it to the backpropagation network.
	BackAxon	Layer of PE's with identity transfer function.	Attaches to "dual" forward Axon, for use in backpropagation network.
	BackTanhAxon	Layer of PE's with transfer function that is the derivative of the TanhAxon.	Attaches to "dual" forward TanhAxon, for use in backpropagation network.
	BackFullSynapse	Back full matrix multiplication.	Attaches to "dual" forward FullSynapse, for use in backpropagation network.
	BackCriteriaControl	Input to backpropagation network.	Attaches to Criterion, for use in backpropagation network. Receives error from Criterion.
	Momentum	Gradient search with momentum.	Updates weights. Momentum increases effective learning rate when weight change is consistently in the same direction.

	StaticControl	Static forward controller	Controls the forward activation phase of network.
	BackStaticControl	Static backpropagation controller	Controls the backward activation phase of network (backpropagation).
	File	File input	For network input and desired data from a file.
	ThresholdTransmitter	Thresholded transmitter	For controlling one component based on the values of another.
	BarChart	Bar chart probe	Displays data bar graph style.
	DataGraph	Graphing probe	Displays data versus time.
	MatrixViewer	Numerical probe	Displays numerical values at the current instant in time.
	DataWriter	Numerical probe	Displays numerical values across time. Also allows for the saving of data to a file.

It is important to note that there may be probes stamped on the breadboard that are not opened by default. The reason for this is that open probe display windows require additional processing during training, which may slow down the simulation considerably. However, you may find some of these probes to be useful, in which case you should open them by double-clicking on the corresponding icon on the breadboard.

Example

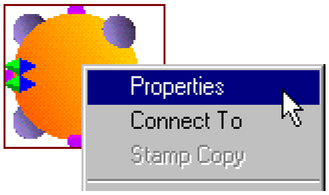
ü Double-click on the top-most DataGraph icon stacked on the right-most TanhAxon (see table above). When the network is run, this probe has been set up to display the network output vs. the desired output for the cross validation set.

The Inspector

Every NeuroSolutions component also has a corresponding parameter set that you can edit. You access a component's parameter set though a dialog box called the Inspector.

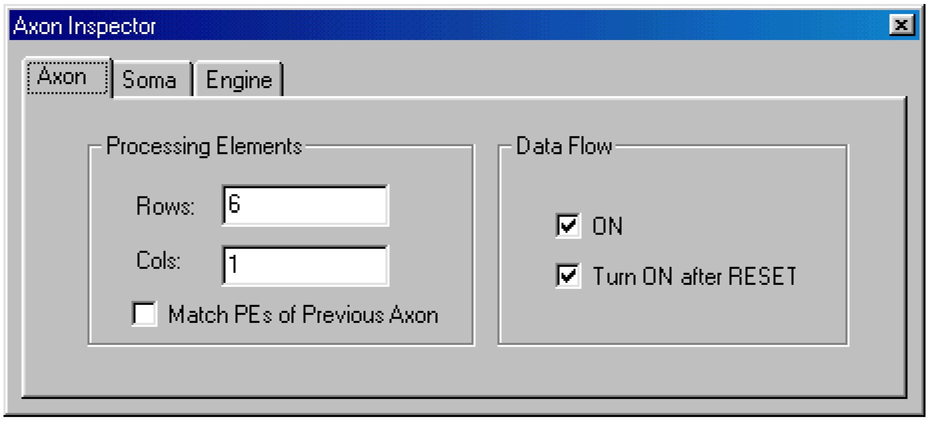
Invoking the Inspector

§ You invoke the Inspector for any component on the breadboard by right-clicking its icon and choosing "Properties":



Invoking the Inspector

§ You can also invoke the Inspector by left-click selecting an icon and then typing "Alt+Enter."



Processing Elements

Rows: 6

Cols: 1

☐ Match PEs of Previous Axon

Data Flow

☑ ON

☑ Turn ON after RESET

Sample Inspector

Property Pages

Once the Inspector is open, selecting any icon on the breadboard will change the Inspector to reflect that component.

A component's parameter set is organized according to property pages. The property pages are labeled at the tabs across the top of the Inspector. You access the various property pages by clicking on the tabs.

Simultaneously Editing Several Components

Thanks to NeuroSolutions' object-oriented design, you will discover strong similarities between the property pages of components within the same family. You can take advantage of this by simultaneously editing the parameters of multiple components:

§ Invoke the Inspector for the first component.

§ Go to the property page of your choice.

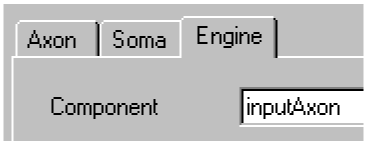
§ Hold down the "Shift" key while left-click selecting the remaining components.

§ Edit any parameter(s) within the Inspector. Any changes you make in the Inspector will be reflected in all selected components, so long as they all have the same parameter.

For example, you can simultaneously change the number of PE's of a group of Axons, even if they have different transfer functions (tanh, sigmoid, etc.).

Component Names

A name is one parameter that all components have in common. A component's name can be found on the **Engine** property page of its Inspector. Every component on a breadboard must have a distinct name. The NeuralBuilder automatically named components in a standard but intuitive fashion. For example, the Axon with a File component is always named "inputAxon":



Component: inputAxon

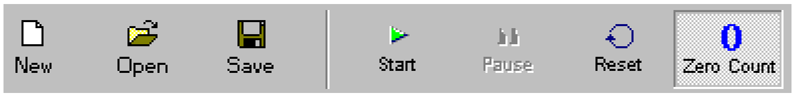
Example Component Name

You can name components any way you choose. However, most NeuroSolutions macros supplied with the package expect that the components be named in a standard way. See the online help topic "Naming" for more information.

Running a Simulation

Default Toolbar

Once you have constructed a network, the next step is to train it. The basic network controls are included in the default toolbar (called the *Necessities* toolbar), which contains the following icons:



A segment of the NeuroSolutions Default Toolbar

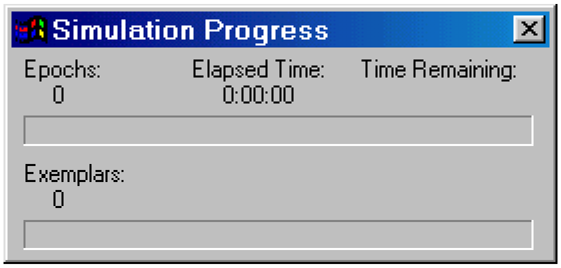
If this toolbar is not visible, you can activate it by going to the *Tools* menu, choosing "Customize", and checking the "Necessities" check box.

The following commands are most commonly used to control a simulation:

Name	**Action**
Start	Starts the simulation.
Pause	Pauses the simulation.
Reset	Resets the epoch and exemplar counters, and randomizes the weights.
Zero Counters	Resets the epoch and exemplar counters without randomizing the weights.

Monitoring a Simulation's Progress

The *Simulation Progress* window provides a real-time graphic view of the number of epochs that have been run. If this window is not visible, double-click the forward controller Next > or ⏱⏱⏱.



Epochs: 0

Elapsed Time: 0:00:00

Time Remaining:

Exemplars: 0

NeuroSolutions Simulation Progress Window

Example

This example picks up where the one from the previous chapter on the NeuralExpert left off. Even if you did not follow the example in the previous chapter, you can still open an existing breadboard that has been pre-saved for you:

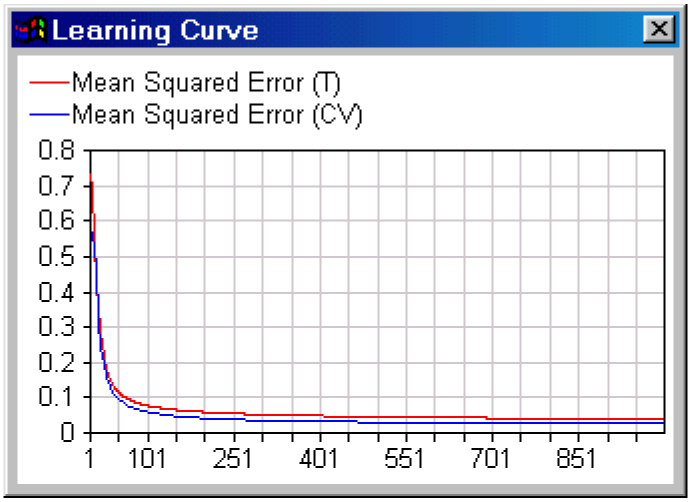
ü Go to the *File* menu and select "Open," or type "Ctrl+O," or click the 📂 toolbar button.

ü Navigate to the file "…\NeuroSolutions\SampleData\CrabMLP.nsb" and double-click it.

You are now ready to train the network.

ü Click the *Start* button ▶ to begin the simulation to train the network. Note: this button may be hidden behind a probe window.

The network will train for 1000 epochs. If the training was successful, the learning curves should look something like this:



Sample Learning Curves

The red line corresponds to the error of the training set and the blue line corresponds the error of the cross validation set. In most cases you will find that the cross validation error will initially fall with the training error, but eventually will rise after the network begins to overtrain or "memorize" the data. The network is configured to automatically save the weights of the network when the cross validation error reaches a bottom.

If your learning curves do not approach zero, then see the following section on Recovering from an Unsuccessful Training for a discussion of possible reasons and options.

It is always a good idea to save a breadboard after a successful training.

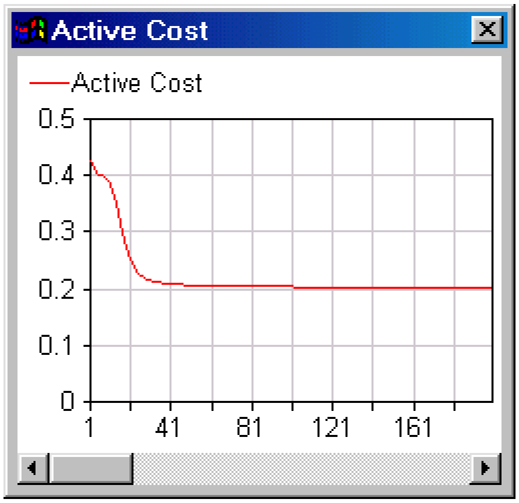
ü Go to the *File* menu and select "Save," or type "Ctrl+S," or click the 💾 icon.

ü If you have not previously saved the breadboard, you will be prompted to save the network with a name and location of your choice.

Recovering from an Unsuccessful Training

Identifying an Unsuccessful Training

It may happen that the network does not learn the problem. This is best evidenced by a learning curve that does not approach zero.



