APPENDIX A

Installation procedures

This appendix describes the contents of the Macintosh and Windows 95 **tlearn** distribution disk and how to install **tlearn. tlearn** is a connectionist modelling simulation package that provides a graphical user interface (GUI) to backpropagation neural networks. There are versions of **tlearn** for Macintosh and Windows 95 and XWindows/ Unix.

Mactintosh installation

The Macintosh distribution disk contains the Macintosh **tlearn** (version 1.0.1) executable and associated exercise files. The tlearn executable file is a FAT executable which means that it can be run on both PowerPC and 680x0 Macintosh machines. The disk also provides a set of Chapter folders which contain the project files corresponding to the exercises in this book.

To install **tlearn,** copy the compressed file called **mac_tlearn** to the folder on your Macintosh where you want the application to reside. This file is a self-extracting-archive (SAE) which means that you only need to double-click on the file to begin the installation. (This installer was created usingstuffit InstallerMaker. © 1990-96 Aladdin Systems, Inc.) You will be asked where you want to place the programme and assciated exercise files. The executable is called **tlearn** and can be launched without any other changes to the machine. Note that for simulation of large networks, it may be necessary to increase the memory of application through the **Get Info...** window available within the finder.

Windows 95 installation

The Windows 95, distribution disk contains the Windows 95 **tlearn** (version 1.0.1) executable and associated exercise files. To install **tlearn**, copy the compressed file called **win_tlearn(** .exe) (using Windows Explorer) to the folder on your PC where you want the application to reside. This file is a self-extracting-archive (SAE) which means that you only need to double-click on the file to begin the installation. The executable is called **tlearn** and can be launched without any other changes to the machine. The disk also provides a set of Chapter folders which contain the project files corresponding to exercises in this book. We do not recommend that **you** use **tlearn** running under Windows 3.x (even with Win32s extensions installed).

Updates and bug reports

We do not guarantee that the software provided with this book is free of bugs. In fact, we guarantee that if you try bard enough you will find some situations where **tlearn** will break! **Please** tell us about any software problems you experience by emailing us at
innate@crl.ucsd.edu.
We may not respond to your queries immediately but we will fix the bugs as time permits. You can obtain thelatest versions of **tlearn** via anonymous **ftp at crl . ucsd. edu (in the pub directory)** or on the world wide web at http: //crl .ucsd.edu/innate. This site is **situated in** San Diego. Users from the Old World may find it easier to obtain the laeest versions of **tlearn** via anonymous ftp at f tp . psych. ox. ac. uk (also in the pub directory). This site is situated in Oxford, UK.

APPENDIX B *User manual*

This reference manual provides documentation on **`tlearn`** software version 1.0. The first section provides a quick introduction to the software and its functionality. The second section gives complete descriptions of the application's menus and dialog boxes. The third section offers a quick reference to command keys and other shortcuts. The final section offers some advice on troubleshooting and other error messages.

Introduction

`tlearn` is a neural network simulator which implements the backpropagation learning rule, including simple recurrent network learning and backpropagation through time, and provides a number of displays which show the network internals. **`tlearn`** includes a fully functional text editor as well as a number of data analysis utilities.

Configuration files

`tlearn` organizes files and simulations into projects. The project file is a machine readable file which stores information about option settings for training and testing. The name of the project file specifies the name of the project and acts as the prefix for other associated files for the project. There are three necessary associated files for every project. Namely the **`<fileroot>.cf`**, **`<fileroot>.data`** and

<fileroot>.teach files, where **<fileroot>** is the name of the project file. A complete description of these files is given in the section on *Network Configuration and Training Files*.

Editor functions

The **tlearn** text editor includes standard text editor features (e.g., Find and Replace, Cut, Copy and Paste, Revert, Go To Line, and a current line number indicator). As well as the standard features **tlearn** provides two text utilities namely **Sort…** and **Translate…** which are found in the Edit menu. Sort provides a mechanism for sorting of files with arrays of numeric values. **Translate** allows a set of find and replace actions to be carried out in an automated way. These utilities are fully described in the section *Menu and dialogue reference* where the description of their associated dialogue boxes is given. Editor short cut keys are given in the section *Command key and shortcut reference*.

Network training and testing functions

tlearn provides two types of network run modes: training and testing. These functions are found in the **Network** menu. The **Train the network** action begins training from an initial set of random weights and trains the network for the specified number of sweeps. The number of training sweeps is set in the **Training Options** dialogue box. The **Resume training** action resumes training from the current set of weights for a further specified number of sweeps. Further options and settings for training are described in the *Menu and dialogue reference* section where the **Training Options** dialogue is described.

The **Verify network has learned** action presents the trained network with a testing set specified in the **Testing Options** dialogue box. The values of the output node/s for each data presentation are given in the **Output** window. The **Probe selected nodes** action similarly presents the specified data set to the network and in this case outputs the values of the selected nodes specified in the **.cf** file. All the training and testing actions can be aborted by choosing **Abort** from the **Network** menu or the **tlearn** status display.

Network utilities

There are two Network utilities located in the Special menu. These are **Output Translation...** and **Lesioning...** . The **Output Translation...** utility allows the translation of 0/1 vectors to text. When the network operation is verified, or the network activation display is used to display network outputs, the output translation can also be given. The **Output Translation...** utility is described fully in *Menu and dialogue reference* section where the Output Translation dialogue is described.

The **Lesioning...** utility allows weight files for a specific network to be modified so that the effects of lesioning the network, i.e., removing some connections or nodes, can be examined. The **Lesioning...** utility is described fully in the *Menu and dialogue reference* section where the **Lesioning** dialogue is described.

Data analysis functions

There are two Data Analysis utilities located in the **Special** menu, namely **Cluster Analysis...** and **Principal Components Analysis....** The **Cluster Analysis...** utility allows a hierarchical clustering to be performed on a set of data vectors. The **Cluster Analysis...** utility is described fully in the *Menu and dialogue reference* section where the Cluster Analysis dialogue is described.

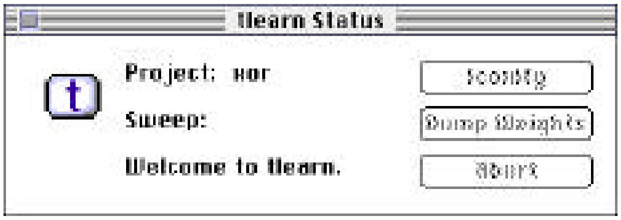
The **Principal Components Analysis...** utility allows the principal components of a set of data vectors to be determined and the projection of such data onto its principal components to be obtained. The **Principal Components Analysis...** utility is described fully in the *Menu and dialogue reference* section when the Principal Components Analysis dialogue is described.

`tlearn` displays

`tlearn` has seven displays which illustrate different aspects of the network simulation or show the results of some of the data analysis utilities. Three of the displays can only be selected when there is a project open as they rely on the definition of the network provided by the object files. The project dependent displays are the **Node Activation** display, the **Connection Weights** diagram and the **Network**

Architecture display. All of the displays (apart from the **Status** display) can be copied to the clipboard and pasted into word processors or other editors that can handle picture graphics.

The **tlearn Status** display (shown in Figure B.1) indicates the name of the open project, if there is one, and the current state of **tlearn** training. The **Status** display also provides buttons to allow the display to be iconified; for the current network weights to be saved to a file; and, for the network training to be aborted


tlearn Status

Project: xor

Sweep:

Welcome to tlearn.

FIGURE B.1 **tlearn Status** display

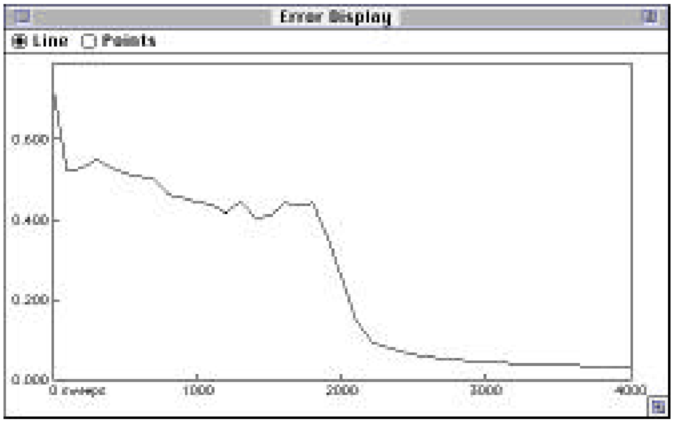


FIGURE B.2 **Error Display**

The **Error Display** (Figure B.2) gives a graph of error values for the training or testing actions carried out on the current network. Selection radio buttons at the top of the display allow the error graph to be shown as lines or as points.

The **Node Activations Display** (Figure B.3) presents the activations of the nodes of the network as Hinton diagram displays. The orientation of the nodes on this display is specified by the network orientation set on the **Network Architecture** display. Two buttons (**First Data Pattern**, **Next Data Pattern**) allow the user to step through the training or testing patterns presented to the network.

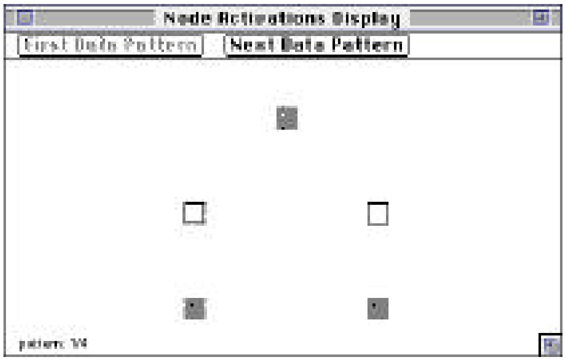
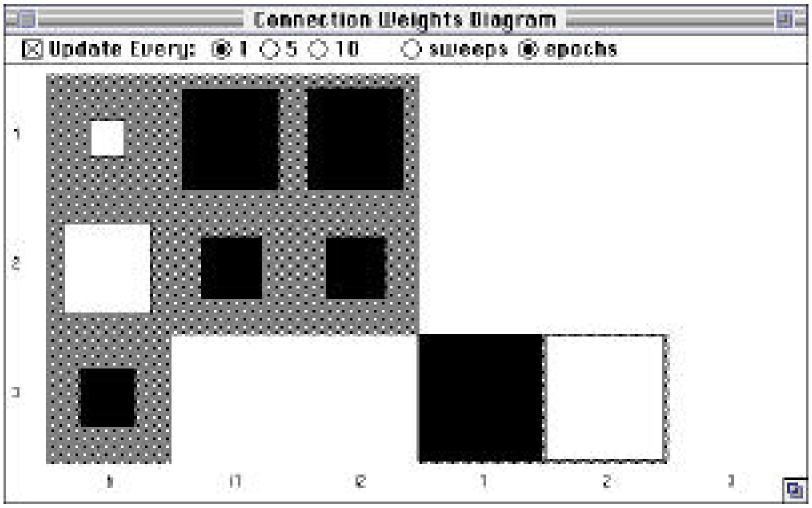


FIGURE B.3 **Node Activations Display**

The **Connection Weights** display (Figure B.4) displays the current network weights with a Hinton diagram showing the weight magnitudes and signs. Controls at the top of the display allow the user to specify whether the display should be updated during training, and the regularity of the diagram update, specifically updates can be set for every 1, 5, or 10 sweeps or epochs.

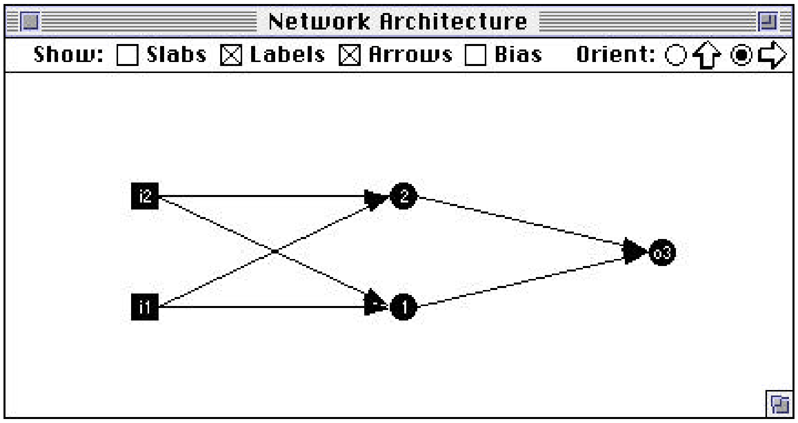
The **Network Architecture** display (Figure B.5) draws the network nodes and connections. The check boxes and radio buttons at the top of the display allow the following settings: The **Slabs** check box, when checked cause the input, hidden and output units to be shown as single slabs rather than as individual units. The **Labels** check box toggles the display of node labels. The **Arrows** check box toggles the appearance of arrow heads on the connection lines. The **Bias** check box toggles the appearance of the bias node. The **Orient:** radio buttons specify the orientation of the network display as bottom-to-top or left-to-right. The orientation specified here is also used in the **Node Activations** display.



Connection Weights Diagram

☒ Update Every: ● 1 ○ 5 ○ 10 ○ sweeps ● epochs

FIGURE B.4 **Connection Weights Diagram**

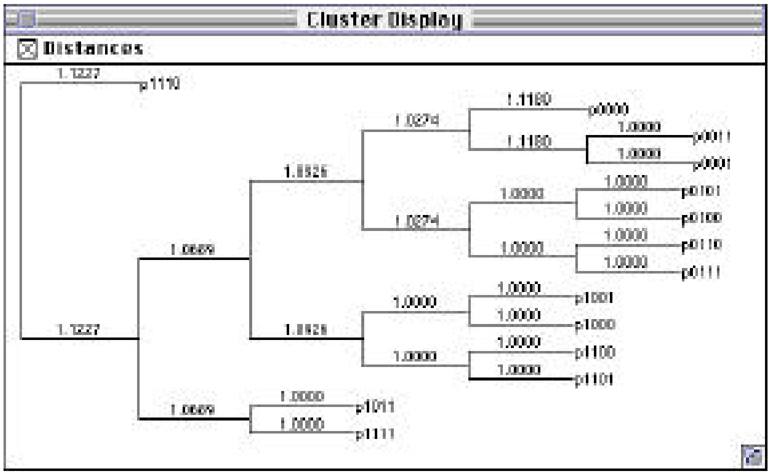


Network Architecture

Show: □ Slabs ☒ Labels ☒ Arrows □ Bias Orient: ○ ⇧ ● ⇨

FIGURE B.5 **Network Architecture** display

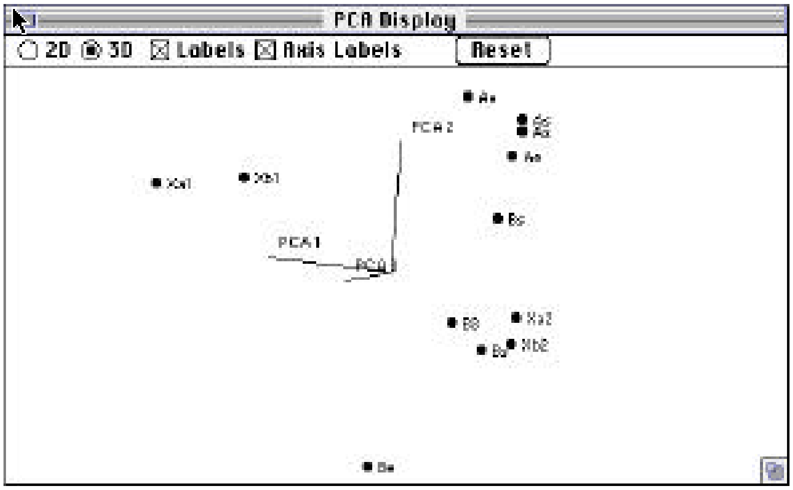
The **Cluster Display** (Figure B.6) is used to present the output of the **Cluster Analysis...** utility (found in the **Special** menu). **Cluster analysis** generates a cluster diagram which is a tree illustrating the clusters of the data. The **Distances** check box toggles the appearance of distance values on the cluster diagram in the display.