An Unsuccessful Training

Three Possible Reasons for an Unsuccessful Training

If a training is unsuccessful, it is most likely due to one of three factors:

1. The network is capable of learning the problem but has not been trained long enough.
2. The network is capable of learning the problem but is stuck in a local minima.
3. The network is not powerful enough to learn the problem.

Increasing the Network's Computing Power

If you have determined that the problem is not due to one of the first two possibilities, then you will need to increase the number of processing elements in the network. If your network was created with the NeuralExpert, you can increase the network's computing power by following these steps:

§ Click the "Modify" button on the breadboard.

§ Switch to the "Network Complexity" panel of the NeuralExpert.

§ If your network was "Low" change it to "Medium". If it was "Medium" change it to "High".

§ Click the "Finish" button to make the modifications to the network.

Increasing the Training Time

If the network has not been trained long enough, the simple remedy is to increase the number of epochs and continue the training:

§ Invoke the Inspector for the forward controller [Next >].

§ On the **Static** property page, increase the "Epochs/Run."

§ Click the *Start* button ▶ to continue the training.

Getting Out of a Local Minima

NeuroSolutions has several controls useful for getting a network out of a local minima. These controls are available within the *Control* toolbar. If this toolbar is not visible, you can activate it by going to the *Tools* menu, choosing "Customize", and checking the "Control" check box.

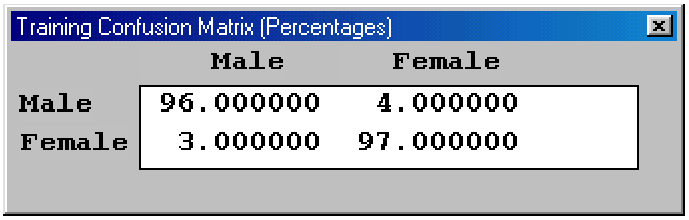
Control	Name	Action

Reset	Resets the epoch and exemplar counters, and randomizes the weights.
Randomize	Randomizes the weights using Mean and Variance, specified on **Soma** property page.
Jog	Randomizes the weights about their present values using the Variance, specified on **Soma** property page.

If you are stuck in a local minima you may want to first try jogging the weights and continuing the training. If that doesn't work you may want to reset or randomize the network and train again.

Verifying the Training

Although the mean square error is a good overall measure of whether a training run was successful, sometimes it can be misleading. This is particularly true for classification problems. When "Classification" is selected as the problem type, the NeuralExpert stamps a pair of confusion matrix probes – one for the training set and one for the cross validation set (assuming that you selected either "Normal" or "High" *Generalization Protection*).



Training Confusion Matrix (Percentages)

	Male	Female
Male	96.000000	4.000000
Female	3.000000	97.000000

A Confusion Matrix Probe for the Training Set

The confusion matrix tallies the results of all exemplars of the last epoch and computes the classification percentages for every output vs. desired combination. For example, in the figure above, 96% of the male exemplars were correctly classified while 4% of the male exemplars were classified incorrectly as female. Similarly, 97% of the female exemplars were correctly classified while 3% of the female exemplars were classified as male.

Example

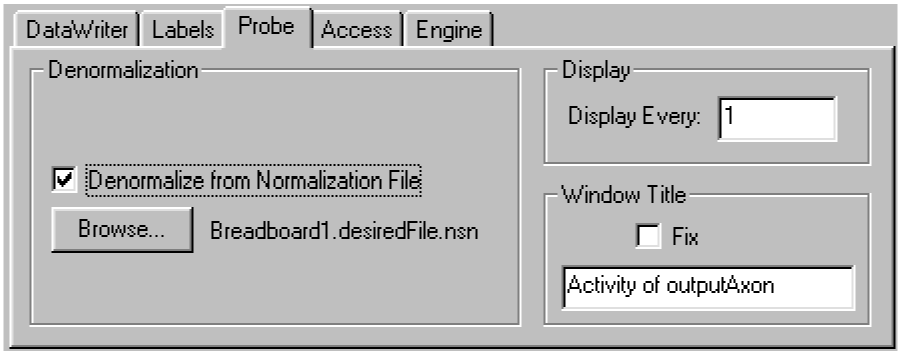
ü Observe the "CrossVal" and "Training" confusion matrix probes displayed on the breadboard to determine how well the training performed.

Denormalizing the Probes

The NeuralBuilder and NeuralExpert automatically set up normalization of the input and desired files. Normalization, the process of scaling and shifting the data to better match the network's range, is an important part of neural network pre-processing, and can significantly speed up training.

Viewing the Data in its Native Range

When probing a network, however, it may be preferable to view the data in its original range. This process is called denormalization. All probes that display the data numerically are capable of denormalization, including the MatrixViewer Next >, MatrixEditor, and DataWriter Next >. Access to this feature is from the **Probes** property page of the Inspector:



DataWriter | Labels | Probe | Access | Engine

Denormalization

☑ Denormalize from Normalization File

Browse... Breadboard1.desiredFile.nsn

Display

Display Every: 1

Window Title

□ Fix

Activity of outputAxon

Probe Denormalization in the Inspector

Automatic Matching of Probe with Normalization File

Every data set generates its own normalization file, which stores the scale and offset of the data. Probes can then use this normalization file to perform the denormalization. If the probe is at the input File, NeuroSolutions will automatically choose the input File's normalization file to perform the denormalization. If the Probe is at either the output Axon or desired File, the desired File's normalization file will be used.

Note that denormalizing the probes does not in any way affect the training of the network.

Example

By default the NeuralBuilder and NeuralExpert configure the probes on the breadboard to display in the original data's range. Let's verify that this setting is correct.

ü Invoke the Inspector for the bottom MatrixViewer Next > on the desired File, and go to the **Probes** property page.

ü Verify that the box "Denormalize from Normalization File" is checked

ü Repeat for the other desired MatrixViewer probe and the DataGraph probes Next > attached to the right-most TanhAxon.

Manually Optimizing the Network

As mentioned previously, it is important to find the network with the minimal number of free weights that can still learn the problem. The minimal network is more likely to generalize well to new data. Therefore, once you have achieved a successful training, you should then begin the process of decreasing the size of the network and repeating the training until it no longer learns the problem to your satisfaction.

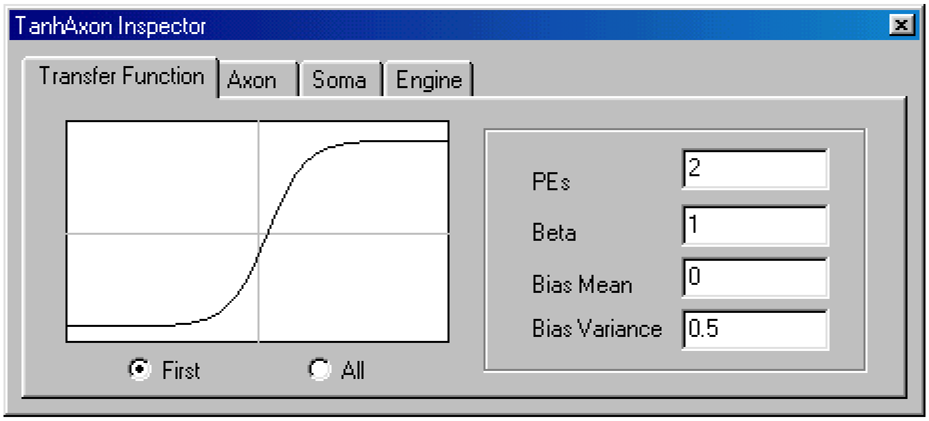
Note that NeuroSolutions does include a facility for optimizing various network parameters using a genetic algorithm, but this topic is too advanced for this manual. Interested users should refer to the NeuroSolutions, NeuralBuilder and/or NeuralExpert documentation for detailed information on genetic optimization within NeuroSolutions.

Example

The NeuralExpert built the neural network with 3 PE's on the hidden layer TanhAxon, and the network learned the problem easily. Now we will decrease that number:

ü Invoke the Inspector for the hidden layer TanhAxon.

ü Enter "2" in the edit box for the number of **Rows**.



PEs 2

Beta 1

Bias Mean 0

Bias Variance 0.5

TanhAxon Inspector After Changing the Number of PE's to 2

Now we will re-train the network:

ü Click the *Reset* button ↺, followed by the *Start* button ▶.

ü Observe the learning curve and confusion matrices to see whether the network learned the problem.

If the network did not learn, re-train several more times to make sure the network was not stuck in a local minima. You should find that this size network can learn the problem. Once the network learns the problem, save the breadboard:

ü Go to the *File* menu and select "Save," or type "Ctrl+S," or click the Next > icon.

Let's now decrease the number of **Rows** to 1:

ü Enter "1" in the edit box for the number of PE's.

ü Repeat the training process as previously described.

You may observe that the network does not solve the problem as well with 1 PE. If this is the case, you can conclude that 2 PE's on the hidden layer is optimal. To load in the breadboard trained with 2 PE's:

ü Go to the *File* menu and choose "Close."

ü When prompted, select "No" No to saving the breadboard.

ü Go to the *File* menu and choose the most recent file name you used from the list at the bottom of the menu.

Starting the TestingWizard

After training a network, you will want to test the network performance on data that the network was not trained with. The TestingWizard automates this procedure by providing an easy way to produce the network output for the testing dataset that you defined within the NeuralExpert or NeuralBuilder, or on a new dataset not yet defined.