☒ Distances

FIGURE B.6 **Cluster Display**

The **Cluster Analysis...** utility is described more fully in the *Menu and dialogue reference* section, where the **Cluster Analysis** dialogue is described.



○ 2D ◉ 3D ☒ Labels ☒ Axis Labels

FIGURE B.7 **PCA Display**

The **PCA Display** (Figure B.7) is used to present the output of the **Principal Components Analysis...** utility (found in the **Special** menu). The output of principal components analysis is the projection of the data points onto their principal components. The **2D**/**3D** radio buttons allow switching between 2 and 3 dimensioned projections. The **Labels** check box toggles the appearance of labels for the data points. The **Axis Labels** check box toggles the appearance of labels for the axes shown in the display. The **Reset** button is used for **3D** projections to return the projection to the standard orientation. For **3D** projections the orientation of the axes can be modified by click-dragging on the diagram which rotates the projection accordingly. The **Principal Components Analysis...** utility is described more fully in *Menu and dialogue reference* section, where the **PCA** dialogue is described.

Network configuration and training files

tlearn requires three input files: the network configuration—the **cf** file, the input pattern—the **data** file, the output (teacher) pattern—the **teach** file and the project file. An additional input file may be used to specify a reset schedule for context nodes—the **reset** file. **tlearn** may also create output files for weights, error, node activations, etc. All files should begin with the same name; this is referred to as the *fileroot*. The project file is called **<fileroot>**. The project file is created automatically when the user starts a new project in **tlearn**. The different files are distinguished by having different extensions (where an extension consists of a period followed by a fixed designator). Every **tlearn** simulation will contain at least the following 4 files:

```
<fileroot>
<fileroot>.cf
<fileroot>.data
<fileroot>.teach
```

The optional reset file is called:

```
<fileroot>.reset
```

Network configuration (`.cf)` file

This file describes the configuration of the network. It must conform to a fairly rigid format, but in return offers considerable flexibility in architecture. There are three sections to this file. Each section begins with the keyword in upper case, flush-left. The three section keywords are **`NODES:`**, **`CONNECTIONS:`**, and **`SPECIAL:`**. Note the colon. Sections must be described in the above order.

`NODES:`

This is the first line in the file. The second line specifies the total number of nodes in the network as "**`nodes = #`**". Inputs do *not* count as nodes. The total number of inputs is specified in the third line as "**`inputs = #`**". The fourth line specifies the number of nodes which are designated as outputs according to "**`outputs = #`**'. (Note that these two lines essentially give the lengths of the **`.data`** and **`.teach`** vectors.) Lastly, the output nodes are listed specifically by number (counting the first node in the network as 1) in the order that the **`.teach`** information is to be matched up with them. The form of the specification is "**`output nodes are <node-list>`**". (If only a single output is present one can say "**`output node is #`**"). If no output nodes are present, this line is omitted. Spaces are critical.

Node number can be important for networks in which there are fixed copy-back links. Copy-back links allow for saving of node activations so that they can be used on the next sweep. Because node activations are calculated in ascending order, with the order determined by the number of the node, it is important that node activations be saved *after* they are calculated. It is also important that a unit which receives input from a node which is serving as a state/context node (and has thus storing some other nodes activation from the previous time cycle) calculate its activation before the state/context node gets updated on the current sweep. Both considerations lead to the following rule of thumb: Any node receiving input from a state/context node must have a *lower* node number than the state/context node. This is illustrated in the example .cf file at the end of this section.

CONNECTIONS:

This is the first line of the next section. The line following this must specify the number of groups, as in "**groups = #**" (All connections in a group are constrained to be of identical strength; e.g., as in the translation invariance problem in Chapter 7; in most cases **groups = 0**.) Following this, information about connections is given in the form:

```
<node-list> from <node-list> [= <fixed> | <group #> | <min & max>]
```

If values are specified for **<min & max>** (e.g., -5.0 & 5.0) then the weights for the relevant connections will not be allowed to exceed these minimum and maximum values. Weights specified as **fixed** will have values fixed at their initialization values (if **<min & max>** are set to 1.0 & 1.0, then the weights are set to 1.0 and remain unchanged throughout learning; this is typically used for connections from context units).

It is also possible to say:

```
<node-list> from <node-list> = <min> & <max> fixed one-to-one
```

This last form is used, e.g., when node 1 is fed from node 4, node 2 is fed from node 5, and node 3 is fed from node 6, as opposed to the usual case of node 1 being fed from nodes 4–6, node 2 being fed by nodes 4-6, and node 3 being fed by nodes 4–6.

A **<node-list>** is a comma-separated list of node numbers, with dashes indicating that intermediate node numbers are included. A **<node-list>** contains *no spaces*. Nodes are numbered counting from 1. Inputs are likewise numbered counting from 1, but are designated as "i1", "i2", etc. Node 0 always outputs a 1 and serves as the bias node. If biases are desired, connections *must* be specified from node 0 to specific other nodes (not all nodes need be biased). Groups must be labeled by integers ascending in sequence from 1. It is also permissible to say

```
<group #> = <min & max>
```

provided that the group has already been completely defined.

SPECIAL:

This is the first line of the third and final section. Optional lines can be used to specify whether some nodes are to be linear ("**linear = <node-list>**"), which nodes are to be bipolar ("**bipolar = <node-list>**")[1], which nodes are selected for special printout ("**selected = <node-list>**"), and the initial weight limit on the random initialization of weights ("**weight_limit = <#>**"). Again, *spaces are critical*.

Example **.cf** files are given at the end of this section for several network architectures.

Network data (.data) file

This file defines the input patterns which are presented to **tlearn**. The first line must either be "**distributed**" (the normal case) or "**localist**" (when only a few of many input lines are nonzero). The next line is an integer specifying the number of input vectors to follow. Since exactly one input vector is used for each time step, this is equivalent to specifying the number of time steps. The remainder of the **.data** file consists of the input. These may be input as integers or floating-point numbers.

In the (normal) "**distributed**" case, the input is a set of vectors. Each vector contains n_i floating point numbers, where n_i is the number of inputs to the network. Note that these input vectors are always used in the exact order that they appear in the **.data** file (unless the randomization option is specified).

In the "**localist**" case, the input is a set of **<node-list>**s (defined below) listing only the numbers of those nodes whose values are to be set to one. Node lists follow the conventions described in the **.cf** file. An example **.data** file is shown below in both the "**localist"** and "**distributed"** case.

[1] Linear nodes simply output the inner -product of the input and weight v ectors, or *net*. Logistic units are sigmoidal: The activation function for each node is $y = 1/(1+e^{-\text{net}})$. Logistic node output is bounded by 0 and 1. Bipolar nodes ha ve an extended range—their output ranges continuously from -1 to +1. The activation function for bipolar nodes is $y = (2/(1+e^{-\text{net}})) - 1$.

```
distributed
4
1 0 0 0 0
0 1 1 0 0
0 1 0 0 0
1 1 1 0 1

localist
4
1
2,3
2
1-3,5
```

Network teach (.teach) file

This file is required whenever learning is to be performed. As with the **.data** file, the first line must be either "**distributed**" (the normal case) or "**localist**" (when only a few of many target values are nonzero). The next line is an integer specifying the number of output vectors to follow.

In the (normal) "**distributed**" case, each output vector contains n_o floating point numbers, where n_o is the number of outputs in the network. An asterisk ("*****") may be used in place of a floating point number to indicate a "don't care" output. In patterns containing "don't care" indicators, no error will be generated for those output units for which a ***** is specified.

In the "**localist**" case, each output vector is a set of **<node-list>**s whose targets are a 1 as opposed to a 0. Node lists follow the conventions described in the **.cf** file. An example **.teach** file for a network with one output unit is given below:

```
distributed
4
0.1
0.9
*
0.
```

Network reset (.reset) file

This file is required whenever the context nodes need to be reset to zero. As with the **.teach** file, the first line must be an integer specifying the number of time stamps to follow. Each time stamp is an integer specifying the time step (i.e., pattern number) at which the network is to be completely reset. As with the **.teach** file, the time stamps must appear in ascending order. An example **.reset** file with 2 time stamps (patterns 0 and 3) is given below:

```
2
0
3
```

Weights (<fileroot>.<runs>.wts) file

At the conclusion of a **tlearn** session, the results of training are saved in a “weights file.” This file name incorporates the fileroot, the number of learning sweeps (runs) which resulted in this network, and ends with “**wts**” as the literal extension. This file contains weights and biases resulting from training. Weights are stored for every node (except the bias node, 0), from every node (including the bias node). A sample weights file for an XOR (**2x2x1**) network is shown below. (Sources are shown explictly here for connections into node 1 only; they do not appear in the actual weights file.)):

```
NETWORK CONFIGURED BY TLEARN
# weights after 10000 sweeps
# WEIGHTS
# TO NODE 1
-6.995693        (from bias node)
4.495790         (from input 1)
4.495399         (from input 2)
0.000000         (from node 1)
0.000000         (from node 2)
0.000000         (from node 3)
# TO NODE 2
2.291545
-5.970089
-5.969466
0.000000
0.000000
0.000000
# TO NODE 3
4.426321
0.000000
0.000000
-9.070239
-8.902939
0.000000
```

This file can also be produced by requesting periodic check-pointing (dumping of a weights file) either in order to recreate intermediate stages of learning, or to avoid having to re-run a lengthy simulation in the event of premature termination. This weights file can be loaded into **`tlearn`** in order to test with a trained network.

Error (.err) file

If error logging is requested, a file will be produced containing the RMS error, saved at user-specifiable intervals.

Example .cf files

EXAMPLE 1:

This illustrates a feed-forward network which implements a **2x2x1** XOR network (cf. Chapter 4). Notice that in **tlearn**, the 2 inputs are *not* nodes; the network itself has only 2 hidden nodes and 1 output node. (There are still learnable connections from the 2 inputs to the 2 hidden nodes.)

```
NODES:
nodes = 3
inputs = 2
outputs = 1
output node is 3
CONNECTIONS:
groups = 0
1-3 from 0
1-2 from i1-i2
3 from 1-2
SPECIAL:
selected = 1-2
weight_limit = 1.0
```

EXAMPLE 2:

This illustrates a network that receives 3 inputs, has 4 hidden nodes, 2 output nodes, and 4 copy-back nodes; each copy-back node receives the activation of the corresponding hidden node at the prior cycle. Notice that the copy-back nodes are linear, receive no bias, and have fixed downward connections from the hidden nodes. In the number scheme, **i1-i3** designate the 3 inputs; nodes **1-4** are the hidden nodes; nodes **5-6** are the output nodes; and nodes **7-10** are the copy-back (state/context) nodes.

```
NODES:
nodes = 10
inputs = 3
outputs = 2
output nodes are 5-6
```

```
CONNECTIONS:
groups = 0
1-6 from 0
1-4 from i1-i3
1-4 from 7-10
5-6 from 1-4
7-10 from 1-4 = 1. & 1. fixed one-to-one
SPECIAL:
linear = 7-10
weight_limit = 1.
selected = 1-4
```

EXAMPLE 3:

This illustrates a network which receives 9 inputs, has 3 hidden nodes (`1-3`) and 1 output node (`4`). The 3 hidden nodes have limited receptive fields; each one receives connections from only 3 of the inputs. In addition, the connections are grouped (i.e., trained to assume the same values), thus benefiting from the learning that occurs for other nodes in the group (e.g., even when deprived of input). The result is that each hidden node has 3 different input weights; each of the 3 weights has a similar weight leading into the other 2 hidden nodes. This scheme is similar to the translation invariance network in Chapter 7. Finally, weights are confined to the range `-5/+5`.

```
NODES:
nodes = 4
inputs = 9
outputs = 1
output node is 4
CONNECTIONS:
groups = 3
1-4 from 0
1 from i1 = group 1
1 from i2 = group 2
1 from i3 = group 3
2 from i4 = group 1
2 from i5 = group 2
2 from i6 = group 3
```

```
3 from i7 = group 1
3 from i8 = group 2
3 from i9 = group 3
4 from 1-3
group 1 = -5 & 5
group 2 = -5 & 5
group 3 = -5 & 5
SPECIAL:
selected = 1-3
weight_limit = 0.1
```

Menu and dialogue reference

The File menu

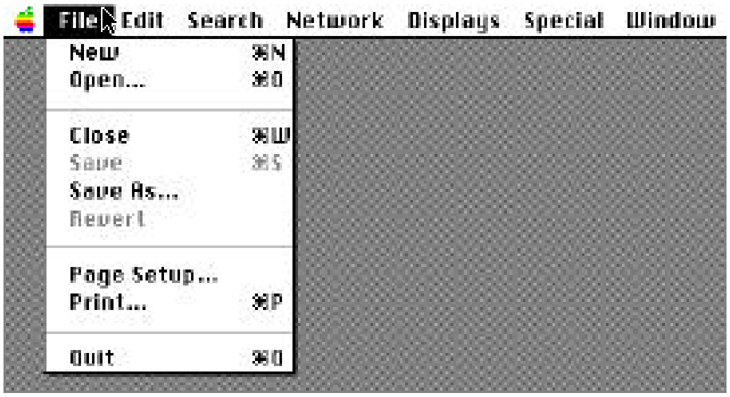


FIGURE B.8 **File** menu

Figure B.8 shows the **File** menu and associated commands which act as follows:

New Creates a new text file window.

Open... Brings up the **File Open** dialogue box to allo w selection of te xt files for editing.

Close Closes the current windo w, either a text window or a display.

Save Saves the current text window.

Save As... Brings up the **Save As** dialogue box where a name for saving the current text window can be selected.

Revert Reverts to the previously saved version of the file.

Page Setup... Brings up the **Page Setup** dialogue box.

Print... Prints the current window. Text windows and displays (apart from the **Status** display) can be printed.

Quit Quits **tlearn**.

The Edit menu

Figure B.9 shows the **Edit** menu. All but the last two commands (**Sort...** and **Translate...**) are standard. Note that the **Copy** command can also be used to copy displays to the clipboard so they can be pasted elsewhere (e.g.. word processors, graphics editors). **Sort...** and **Translate...** are text utilities used for manipulating text files for use with **tlearn**.