To launch the TestingWizard from within NeuroSolutions:

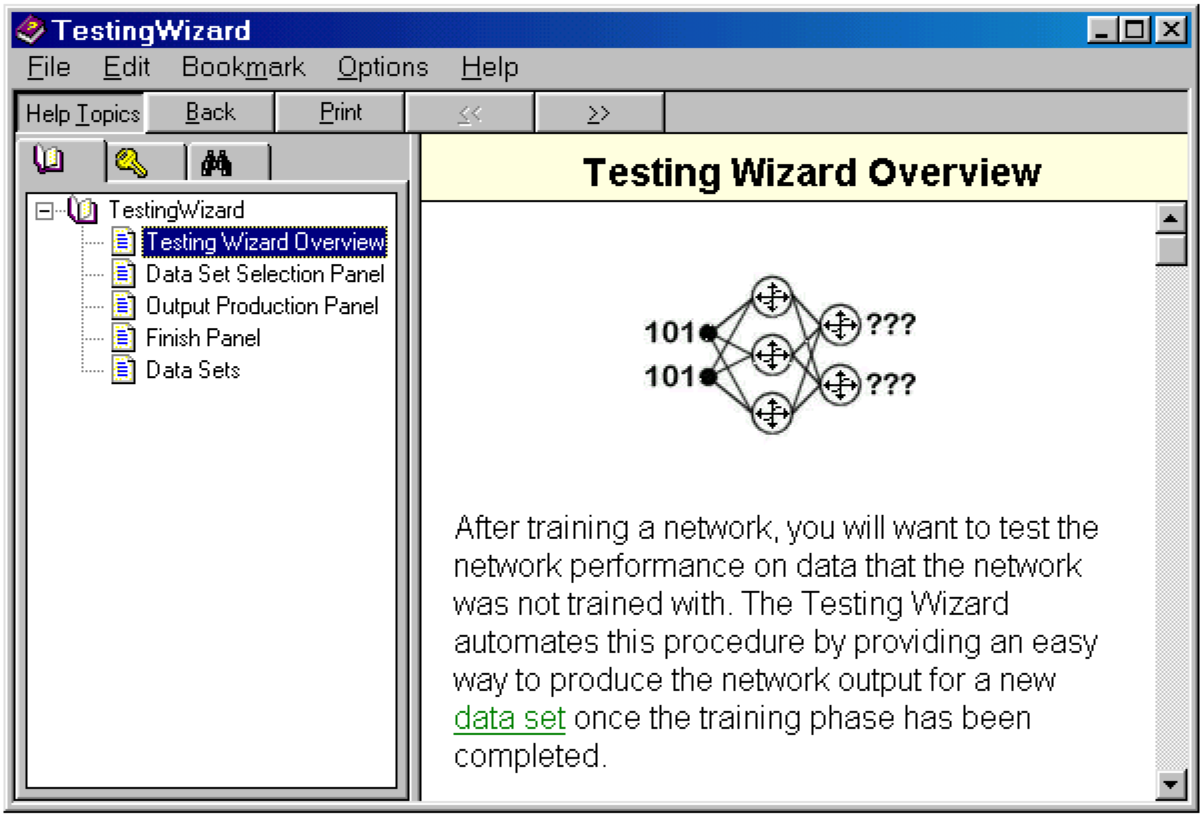
- Go to the *Tools* menu and choose "TestingWizard" or click the "Testing" toolbar button .

If your breadboard was built with the NeuralExpert, you may alternatively:

- Click the "Test" button in the upper-left corner of the breadboard.

Getting Help in the TestingWizard

Online help is available from all TestingWizard panels. To access help, click the *Help* button in the lower left corner of the wizard.

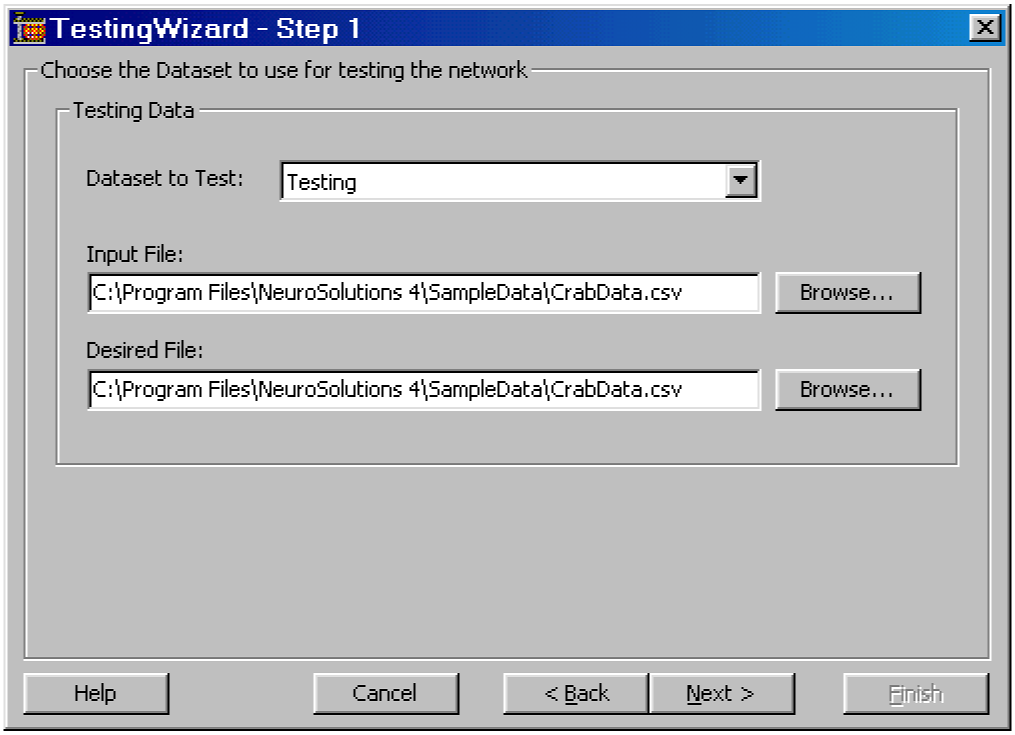


Testing Wizard Overview

After training a network, you will want to test the network performance on data that the network was not trained with. The Testing Wizard automates this procedure by providing an easy way to produce the network output for a new data set once the training phase has been completed.

TestingWizard Help Window

TestingWizard Data Set Selection Panel


TestingWizard - Step 1

Choose the Dataset to use for testing the network

Testing Data

Dataset to Test: Testing

Input File: C:\Program Files\NeuroSolutions 4\SampleData\CrabData.csv

Desired File: C:\Program Files\NeuroSolutions 4\SampleData\CrabData.csv

TestingWizard Data Set Selection Panel

This panel is used to specify the data to use for your test. The two most common data sets to use are Testing and Production. Testing is used if there is desired data that corresponds to the input data. Production is used if you do not know what the output is supposed to be – you want to neural network to tell you. You may also test the network using the Training or Cross Validation sets.

If the selected data set has already been defined, either manually or using one of the wizards, then the file path(s) will be filled in for you by default. If you have not yet defined the data set, or would like to override the existing file settings, click the Browse button(s) to define the input and/or desired files.

Note that the input file for the Testing/Production set must have the same column labels as the input file used for the training set. Likewise, the desired file of the Testing set must be of the same structure as the training desired file.

Example

Recall that within the NeuralExpert we had allocated 20% of the data file for the testing set. The TestingWizard detects this automatically, so there is no need to specify the files again.

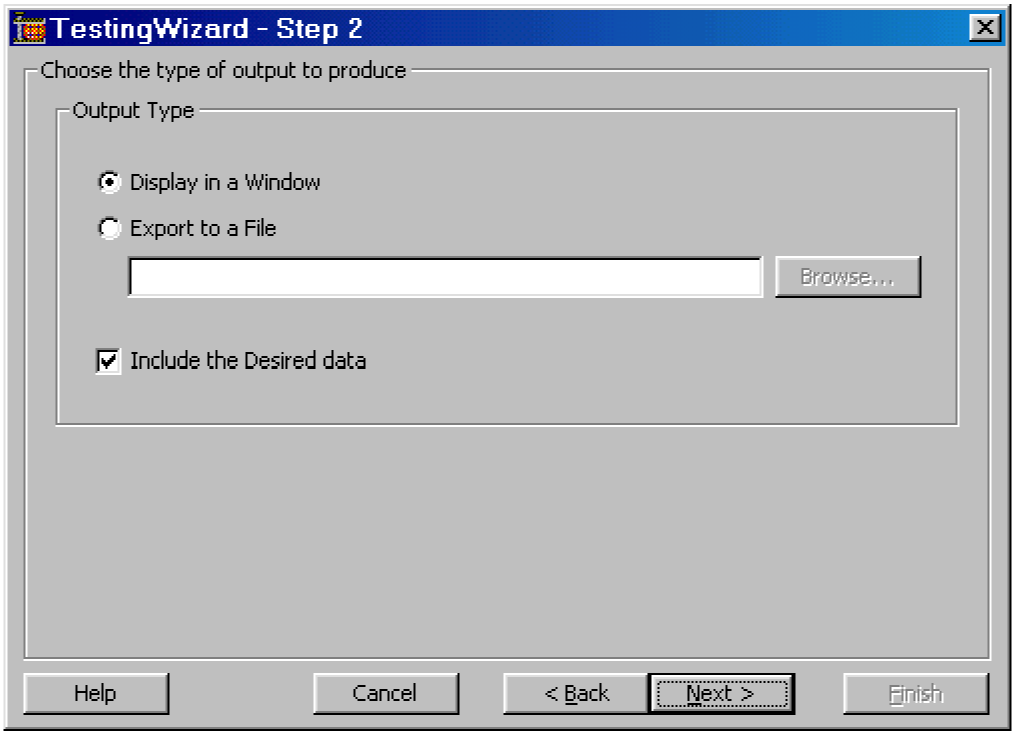
ü Launch the TestingWizard.

ü Read the Introduction panel and click the *Next* button Next > to advance to the file selection panel.

ü Note that the data set defaults to Testing and the input and desired file paths are already filled in.

ü Click the *Next* button Next > to advance to the next panel.

TestingWizard Output Production Panel



TestingWizard - Step 2

Choose the type of output to produce

Output Type

● Display in a Window

○ Export to a File

☑ Include the Desired data

TestingWizard Output Production Panel

This panel is used to specify how the network output of the testing set should be produced. The DataWriter probe is the component used to extract the output data. You may either display the output data in a window or have the data written to an ASCII file. To do this, click the "Export to a file" radio button, click the "Browse" button and specify a file name and directory to write the output data to.

You may also include the desired data within the display/file in order to do a side-by-side comparison. If you only want to produce the output data, uncheck the "Include the Desired Data" box.

Example

We will keep the default settings, which will produce the testing results within a display window and include the desired response with the network output.

ü Make sure that the "Display in a Window" radio button is selected and the "Include the Desired Data" checkbox is checked.

ü Click the *Next* button Next > to advance to the next panel.

ü Click the *Finish* button to test the network.

ü Observe the results by scrolling through the window labeled "Desired and Output".

Starting the NeuralBuilder

An alternative to the NeuralExpert is the NeuralBuilder. This neural network construction utility provides more flexibility in specifying the neural network topology and parameters.

In the following topics, we will take you through the various design panels of the NeuralBuilder. At each panel, the basic design options will be explained, and suggestions offered. If you would like to understand the rationale behind some of the suggestions, please see the section on Neural Network Training Hints.

We encourage you try the examples that are presented at the end of each panel's description.   Each example continues throughout this chapter.   For this reason, we recommend you do not skip any panels.