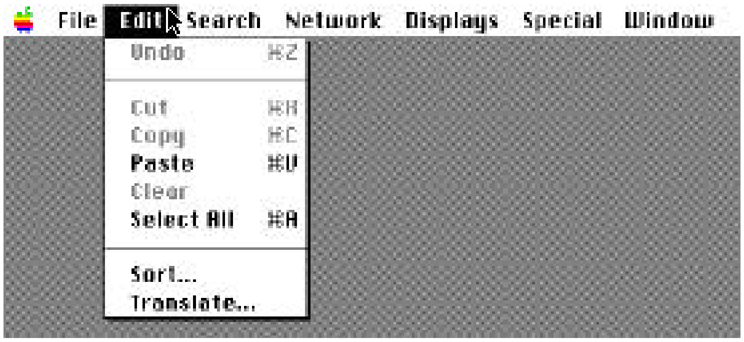
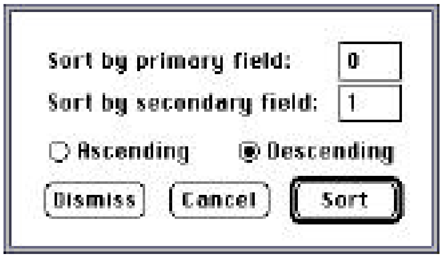


FIGURE B.9 **Edit** menu

The **Sort...** command brings up the **Sort** dialogue (shown in Figure B.10) which is used to specify settings for a **sort** action to be applied to the current text window. Sorting can only be applied to text files which contain a couple array of numerical values, that is each line of the text window must contain an equal number of numerical



Sort by primary field: 0

Sort by secondary field: 1

○ Ascending ● Descending

FIGURE B.10 **Sort** dialogue box

values separated by white space. The counting of fields begins at zero (0) rather than one (1). Sorting can be done over a primary field or a primary and secondary field, and the lines can be sorted in ascending or descending order. As with many of the dialogue boxes the **Sort** dialogue settings can be specified but dismissed (by pressing the **Dismiss** button). Instead of executing the action immediately, the **sort** dialogue simply saves the settings. Pressing the **Cancel** button reverts the settings to the previous values. Pressing the **Sort** button causes the **sort** action to be done. The lines of text are sorted and the result is returned into a new text window which is called "**<WindowName> Sorted**" where **<WindowName>** is the name of the window being sorted. Note that the text of the sorted window is not saved to a file, but when the file is saved, a **Save As** dialogue prompts the user for a filename for the new text window. An example of the use of **Sort...** is given in Chapter 5.

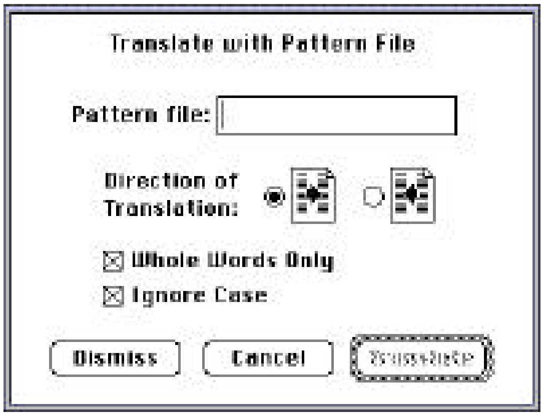
The **Translate...** command brings up the **Translate** dialogue (shown in Figure B.11) for a translation action to be applied to the current text window. The translation requires a pattern file which contains lines which specify the translations to be performed. The format of the lines which specify the translation is as follows:

```
<find string> <replace_string>
```

The **find** string cannot contain spaces, but the **replace** string, which is all of the line apart from the first word or string, can contain spaces.

An example pattern file is:

```
JE Jeff Elman
KP Kim Plunkett
TL tlearn
BP Backpropagation
```



Translate with Pattern File

Pattern file:

Direction of Translation: ● ○

☒ Whole Words Only

☒ Ignore Case

FIGURE B.11 **Translate** dialogue box

The translation action is performed in the following manner: A new window with a copy of the text of the current window is displayed (the new window is given the name of the current window with the word “Translated” appended to it), then each line of the pattern file is used in turn and the translation is applied to the new window. If the direction of translation is from left to right then occurrences of the **find** string are replaced by the **replace** string. If the direction of translation is from right to left then occurrences of the **replace** string are replaced by the **find** string. The **Whole Words Only** check box when checked ensures that the string being searched for are whole words only, that is, the string has surrounding white space. The **Ignore Case** check box allows the searching to find strings regardless of the mixture of upper and lower case letters. An example of the use of **Translate** is given in Chapter 8.

The Search menu

Figure B.12 shows the **Search** menu. The **Search** menu commands act as follows:

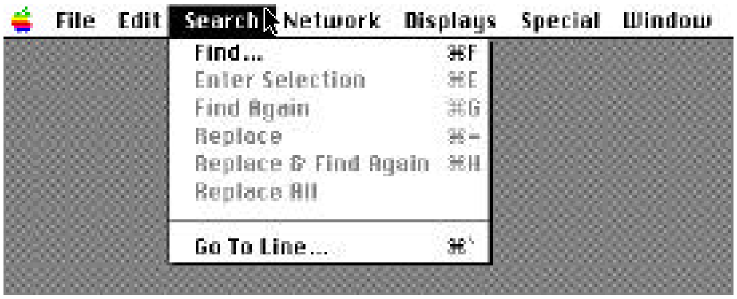


FIGURE B.12 The **Search** menu

Find... Brings up the **Find and Replace** dialogue (shown in Figure B.13), where the **Find** string and the **Replace** string can be entered and conditions for document searching can be set.

Enter Selection Copies the current selection into the **Find** string. If no text is currently selected, then this menu item is disabled.

Find Again Repeats the previous search in the current search direction.

Replace Replaces the currently selected **Find** string with the **Replace** string. If the current selection is empty or not equal to the **Find** string then no replacement is made. Hence, the **Replace** action is only sensibly done after a successful **Find** action.

Replace & Find Again Replaces the selected **Find** string with the **Replace** string and searches for the next occurrence of the **Find** string.

Replace All Replaces all occurrences of the **Find** string from the current cursor position to the end of the document. If the **Wrap Around** option is chosen then all occurrences of the **Find** string in the whole document are replaced.

Go To Line... Brings up the **Go To Line** dialogue where the cursor is moved to the line number entered.

When a **Find** string has been entered, then the **Find** button can be pressed and a search begins for occurrences of the **Find** string in the current window starting from the current cursor position. If an occurrence of the **Find** string is found, then the search stops and the occurrence is highlighted. If no occurrence is found then the system beeps. Two of the options in the **Find** and **Replace** dialogue are the same as those found in the **Translate...** dialogue; namely **Whole Words Only**, which ensures that the occurrences of the **Find** string found have surrounding white space; and **Ignore Case**, which allows searching to find strings regardless of upper or lower case letters. The **Wrap Around** option causes searching which reaches the end of the current window to begin again from the start of the document.