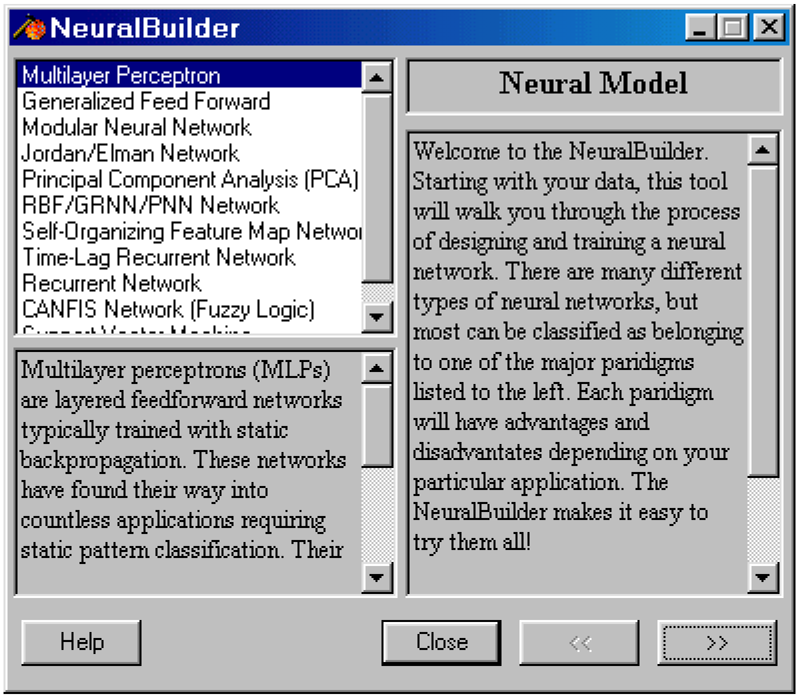
To launch the NeuralBuilder from the Windows *Start* Menu [Next >]:

§ Select *Start*⇒*NeuroSolutions 4*⇒*NeuralBuilder*.

To launch the NeuralBuilder from within NeuroSolutions:

§ Go to the *Tools* menu and choose "NeuralBuilder" or click the "NBuilder" toolbar button .

The following panel will appear:



Multilayer Perceptron
Generalized Feed Forward
Modular Neural Network
Jordan/Elman Network
Principal Component Analysis (PCA)
RBF/GRNN/PNN Network
Self-Organizing Feature Map Networ
Time-Lag Recurrent Network
Recurrent Network
CANFIS Network (Fuzzy Logic)

Multilayer perceptrons (MLPs) are layered feedforward networks typically trained with static backpropagation. These networks have found their way into countless applications requiring static pattern classification. Their

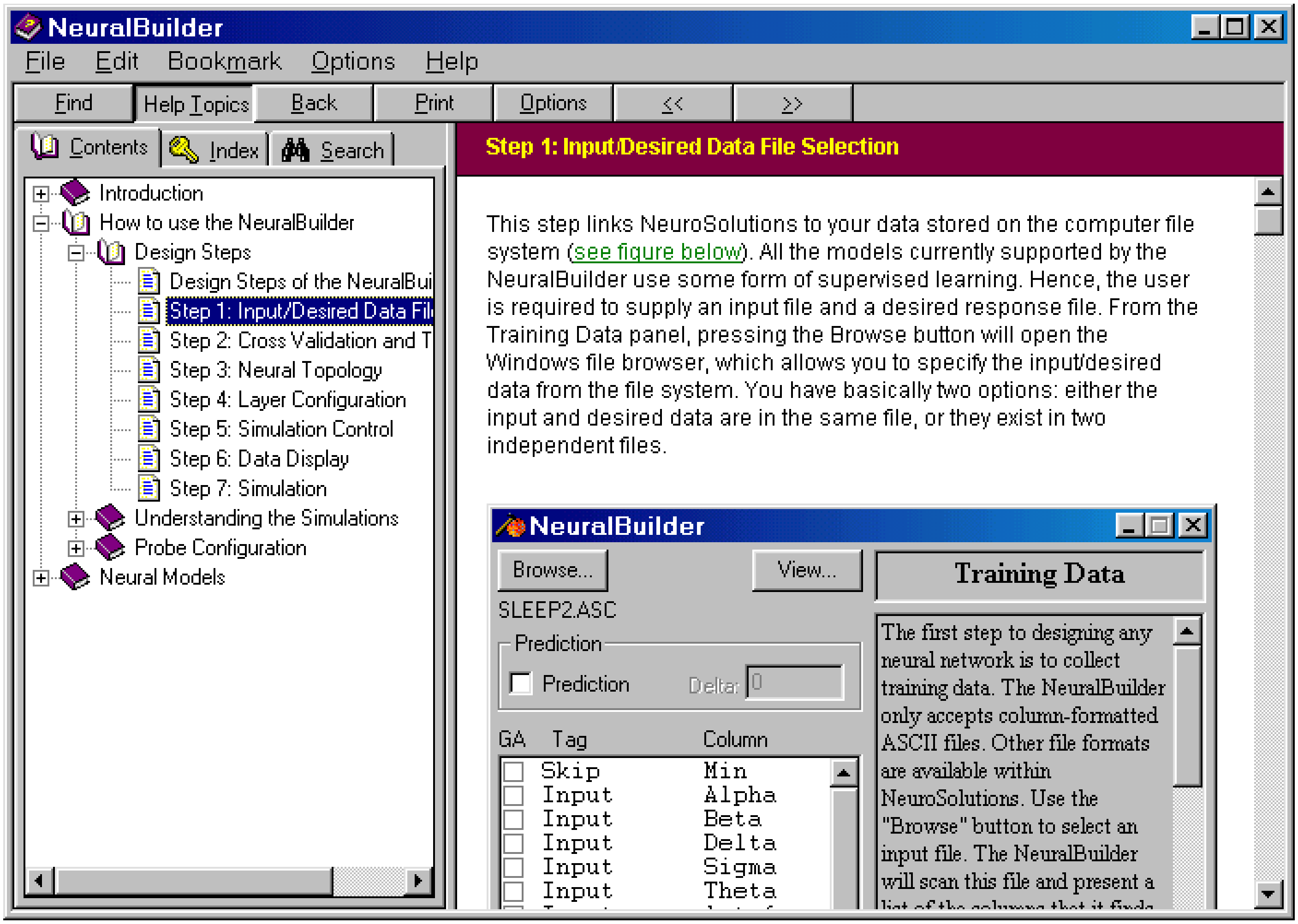
Neural Model

Welcome to the NeuralBuilder. Starting with your data, this tool will walk you through the process of designing and training a neural network. There are many different types of neural networks, but most can be classified as belonging to one of the major paridigms listed to the left. Each paridigm will have advantages and disadvantates depending on your particular application. The NeuralBuilder makes it easy to try them all!

NeuralBuilder Neural Model Panel

Getting Help in the NeuralBuilder

Online help is available from all NeuralBuilder panels. To access help for the current panel, click the *Help* button in the lower left corner of the wizard. Note that all other topics of the help file can be accessed through the table of contents on the left side of the help window.


NeuralBuilder
File Edit Bookmark Options Help
Find
Help Topics
Back
Print
Options
<<
>>
Contents
Index
Search
Introduction
How to use the NeuralBuilder
Design Steps
Design Steps of the NeuralBui
Step 1: Input/Desired Data Fil
Step 2: Cross Validation and T
Step 3: Neural Topology
Step 4: Layer Configuration
Step 5: Simulation Control
Step 6: Data Display
Step 7: Simulation
Understanding the Simulations
Probe Configuration
Neural Models

Step 1: Input/Desired Data File Selection

This step links NeuroSolutions to your data stored on the computer file system (see figure below). All the models currently supported by the NeuralBuilder use some form of supervised learning. Hence, the user is required to supply an input file and a desired response file. From the Training Data panel, pressing the Browse button will open the Windows file browser, which allows you to specify the input/desired data from the file system. You have basically two options: either the input and desired data are in the same file, or they exist in two independent files.

SLEEP2.ASC

□ Prediction Delta: 0

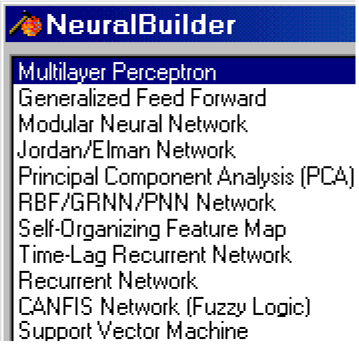
GA	Tag	Column
□	Skip	Min
□	Input	Alpha
□	Input	Beta
□	Input	Delta
□	Input	Sigma
□	Input	Theta

The first step to designing any neural network is to collect training data. The NeuralBuilder only accepts column-formatted ASCII files. Other file formats are available within NeuroSolutions. Use the "Browse" button to select an input file. The NeuralBuilder will scan this file and present a

NeuralBuilder Help Window

NeuralBuilder Supported Models

The first step in building a neural network is choosing a neural model. The NeuralBuilder supports eleven neural models, shown below. For a complete description of these models, and how to choose one for your application, see Choosing a Neural Architecture.



NeuralBuilder

Multilayer Perceptron
Generalized Feed Forward
Modular Neural Network
Jordan/Elman Network
Principal Component Analysis (PCA)
RBF/GRNN/PNN Network
Self-Organizing Feature Map
Time-Lag Recurrent Network
Recurrent Network
CANFIS Network (Fuzzy Logic)
Support Vector Machine

NeuralBuilder Supported Models

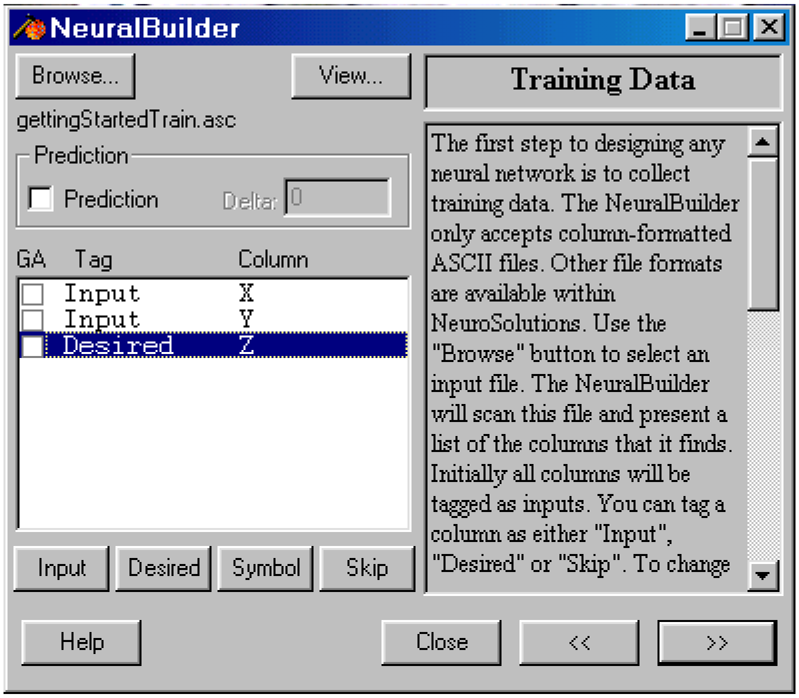
The Multilayer Perceptron is by far the most common neural network model. We will use it as the basis of our example as we take you through the various NeuralBuilder panels. Now, to get started …

Example

ü Use the default "Multilayer Perceptron" in the NeuralBuilder ***Neural Model*** panel.

ü Click the *Forward* button >> to advance to the next panel.

NeuralBuilder Training Data Panel



gettingStartedTrain.asc

☐ Prediction Delta: 0

GA	Tag	Column
☐	Input	X
☐	Input	Y
☐	Desired	Z

The first step to designing any neural network is to collect training data. The NeuralBuilder only accepts column-formatted ASCII files. Other file formats are available within NeuroSolutions. Use the "Browse" button to select an input file. The NeuralBuilder will scan this file and present a list of the columns that it finds. Initially all columns will be tagged as inputs. You can tag a column as either "Input", "Desired" or "Skip". To change

NeuralBuilder Training Data Panel

The next step in constructing your neural model is to select the training data. The ***Training Data*** panel is where you:

§ Choose your file.

§ Tag the columns according to usage.

§ For time series only, specify that the goal of the training is prediction, where "Delta" is the number of time steps in the future to predict.

Please see the section on File Format Requirements before making the file selection.

Tagging Columns

By default, all columns are tagged as "Input." You can change the default setting of any column to the "Desired" response of the network, specify that the column contains "Symbols" that need to be converted to numeric data, or "Skip" the column entirely.

The "GA" checkboxes are used to indicate that a genetic algorithm is to determine if the corresponding input is to be included or skipped. Genetic Algorithms are general-purpose search algorithms based upon the principles of evolution observed in nature. Genetic algorithms combine selection, crossover, and mutation operators with the goal of finding the best solution to a problem. They search for this optimal solution until a specified termination criterion is met. In NeuroSolutions the criteria used to evaluate the fitness of each potential solution is the lowest cost achieved during the training run. Please see the documentation for NeuroSolutions and the NeuralBuilder for more information on this topic.

Training and Desired Files

The input and desired data can be either in the same file, or split into separate files. If they are in separate files, and you do not tag any columns as "Input" in this panel, then the ***Desired Response*** panel will immediately follow this panel, where you can specify the separate desired file.

Example

Our example contains both input and desired data in the same file:

ü Click the *Browse* button Next >. The ***Open*** panel will display.

ü Navigate to the file "…\NeuroSolutions 4\SampleData\gettingStartedTrain.asc" and double-click it.

Back in the ***Training Data*** panel, you should see a list of all the columns in the file. Note that they are all initially tagged as Input.

ü Click the *View* button View... to view the file's contents:

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

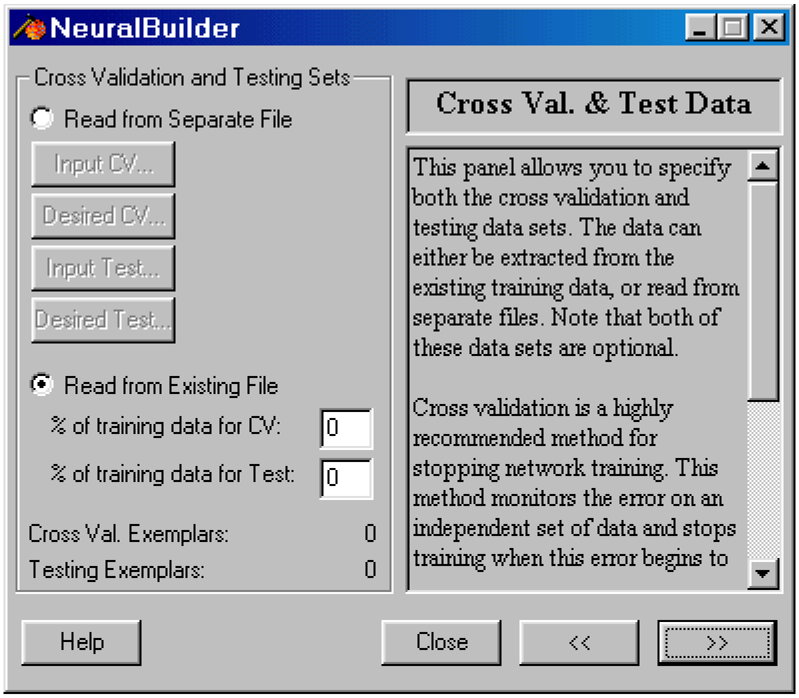
This data is the input-output response of an exclusive-or logic gate. The X and Y columns are the voltage inputs to the gate, and the Z column is the expected output of the gate, given the inputs. There are a total of four exemplars. To tag column Z as the desired data:

ü Select column "Z."

ü Click the *Desired* button Desired. You should see the "Z" column tag change from "Input" to "Desired":

ü Click the *Forward* button Next > to advance to the next panel.

Cross Validation and Test Data Panel



NeuralBuilder

Cross Validation and Testing Sets

○ Read from Separate File

Input CV... | Desired CV... | Input Test... | Desired Test...

● Read from Existing File

% of training data for CV: 0

% of training data for Test: 0

Cross Val. Exemplars: 0

Testing Exemplars: 0

Cross Val. & Test Data

This panel allows you to specify both the cross validation and testing data sets. The data can either be extracted from the existing training data, or read from separate files. Note that both of these data sets are optional.

Cross validation is a highly recommended method for stopping network training. This method monitors the error on an independent set of data and stops training when this error begins to

NeuralBuilder Cross Validation and Test Data Panel

This panel is used to specify the cross validation and/or testing data sets. These data sets can be specified as a percentage of the exemplars from the training file, or they can be input from separate files.

Cross Validation Set

Neural networks can be overtrained to the point where performance on new data actually deteriorates. Roughly speaking, overtraining results in a network that memorizes the individual exemplars, rather than trends in the data set as a whole. Cross validation is a process whereby part of the data set is set aside for the purpose of monitoring the training process, to guard against overtraining.

Testing Set

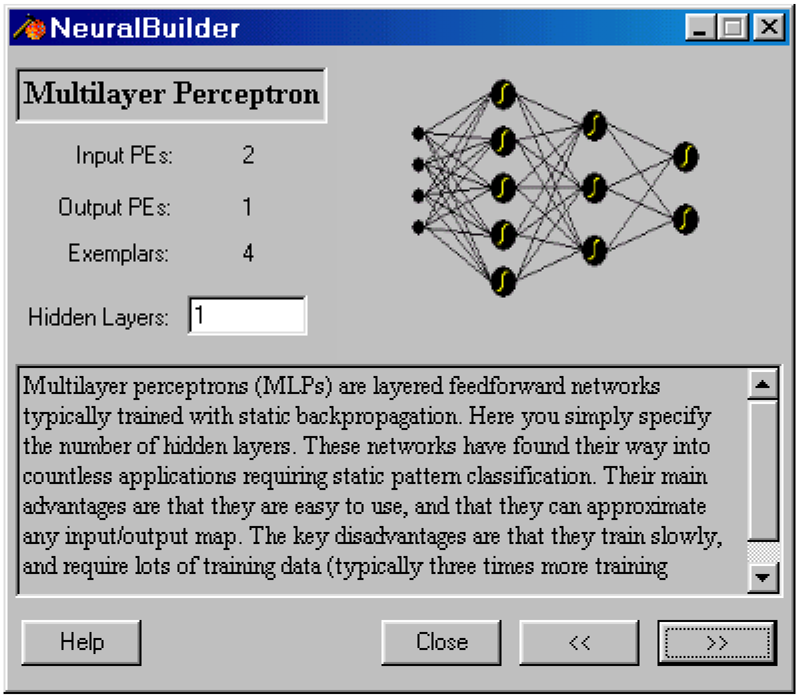
The testing set is used to test the performance of the network. Once the network has been trained, the weights are then frozen, the testing set is fed into the network, and the network output is compared with the desired output.

Example

Because we are working with a small data set, we will not specify a cross validation or testing set.

ü Click the *Forward* button Next > to advance to the next panel.

NeuralBuilder Topology Panel



Multilayer Perceptron

Input PEs: 2

Output PEs: 1

Exemplars: 4

Hidden Layers: 1

Multilayer perceptrons (MLPs) are layered feedforward networks typically trained with static backpropagation. Here you simply specify the number of hidden layers. These networks have found their way into countless applications requiring static pattern classification. Their main advantages are that they are easy to use, and that they can approximate any input/output map. The key disadvantages are that they train slowly, and require lots of training data (typically three times more training

NeuralBuilder Multilayer Perceptron Panel

The Topology panel is specific to the neural model chosen in the first panel. This is where you set:

§ The number of hidden layers, and

§ Parameters specific to the architecture you chose in the first panel.

Number of Hidden Layers

Common to all models is an text box for the number of hidden layers. Unless you believe your problem is particularly difficult, start with the default setting of one hidden layer; you can always return later and add additional layers.

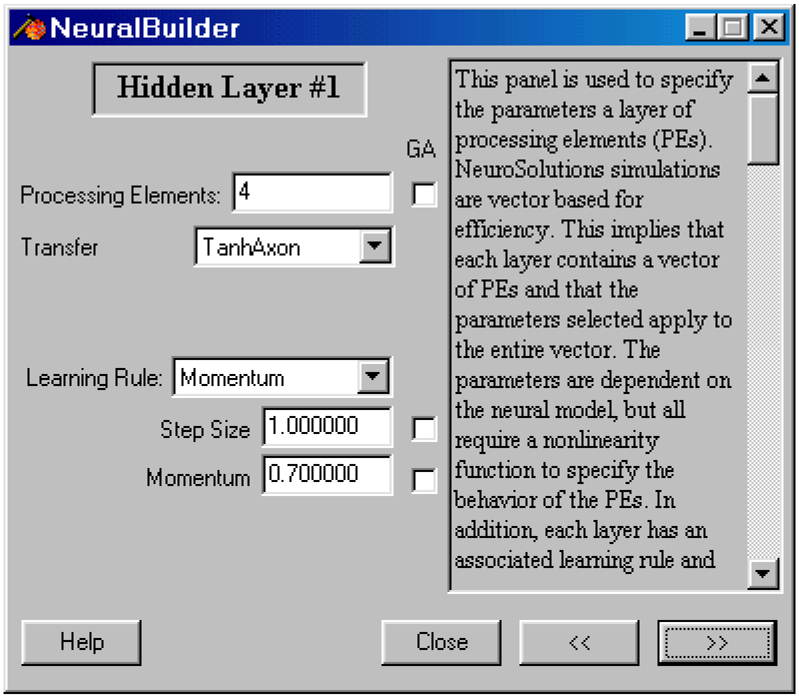
Example

For the Multilayer Perceptron, the number of hidden layers is the only parameter in this panel. One hidden layer should be sufficient to solve the exclusive-or problem.