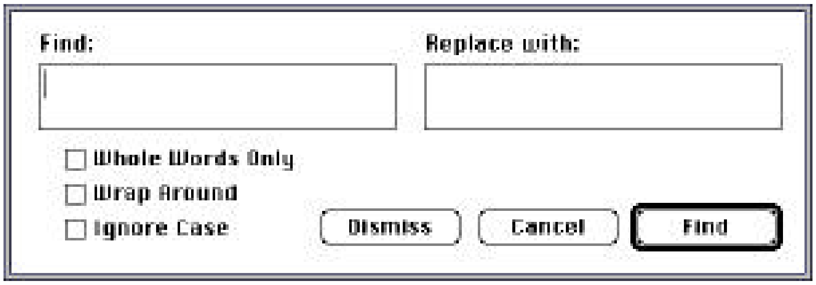


FIGURE B.13 The **Find and Replace** dialogue box

The Network menu

Figure B.14 shows the **Network** menu. The **Network** menu commands act as follows:

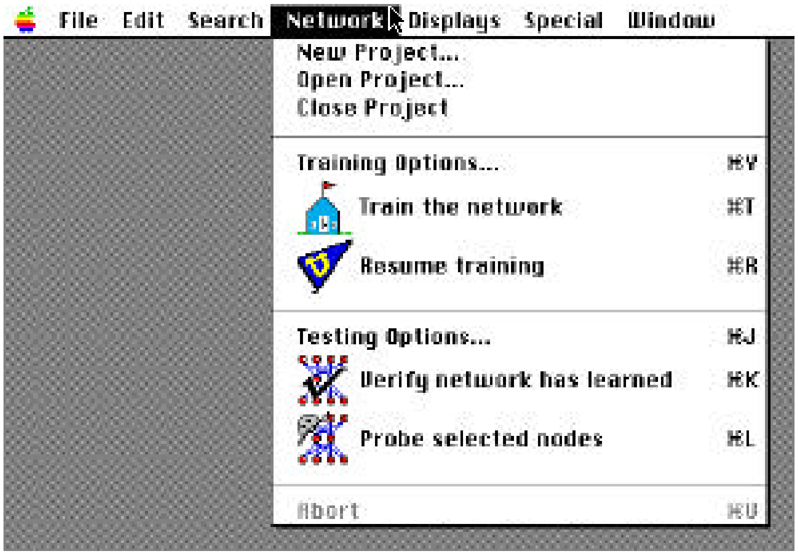


FIGURE B.14 The **Network** menu

New Project... Brings up the **New Project** dialogue box, which allows the entry of a new project name. Once the name is chosen, **tlearn** opens three new text windows associated with the project, that is, the **<name>.cf**, **<name>.data** and **<name>.teach** files. If these files already exist in the directory in which the project was opened then the files are opened, otherwise the windows are empty.

Open Project... Brings up the **Open Project** dialogue box, which allows the selection of **tlearn** project files. Once a project is chosen for opening, tlearn opens the three associated **.cf**, **.data** and **.teach** text files.

Close Project Sets the current project to none and closes the current projects **.cf**, **.data** and **.teach** file windows, if they are currently open.

Training Options... Brings up the **Training Options** dialogue box. The **Training Options** dialogue box is described in more detail below.

Train the network Begins training the network according to the training options set in the **Training Options** dialogue box.

Resume training Resumes training from the current set of network weights.

Testing Options… Brings up the **Testing Options** dialogue box. The **Testing Options** dialogue box is described in more detail below.

Verify network has learned Runs the **Verify Network** action which presents the testing set (specified in the **Testing options** dialogue) and calculates the output values of the output units and prints these values into the **Output** window. If the **Output** window is not currently opened then it is opened and selected.

Probe selected nodes Runs the **Probe Network** action which presents the testing set (specified in the **Testing Options** dialogue) and calculates the output values of the selected nodes and prints these values into the **Output** window. The selected nodes are specified in the .cf file.

Abort Aborts a currently running network action.

The Displays menu

Figure B.15 shows the **Displays** menu. The display menu indicates with check marks which displays are currently being shown by **tlearn**. Selecting an item in the **Displays** menu either hides the display, if it is currently being shown, or shows the display if it is currently hidden. Note that a display window, if it is currently shown, can be brought to the front by selecting it from the **Window** menu.

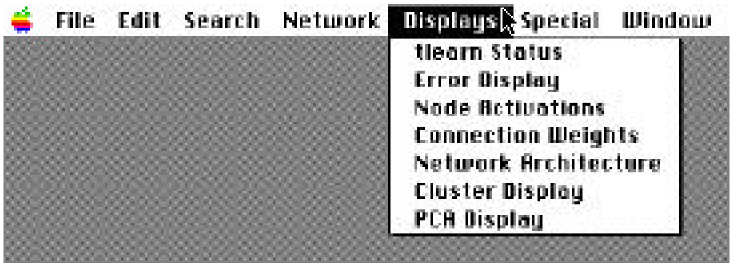


FIGURE B.15 The **Displays** menu

The Special menu

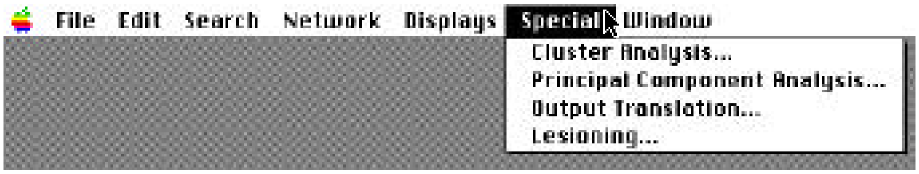


FIGURE B.16 The **Special** menu

Figure B.16 shows the **Special** menu. The **Special** menu commands act as follows:

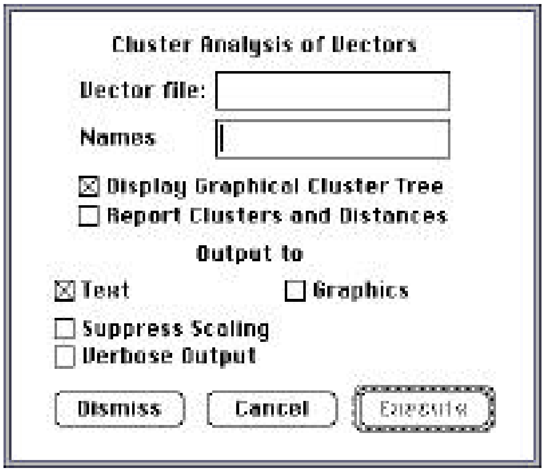
Cluster Analysis... Brings up the **Cluster Analysis** dialogue. The **Cluster Analysis** action allows a hierarchical clustering to be performed on a set of vectors specified in a file and for the results of the clustering to be displayed in the form of a cluster diagram which shows the tree of clusters that are identified by the clustering method. The **Cluster Analysis** dialogue box is described in more detail later in this section

Principal Component Analysis... Brings up the **Principal Component Analysis** dialogue. The **PCA** action allows the principal components of a set of vectors to be calculated and the projection of the vectors onto these principal components to be output or displayed graphically. The **Principal Components Analysis** dialogue box is described in more detail below.

Output Translation... Brings up the **Output Translation** dialogue box. This dialogue allows the setting of the pattern file for an output translation and a setting of the **Output Mapping Criteria**. The **Output Translation** dialogue box is described below.

Lesioning... brings up the **Lesioning** dialogue box. The **Lesioning** action allows the lesioning of a saved weight file which can selectively remove a random proportion of a set of the network nodes or connections. The **Lesioning** dialogue box is describe below.

The **Cluster Analysis...** dialogue is shown in Figure B.17. The items



Cluster Analysis of Vectors

Vector file:

Names

☒ Display Graphical Cluster Tree

□ Report Clusters and Distances

Output to

☒ Text □ Graphics

□ Suppress Scaling

□ Verbose Output

FIGURE B.17 **Cluster Analysis** dialogue box

in the dialogue are:

Vector file: Specifies the file which contains the data vectors upon which the clustering is to be done.The file name can be entered by typing the file name directly, or by double-clicking on the entry box and choosing the file name from a file selection box.

Names Specifies a file of names to be associated with the data vectors. These names are used in the cluster diagram that is displayed. The file name can be entered by typing the file name directly, or by double-clicking on the entry box and choosing the file name from a file selection box.

Display Graphical Cluster Tree Toggles whether the cluster tree is displayed or not.

Report Clusters and Distances Toggles whether the clusters and distances are reported in the text output and whether the distance values are reported in the graphics output.

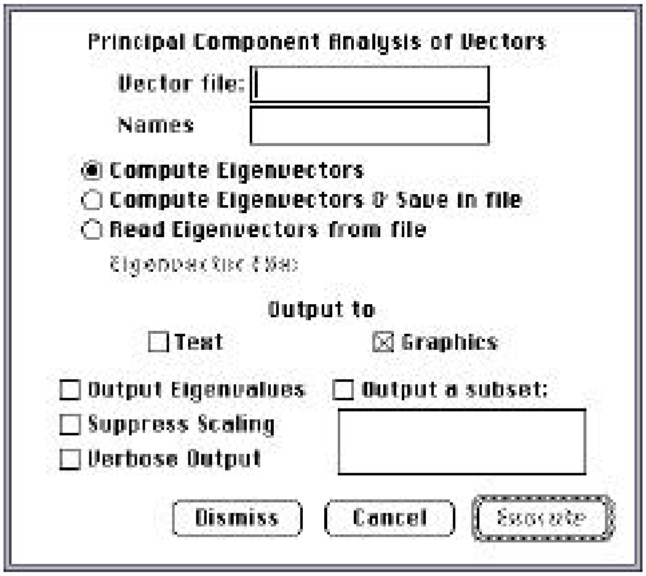
Output to Text/Graphics These check boxes specify whether the cluster analysis output is sent to the **Output** text window or the **Cluster Display** or both.

Suppress Scaling This check box suppressing scaling in the cluster analysis.

Verbose Output This check box toggles verbose output for the text output.

Once the settings for the **Cluster Analysis** have been set, then the cluster analysis can be performed. Either the text **Output** window or the **Cluster Display** window or both is brought to the front and the cluster diagram (and other text output, if specified) is given. An example of the use of **Cluster Analysis** is given in Chapter 6.

The **Principal Component Analysis...** dialogue is shown in Figure B.18. The items in the **PCA** dialogue are:



Principal Component Analysis of Vectors

Vector file:

Names

● Compute Eigenvectors
○ Compute Eigenvectors & Save in file
○ Read Eigenvectors from file

Eigenvector file:

Output to

□ Text ☒ Graphics

□ Output Eigenvalues □ Output a subset:
□ Suppress Scaling
□ Verbose Output

FIGURE B.18 **Principal Component Analysis** dialogue box

Vector file: Specifies the file which contains the data vectors upon which the principal components analysis is to be done. The file name can be entered by typing the file name directly, or by double-clicking on the entry box and choosing the file name from a file selection box.

Names Specifies a file of names to be associated with the data vectors. These names are used in the projection of the vectors onto their principal components. The file name can be entered by typing the file name directly, or by double-clicking on the entry box and choosing the file name from a file selection box.

Compute Eigenvectors/ Compute Eigenvectors & Save in file/Read Eigenvectors from file These radio buttons specify whether the eigenvectors of the analysis are computed; computed and saved in a file; or, not computed but read from a supplied file.

Eigenvector file Specifies the file where the eigenvectors are saved or read.

Output to Text/ Graphics Specify whether the principal component analysis output is sent to the output text window, or the **PCA Display**, or both.

Output Eigenvalues Toggles whether the eigenvalues are printed in the text output.

Suppress Scaling Toggles whether scaling is suppressed in the principal components analysis.

Verbose Output Toggles **Verbose Output** for the text output.

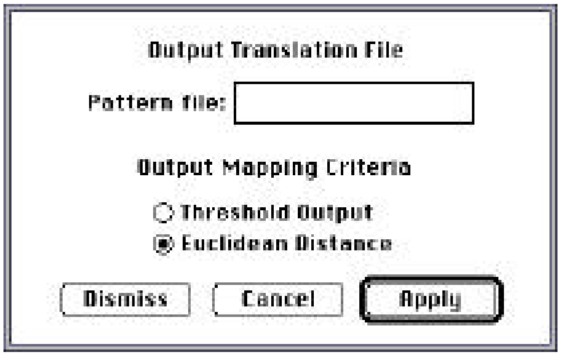
Output a subset Toggles whether only a set of the principal components are output or displayed. For example, if the user wants to display the projection of the vectors onto their second, third and fourth principal components, then the text "**2, 3, 4**" can be entered in the text box here and the **Output a subset**: check box can be clicked so the **PCA display** will show the projection onto principal components 2, 3, and 4.

Output Translation... brings up the **Output Translation** dialogue which is shown in Figure B.19. This dialogue allows the setting of the pattern file for an **Output Translation** and a setting of the **Output Mapping Criteria**. The items in the **Output Translation** dialogue are described here.

Pattern file Specifies the **Output Translation** definition file. The format of the **Output Translation** file is given below. An example **Output Translation** file is described in Chapter 11.

Threshold Output/ Euclidean Distance These radio buttons specify the **Output Mapping Criteria**. In **Threshold Output** mode, when the **Output Translation** is applied, a threshold function is applied to the elements of the output vector converting the values to 0 and 1. If the converted output vector is not present in the **Output Translation** mapping then the **Output Translation** includes a question mark character . In **Euclidean Distance** mode, no function is applied to the output vector. Instead the **Output Translation** is determined by the closest vectors according to Euclidean distance in the **Output Translation** mapping.

The **Output Translation** utility allows the specification of arbitrary translations of 0/1 vectors to text. The **Output Translation** def-



Output Translation File

Pattern file:

Output Mapping Criteria

○ Threshold Output

● Euclidean Distance

FIGURE B.19 **Output Translation** dialogue box

inition file allows the user to specify the splitting up of the output vector, permitting each part of the output vector to be assigned to different mappings, which are also specified in the file.

The format of the **Output Translation** definition file is as follows: The file begins with a **MAPPINGS:** section which specifies how the output vector is split up and by which mappings the parts of the output vector are translated. The format of lines in the **MAPPINGS:** section is as follows:

```
<node_list> from <MAPPING_NAME>
```

where **<node_list>** is a specification of a contiguous set of outputs, e.g., **1-4**; **<MAPPING_NAME>** is any unique name for a mapping which will be specified in the file.

An example **MAPPINGS:** section of an **Output Translation** definition would be:

```
MAPPINGS:
1-6 from PHONEMES
7-12 from PHONEMES
13-18 from PHONEMES
19-20 from SUFFIXES
```

Here a mapping **PHONEMES** is to be used for nodes 1 to 6, 7 to 12 and 13 to 18 and a mapping **SUFFIXES** is to be used for nodes 19 to 20.

Following the **MAPPINGS:** section come each of the mappings for the **Output Translation**. Each mapping begins with a line which contains its name followed by a colon (“**`:`**”) then any number of lines to specify the vectors to be mapped which take the following format:

```
<Label> <Vector>
```

An example mapping definition would be:

```
SUFFIXES:
W 0 0
X 1 0
Y 0 1
Z 1 1
```

The **Lesioning** dialogue box is shown in Figure B.20.The items in the **Lesioning** dialogue are:

Weight File: Specifies the weight file on which the lesioning is to be performed.

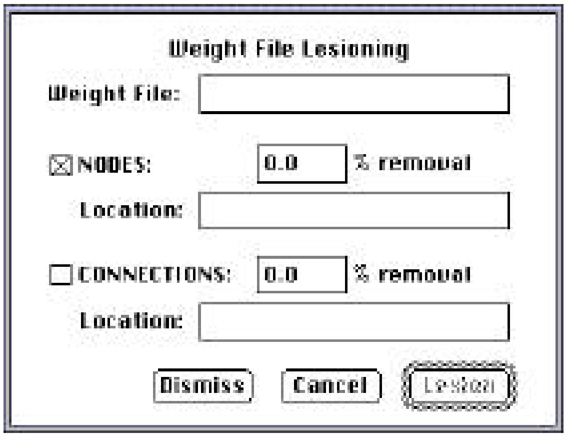
NODES: Specifies whether any nodes are to be lesioned. The **Location:** entry box is used to specify the nodes that are to be lesioned. Any list of node numbers can be entered in the format of node lists that is used in the **.cf** file. If no list of nodes is given then all the nodes are assumed to be chosen. The **% removal** entry box specifies the proportion of the nodes from the node list that are to be randomly chosen to be lesioned.