ü Use the default value of "1" as the number of **Hidden Layers**.

ü Click the *Forward* button Next > to advance to the next panel.

NeuralBuilder Hidden Layer Panel



Processing Elements: 4 □

Transfer TanhAxon

Learning Rule: Momentum

Step Size 1.000000 □

Momentum 0.700000 □

This panel is used to specify the parameters a layer of processing elements (PEs). NeuroSolutions simulations are vector based for efficiency. This implies that each layer contains a vector of PEs and that the parameters selected apply to the entire vector. The parameters are dependent on the neural model, but all require a nonlinearity function to specify the behavior of the PEs. In addition, each layer has an associated learning rule and

NeuralBuilder Hidden Layer #1 Panel

The ***Hidden Layer*** panel is where you choose:

- The number of processing elements in the layer.
- The layer's transfer function.
- The layer's learning rule and parameters.

For each number of hidden layers requested in the Topology panel, separate ***Hidden Layer*** panels will appear in sequence.

Number of Processing Elements

The most important parameter you need to set here is the number of processing elements (PE's). The number of PE's directly affects the overall computing power of the network. Ideally, the number of PE's should be chosen based on the complexity of the desired input-output mapping of the data, but really can only be determined experimentally.

It is important to note that the minimal number of PE's needed to solve a problem need not be related to the size of the data set. However, the NeuralBuilder does not know the complexity of the data, and thus chooses the number of PE's proportional to the number of input channels. This is likely to overestimate the required number of PE's, but at least the network is likely to learn on the first trial. However, good generalization to new data depends on finding the minimal number of PE's that can solve the problem.

The "GA" checkbox next to the "Processing Elements" text box is used to indicate that a genetic algorithm will be used to determine the number of processing elements that produced the lowest cross validation error during a large number of training runs. Genetic optimization is beyond the scope of this manual. Please see the documentation for NeuroSolutions and the NeuralBuilder for more information on this topic.

Transfer Function

It is the non-linearity of the hidden layer's axons that give a neural network the computational ability to learn difficult problems. Of the eight axons offered by the NeuralBuilder,

Axon	Output Range	Features
TanhAxon	-1 to 1	Nonlinear axon of choice.

SigmoidAxon	0 to 1	Same general shape as TanhAxon.
LinearTanhAxon	-1 to 1	Piecewise linear approximation to TanhAxon.
LinearSigmoidAxon	0 to 1	Piecewise linear approximation to SigmoidAxon.
SoftMaxAxon	0 to 1	Outputs sum to 1. Used for classification.
BiasAxon	Infinite	Linear axon with adjustable slope and adaptable bias.
LinearAxon	Infinite	Linear axon with adaptable bias.
Axon	Infinite	Simplest axon, identity transfer function.

only the first five are non-linear. Of those five, only the TanhAxon and SigmoidAxon are commonly used on the hidden layer(s).

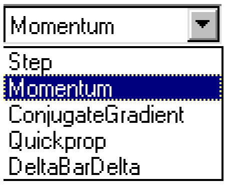
For most problems, use the recommended TanhAxon transfer function.

Input File Normalization

The NeuralBuilder will automatically instruct NeuroSolutions to scale and shift the input data to match the range of the first hidden layer's transfer function. For example, if the first hidden layer's axon is a TanhAxon, the input data will be scaled and shifted to lie between –1 and 1. This important pre-processing step is called normalization, which will be discussed later.

Learning Rules

The learning rule, also called gradient search, is used to calculate the weight update. NeuroSolutions and the NeuralBuilder offer five learning rules:



NeuralBuilder Learning Rules

Beginning users should use Momentum learning for all layers. Though not potentially as fast as Quickprop, DeltaBarDelta or ConjugateGradient, Momentum learning is generally more stable.

It is usually quite safe to go with the recommended values for the learning parameters. The "GA" checkboxes next to the learning parameters are used to indicate that a genetic algorithm will be used to determine the parameters. Genetic optimization is beyond the scope of this manual. Please see the documentation for NeuroSolutions and the NeuralBuilder for more information on this topic.

Example

Although non-linear, the exclusive-or problem is not particularly difficult. Therefore, we will start out with 3 processing elements:

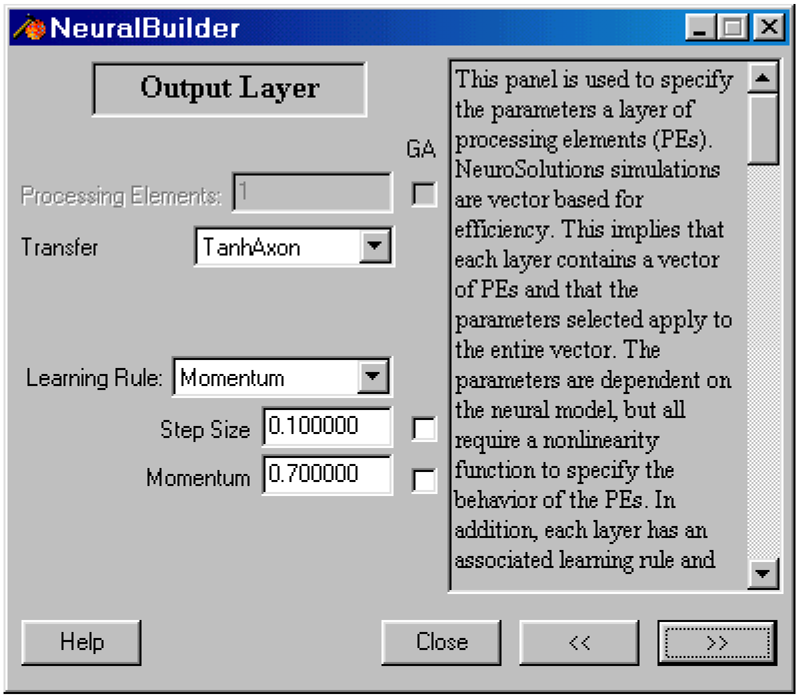
ü Enter "3" into the **Processing Elements** edit box.

Use the default "TanhAxon" as the layer's **PE Transfer** function.

ü Use the default "Momentum" as the **Learning Rule**, along with the default "0.1" **Step Size** and "0.7" **Momentum**.

ü Click the *Forward* button to advance to the next panel.

NeuralBuilder Output Layer Panel



Processing Elements: 1 □

Transfer: TanhAxon

Learning Rule: Momentum

Step Size 0.100000 □

Momentum 0.700000 □

This panel is used to specify the parameters a layer of processing elements (PEs). NeuroSolutions simulations are vector based for efficiency. This implies that each layer contains a vector of PEs and that the parameters selected apply to the entire vector. The parameters are dependent on the neural model, but all require a nonlinearity function to specify the behavior of the PEs. In addition, each layer has an associated learning rule and

NeuralBuilder Output Layer Panel

This panel is identical to the panel for the hidden layer(s), except that the number of Processing Elements is completely determined by the number of channels of desired response data.

Output File Normalization

Just as for the input data, the NeuralBuilder will automatically normalize the desired response data to match the range of the output transfer function. For example, if the output layer is a TanhAxon, the desired response data will be normalized to lie between –1 and 1.

Transfer Function

It is important to choose the output layer's transfer function to match the problem. Use the following table as a guide to choosing the proper output Axon:

Problem	Description	Output Range	Output Axon
Multi-way classification	1 of N classification.	0 to 1	SoftMaxAxon
Two-way classification	1 of 2 classification, with only a single desired channel.	-1 to 1 or 0 to 1	TanhAxon or Sigmoid
Regression	Desired response is a continuous function of the input.	Infinite	Axon BiasAxon LinearAxon

Again, it is usually quite safe to stick with the recommended values for the learning parameters.

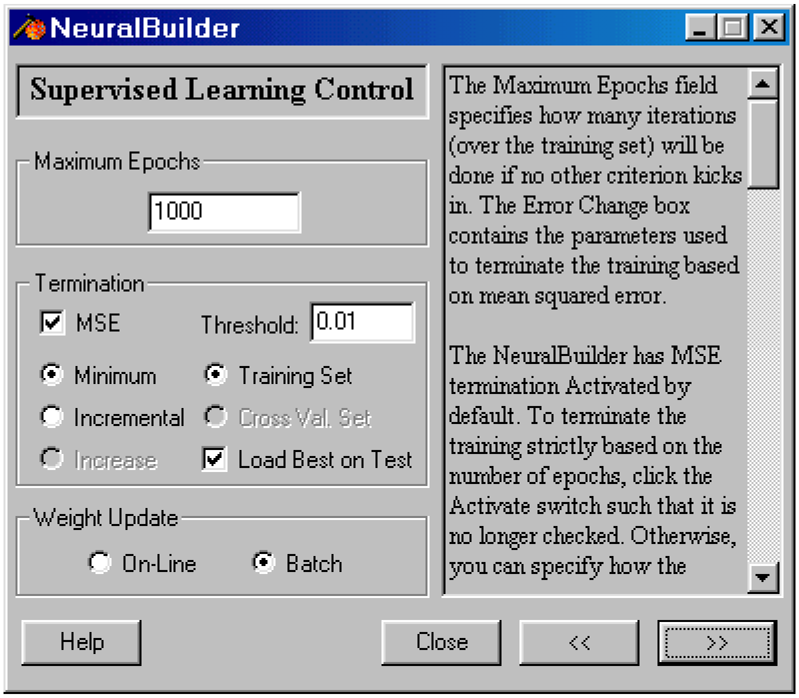
Example

The exclusive-or problem has only a single channel of desired response, and therefore the number of PE's chosen by the NeuralBuilder is 1. We will choose the TanhAxon as the output layer transfer function.

ü Use the default "TanhAxon" for the output layer **PE Transfer** function.

ü Click the *Forward* button Next > to advance to the next panel.

NeuralBuilder Supervised Learning Panel



Supervised Learning Control

Maximum Epochs: 1000

Termination

☑ MSE Threshold: 0.01

● Minimum ○ Incremental ○ Increase

● Training Set ○ Cross Val. Set ☑ Load Best on Test

Weight Update

○ On-Line ● Batch

The Maximum Epochs field specifies how many iterations (over the training set) will be done if no other criterion kicks in. The Error Change box contains the parameters used to terminate the training based on mean squared error.

The NeuralBuilder has MSE termination Activated by default. To terminate the training strictly based on the number of epochs, click the Activate switch such that it is no longer checked. Otherwise, you can specify how the

NeuralBuilder Supervised Learning Panel

The ***Supervised Learning*** panel is where you:

§ Enter the **Maximum Epochs** to train.

§ Specify **Termination** criteria for stopping the training.

§ Choose the frequency of **Weight Update**.

Maximum Epochs

Many problems should be learnable within the default 1000 epochs; thus the default is a reasonable starting point.

Termination Criteria

The **Termination** criteria, when "MSE" (mean square error) and "Minimum" are chosen, sets up the necessary breadboard components so that the training stops when the error reaches the "Threshold." Stopping the learning in this fashion keeps the network from overtraining, and thus is particularly desirable if cross validation is not being used. The default "Threshold" is usually adequate.

When cross validation is used, the default configuration is to have the training stop when the cross validation error begins to increase. The best weights of the network are automatically saved at the point when the cross validation error is at its lowest point. When testing the network, these best weights are loaded into the network before the testing data is fed through the network (if the "Load Best on Test" switch is set).

Weight Update

The **Weight Update** option determines when the network weights are updated. "Online" learning updates the weights after the presentation of each exemplar, while "Batch" learning waits until the entire training set has been presented, and thus effectively averages the weight updates over the data set. For beginning users, we recommend "Batch" weight update because the learning is stable over a wider range of step sizes. However, for large data sets, the "Online" weight update can be significantly faster, although it requires a more careful tuning of the learning parameters.

Example

The exclusive-or problem is not particularly difficult, therefore we will stop the training after 200 epochs maximum. However, we do not want to overtrain, so we will use a **Termination** criterion.

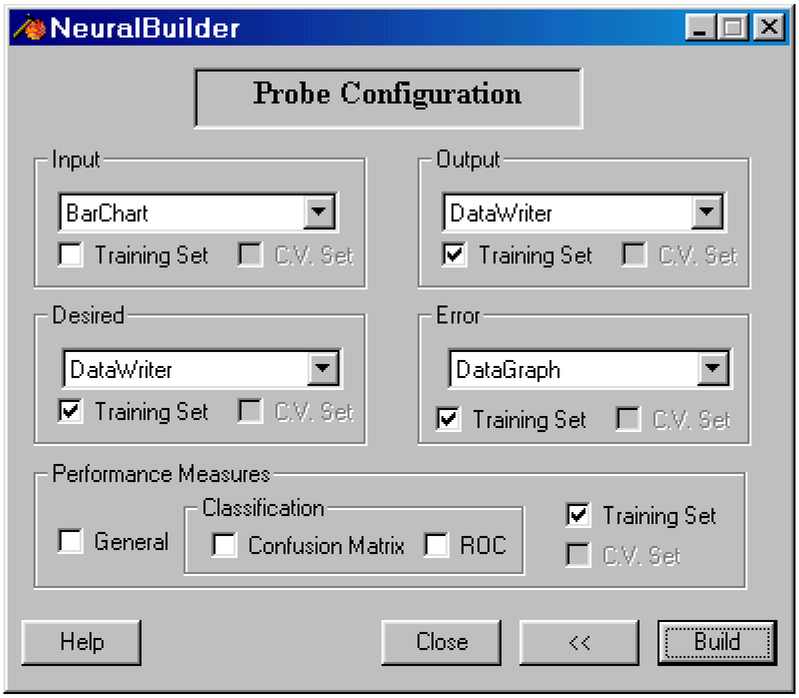
ü Enter "200" as the **Maximum Epochs**.

ü Use the defaults "MSE" and "Minimum" as the **Termination** criteria, and "0.01" as the **Threshold**.

ü Use the default "Batch" as the **Weight Update**.

ü Click the *Forward* button Next > to advance to the next panel.

NeuralBuilder Probe Configuration Panel



Probe Configuration

Input: BarChart
□ Training Set □ C.V. Set

Output: DataWriter
☑ Training Set □ C.V. Set

Desired: DataWriter
☑ Training Set □ C.V. Set

Error: DataGraph
☑ Training Set □ C.V. Set

Performance Measures
□ General
Classification: □ Confusion Matrix □ ROC
☑ Training Set
□ C.V. Set

NeuralBuilder Probe Configuration Panel

NeuroSolutions can probe the data flowing through a network, or any of its parameters, using a variety of visualization tools. The NeuralBuilder concentrates on the most important of these. In the ***Probe Configuration*** panel, you can:

- Choose the type of probe for any of the following points in the network, such as **Input**, **Output**, **Desired** response, **Error** or **Weights**.
- Monitor the performance of the network during training.
- Specify whether the probes will monitor training, cross validation, or both.
- Build the network.

Probe Types

NeuroSolutions offers many different probes. The NeuralBuilder handles the seven most important probes, shown in the chart below:

Probe	**Matrix Style?**	**Accumulates Data Over Time?**	**Feature**
BarChart	No	No	Length of bar is proportional to value.
DataWriter	Yes	Yes	Accumulates the data over time in a separate window, which can be saved as a text file. Also dumps direct to a file.
Hinton	Yes	No	Size of box is proportional to value.
ImageViewer	Yes	No	Intensity of pixels proportional to value.
MatrixEditor	Yes	No	For editing network parameters, even as a simulation is running.
MatrixViewer	Yes	No	Standard numerical display.
DataGraph	No	Yes	Graphs the signal over time.

Matrix style probes display their data in a two-dimensional array if the underlying data is arranged that way.

Learning Curve

You should always monitor the error during training, and the best way to do this is with the DataGraph. The resulting plot is called the learning curve. If you are using a cross validation data set, then you should also monitor its learning curve.

Example

We would like to view the learning curve, the network output and the desired output. Since this is a classification problem, we will also include a confusion matrix to see when all 4 exemplars are classified correctly.

ü Use the default settings of the "DataWriter" for the **Output** and **Desired** response and the "DataGraph" for the **Error**.

ü Leave the "Training Set" switches at their defaults (switched on for the **Output**, **Desired** and **Error**).

ü Check the "Confusion Matrix" box under **Performance Measures**.

Note that because we are not performing cross validation, that option is not available for the probes in this example.

Building the Network with the NeuralBuilder

Upon clicking the *Build* button Build from the ***Probe Configuration*** panel, the NeuralBuilder opens NeuroSolutions and sends the commands necessary to build the network, exactly as you have designed it.

Keeping the NeuralBuilder in the Background for Future Use

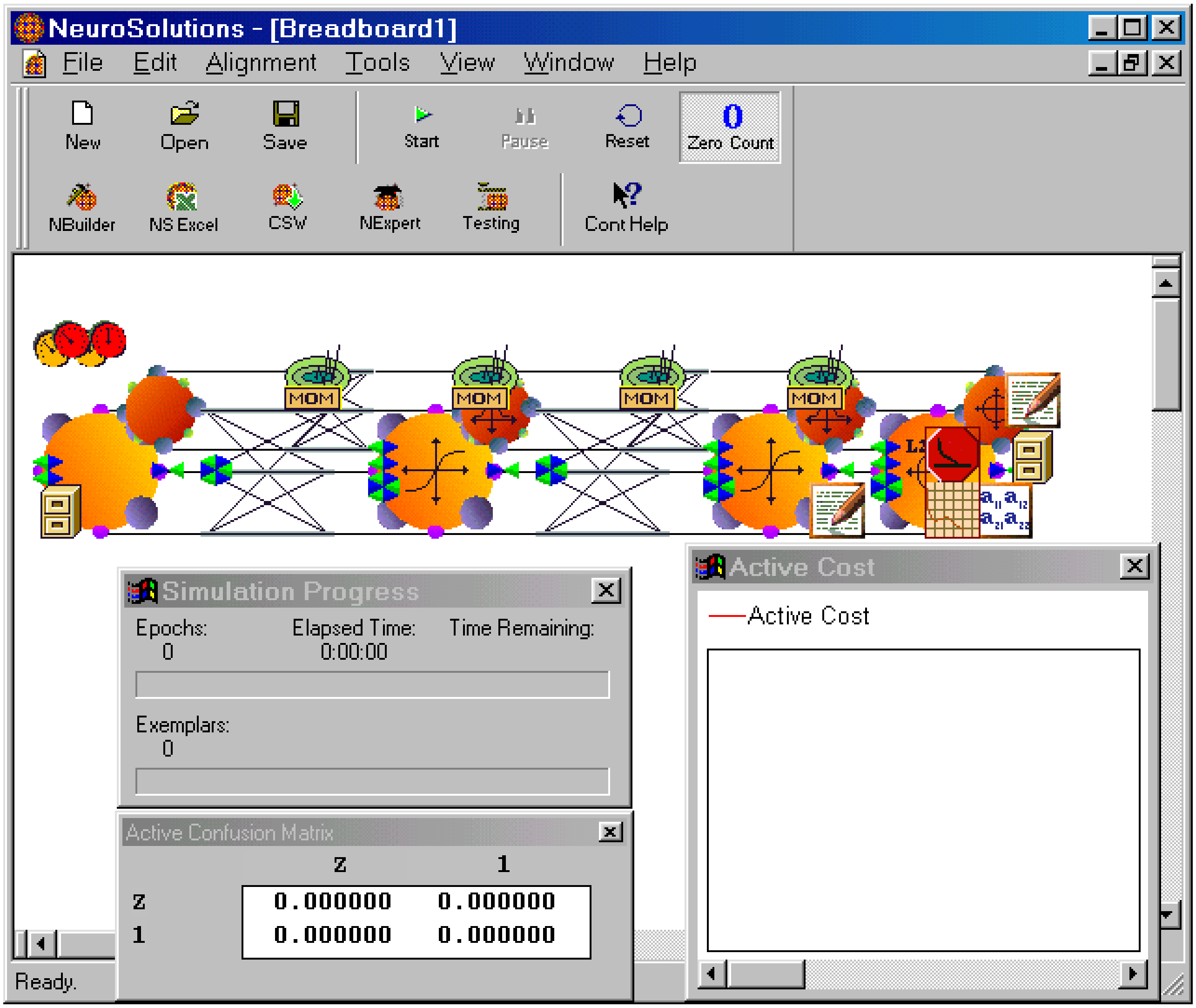
After the network has been built, the NeuralBuilder minimizes itself to the background. We recommend that you keep the NeuralBuilder in the background with all of your most recent settings. Should you decide to change your design, simply return to the NeuralBuilder, backtrack using the *Back* button <<, make only the necessary changes, and then re-build the network.

Example

Finally, we are ready to build the Multilayer Perceptron.

ü Click the *Build* button Build.

Within NeuroSolutions itself, you should see the following network being built:



Simulation Progress

Epochs: 0

Elapsed Time: 0:00:00

Time Remaining:

Exemplars: 0

Active Confusion Matrix

	z	1
z	0.000000	0.000000
1	0.000000	0.000000

Example MLP Breadboard

Once a breadboard has been built, your first step should be to save it:

ü Go to the *File* menu and select "Save as…".

Save the breadboard using a name and location of your choice.

Summary of Neural Network Theory

It is not necessary to know the details of neural networks in order to use them, but this basic introduction can be helpful. A complete coverage of neural network theory can be found within the interactive book "Neural and Adaptive Systems: Fundamentals Through Simulations" (ISBN: 0471351679) by Principe, Euliano, and Lefebvre. More information on this book is available from the NeuroDimension web site at:

http://www.nd.com/products/nsbook.htm

Neural Network Definition

A neural network is an adaptable system that can learn relationships through repeated presentation of data and is capable of generalizing to new, previously unseen data. Some networks are supervised, in that a human determines what the network should learn from the data. In this case, you give the network a set of inputs and corresponding desired outputs, and the network tries to learn the input-output relationship by adapting its free parameters. Other networks are unsupervised, in that the way they organize information is hard-coded into their architecture.

Neural Network Use

Neural networks are used for both regression and classification. In regression, the outputs represent some desired, continuously valued transformation of the input patterns. In classification, the objective is to assign the input patterns to one of several categories or classes, usually represented by outputs restricted to lie in the range from 0 to 1, so that they represent the probability of class membership.

Important Neural Network Theories

- For regression, it can be shown that a single hidden layer Multilayer Perceptron can learn any desired continuous input-output mapping if there are sufficient numbers of axons in the hidden layer(s).
- For classification, Multilayer Perceptrons can learn the Bayesian posterior probability of correct classification. This means that the neural network takes into account the relative frequency of occurrence of the classes, giving more weight to frequently occurring classes.
- For temporal problems, it can be shown that recurrent networks can follow any desired trajectory over time.

The Neural Network Design and Use Life Cycle

The Neural Network design and use life cycle is a complex dynamic process with many steps. However, careful consideration of the following steps can yield a vast improvement in neural network performance.

1) **Understand the Data**

   Neural networks cannot be used as black boxes, even in the best of circumstances. There is no substitute for a firm understanding of the data. Explore the data in as many ways as possible:

   a) Try to understand the physical process that produced the data.

   b) Plot the data: Plot the individual channels, as well as one channel against another.

   c) Examine the statistics: Calculate the within channel mean and variance, as well as the between channel correlations.

   d) Use digital signal processing (DSP) analysis techniques, such as the Fast Fourier Transform (FFT), to understand the data in the frequency domain.

2) **Preprocess the Data**

   The goal here is to take the insight gained in (1) above, and to encode it into the data. Examples include:

   a) Transform the data so that it better represents "features" of the known physical process.

   b) Transform the data to a more compact representation using universal techniques such as Principal Component Analysis (PCA) or Kohonen Self-Organizing Feature Maps (SOFM). Note that these techniques can also be implemented online at the first layer of a neural network.

   c) Eliminate superfluous input channels. It is also possible to identify superfluous input channels after a network has been trained (see (7) below).

3) **Choose a Desired Input-Output Mapping**

   Decide what the neural network is to accomplish. In particular, what is to be the desired input-output relationship? Sometimes this can require hand coding of the desired data.

4) **Choose a Neural Architecture** (see Choosing a Neural Architecture)

   For regression, always start out with a linear network. For classification, always start out with a linear discriminant classifier. Even if these networks do not perform well, they provide a baseline comparison for other networks as you graduate in complexity. Also, a consideration here is whether an unsupervised network can perform the desired input-output mapping.

5) **Train the Network** (see Neural Network Training Hints)

   If possible, monitor the training with a subset of the training exemplars set aside as a cross validation set. If the data set is too small to use cross validation, then stop the training when the learning curve first starts to level off.

6) **Repeat the Training**

   There is a high degree of variability in the performance of a network trained multiple times, but starting from different initial conditions. Therefore, the training should be repeated several times, varying the size of the network, and/or the learning parameters. Among those networks that perform the best (on the cross validation set, if available), choose the one with the smallest number of free weights.

7) **Perform Sensitivity Analysis**

   Sensitivity analysis measures the effect of small changes in the input channels on the output, and is computed over the whole training set. It can be used to identify superfluous input channels. Eliminate those channels and repeat the training process.

8) **Test the Network on New Data**

   This is where you put the network to use. If you have carefully followed the previous steps, the network should generalize well to new data.

9) **Update the Training**

   Occasionally, when enough new data is accumulated, include old test data in with the existing training set, and repeat the entire training process.

Choosing a Neural Architecture

Neural networks can be very powerful learning systems. However, it is very important to match the neural architecture to the problem. As seen previously, the NeuralBuilder constructs the most popular neural architectures. The chart below is a guide to choosing one of the NeuralBuilder's models for your problem.

Model	Description	Primary Use or Advantage
Multilayer Perceptron (MLP)	The most widely used neural network	General Classification or Regression.
Generalized Feedforward MLP	MLP plus additional layer-to-layer forward connections	Additional computing power over standard MLP.
Modular Feedforward	Several parallel MLP's that combine at the output	Reduced number of weights between layers compared to standard MLP.
Radial Basis Function (RBF)	Linear combination of Gaussian Axons	Fast training, simple interpretation of Gaussian centers and widths.
Jordan and Elman	MLP with non-adaptable recurrent feedback	Adds fixed memory to the MLP for simple temporal problems with fixed temporal dependencies.
Principal Component Analysis (PCA) Hybrids	Unsupervised PCA at input followed by a supervised MLP	Project high dimensional redundant input data onto smaller dimension. Resulting outputs are orthogonal.
Self-Organizing Feature Map (SOFM) Hybrid	Unsupervised SOFM at input followed by a supervised MLP	Project high dimensional data onto smaller dimension while preserving neighborhoods.
Time Lagged Recurrent	Locally recurrent layer(s) with a single adaptable weight	For temporal problems with short temporal dependencies. Guaranteed stability, simple interpretation of recurrent weight in terms of memory depth of data.
General Recurrent	Fully and partially recurrent networks	For more difficult temporal problems, the most powerful neural network, but also the most difficult to train. Often become unstable.
CANFIS	Adaptable Fuzzy membership function at the input	For poorly defined problems. The fuzzy preprocessing makes the neural network's job easier by characterizing inputs that are not easily discretized, often resulting in a better overall model.
Support Vector Machine	Trained using the kernel adatron algorithm.	Strictly used for small to medium sized classification problems. The SVM is especially effective in separating sets of data that share complex boundaries.

Within NeuroSolutions itself, you can manually construct an even wider variety of neural architectures.

Neural Network Training Hints

Here are some key tips for getting better results. Keep in mind that the NeuralBuilder automatically implements many of these suggestions for you.

Speeding Up Training

§ For large data sets, use the online weight update.

Updating the network weights after the presentation of each exemplar acts as a random source of perturbation to the gradient descent, which can help keep the network from getting stuck in a local minima.

§ On the hidden layers, use the Tanh non-linearity, not the Sigmoid.

Sigmoids produce zeros, and learning does not occur when the input to a learning element is zero.

§ Scale your data by input channel to match the range of the first hidden layer's transfer function.

Scaling your data by channel to match the first hidden layer's range can speed up training because a solution is not far from the initial weight values in the weight space (the relative magnitudes of the weights before and after training are comparable). For the Tanh non-linearity, scale the data to lie between –1 and 1.

§ Use a small learning rate for layers near the output, and an increasingly larger learning rate as you move back in the network towards the input.

The gradient gets attenuated by each layer as it is backpropagated from the output to the input, and this can cause layers near the input to learn very slowly.

Improving Generalization

§ Don't train forever.

Overtraining can actually degrade performance on the test set. If you can, set aside part of your training data for cross validation, and stop the training when the performance on the cross validation set deteriorates. If your data set is too small to afford cross validation, then stop the training when the learning curve (error vs. epoch) just begins to flatten out.

§ Repeat the training several times.

Neural networks are highly non-linear and can have many local minima in the weight space. Repeating the training increases your chance of finding, or getting closer to, the global minimum.

§ For multi-way classification, where the desired response is either 0 or 1, use a SoftMax at the output layer (to constrain the outputs to sum to one). Monitor the confusion matrix, in addition to mean square error, to monitor the training progress.

If you use a SoftMax output, then you can interpret the outputs directly as probabilities, even for the test set.

§ For regression, where the desired response varies continuously over some range, use a linear output.

Even if your data is scaled, the larger dynamic range of a linear output will both speed up learning and improve generalization.

§ Find the minimum number of total PE's in the network such that it is still able to learn the problem.

You can either start with a small number of PE's per layer and then increase until the network is first able to learn the problem, or you can start with a large number of PE's and then decrease until the network no longer learns. The network with the smallest number of free weights is also the one most likely to generalize well on new data. As a general rule, start with a network that has approximately one free weight for every ten exemplars in the training set.

§ For high-dimensional input data, use a Principal Component Analysis (PCA) layer at the input to project the input onto a smaller dimension.

The smaller dimensional input reduces the number of free weights in the layers that follow, which can improve generalization, and the orthogonalization of the input can speed up training.

File Format Requirements

Whereas NeuroSolutions can input data in many formats, including binary and bitmaps, the NeuralExpert and NeuralBuilder require that your data be formatted as follows:

- ASCII file format.
- Data arranged as columns, which may include inputs, desired outputs, or data that is not used by the neural network.
- Each row contains one exemplar (pattern) of data (with the exception of the first row, which may contain column headings).
- The data elements within each row are separated by tabs, spaces, or commas.

Example

```
SpecimenNumber,Species,FrontalLip,RearWidth,Length,Width,Depth,Male,Female
142,0,20.6,14.4,42.8,46.5,19.6,1,0
19,1,13.3,11.1,27.8,32.3,11.3,1,0
169,0,16.7,14.3,32.3,37,14.7,0,1
56,1,9.8,8.9,20.4,23.9,8.8,0,1
164,0,15.6,14.1,31,34.5,13.8,0,1
53,1,9.1,8.1,18.5,21.6,7.7,0,1
```

{ewl RoboEx32.dll, WinHelp2000, }