CONNECTIONS: Specifies whether any connections are to be lesioned. The **Location:** entry box is used to specify the connections that are to be lesioned. A comma separated list of connections specifications in the format:

```
<node-list> from <node-list>
```

as used in the **CONNECTIONS:** section of a **.cf** file can be entered here. If no list of connections is given then all connections are assumed to be chosen. The **% removal** entry box specifies the proportion of the connections that are to be randomly chosen to be lesioned.

When the **Lesioning** settings have been completed and the **Lesion** button is pressed then the weights from the specified weights file are read in. If the **NODES:** check box was set, then the nodes to be lesioned are randomly chosen and the chosen nodes are removed from the network. This means that the nodes to be lesioned have all connections (both input and output connections) set to zero. If the **CONNECTIONS:** check box was set, then the connections to be lesioned are randomly chosen and the chosen connections are set to zero. The



Weight File Lesioning

Weight File:

☒ NODES: 0.0 % removal

Location:

□ CONNECTIONS: 0.0 % removal

Location:

FIGURE B.20 **`Lesioning`** dialogue box

newly lesioned set of weights is displayed in a new window entitled **`<name>.lesion.wts`**. This window is an unsaved text window which can be saved and used in subsequent testing or training.

The Window menu

The **`Window`** menu lists the current windows that are being displayed by **`tlearn`** and if a window in the **`Window`** menu is chosen it is selected as the current window and brought in front of all the other windows.

Dialogue reference

Most of the dialogues used in **`tlearn`** have been discussed in the corresponding menu item that relates to their use. There remains only the **`Training Options`** and **`Testing Options`** dialogues to be discussed in this section. Before these dialogues are discussed some general notes on the dialogues used in **`tlearn`** are required.

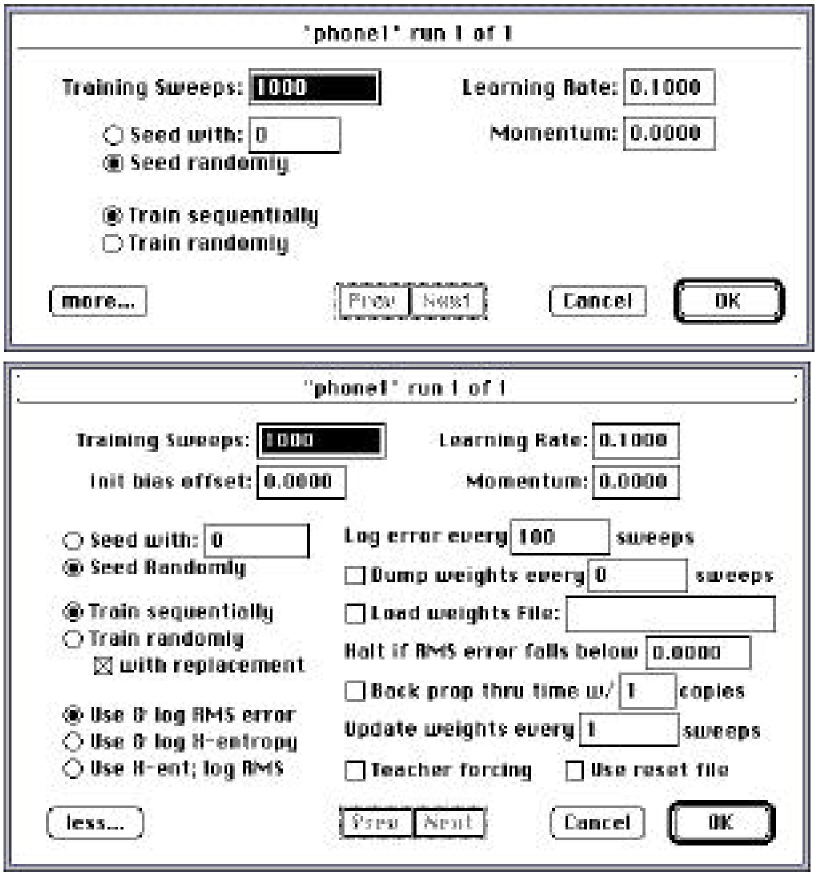
For any entry box on a dialogue that refers to a filename, the following special action is available. If the user double-clicks on the entry box associated with the filename, then a file selection box appears which allows the selection of the appropriate file, or the entry of the (possibly new) file name. This action also ensures that the file that is selected or to be saved is in the correct folder on the file system.

All dialogues have **OK** or **Action** buttons and **Cancel** buttons. Generally the **OK** or action button is the default button and can be selected by pressing Return or Enter. A dialogue can be cancelled by pressing the escape key. When a dialogue is cancelled any of the changes made to the dialogue are also cancelled and the dialogue's items revert to the state they had when the dialogue was brought up. Some of the dialogues have **Dismiss** buttons which allow the dialogue to be dismissed, and the action associated with the dialogue not to be performed, but instead of the dialogue item values reverting to their previous state, the current state of the dialogue items is kept rather than cancelled.

The Training Options dialogue

There are two versions of the **Training Options** dialogue box: small and large. These are shown in Figure B.21. The small **Training Options** dialogue is displayed by default for new projects. The small dialogue allows the modification of a subset of the training options, while the large dialogue gives the user access to all the options that are associated with training. The user can switch between the two dialogues by clicking on the **more...** and **less...** buttons at the bottom left of the dialogues.

The **Training Options** dialogue gives an interface to all of the parameters for all of the training runs performed in the current project. The parameters for different training runs are accessed via the **Prev/Next** buttons at the bottom of the dialogue. For a new project, these buttons are not highlighted. The number of the run and the total number of saved training parameters is given at the top of the dialogue. If the dialogue is showing a set of parameters other than that associated with the latest training run, then a **Remove run** button appears which allows the user to delete previous training run parameters. If the user wants to distinguish a specific training run by giving it a name, then this can be done by double-clicking the top part of the dialogue and the name of the training run can be entered.



"phone1" run 1 of 1

Training Sweeps: 1000

Learning Rate: 0.1000

○ Seed with: 0

● Seed randomly

Momentum: 0.0000

● Train sequentially

○ Train randomly

"phone1" run 1 of 1

Training Sweeps: 1000

Learning Rate: 0.1000

Init bias offset: 0.0000

Momentum: 0.0000

○ Seed with: 0

● Seed Randomly

● Train sequentially

○ Train randomly

☒ with replacement

● Use & log RMS error

○ Use & log X-entropy

○ Use X-ent; log RMS

Log error every 100 sweeps

□ Dump weights every 0 sweeps

□ Load weights File:

Halt if RMS error falls below 0.0000

□ Back prop thru time w/ 1 copies

Update weights every 1 sweeps

□ Teacher forcing

□ Use reset file

FIGURE B.21 **Training Options** dialogue boxes

A description of each of the training options is given in the following.

Training Sweeps The number of training sweeps for the training run.

Learning Rate The value of the learning rate for backpropag ation. This value is limited between 0.0 and 10.0.

Momentum The value of momentum for backpropg ation. Momentum is limited between 0.0 and 1.0.

Seed with:/
Seed randomly

These radio buttons allow the user to specify the initial random seed value (**Seed with:**) or to allow the random seed itself to be chosen randomly. For training runs which are seeded randomly the random seed which was used is displayed in the text box next to **Seed with:**. The random number generator is used for generating the initial random weights for the network and for determining the training data presentation order if the data are to be presented randomly.

Train sequentially/
Train randomly

These radio buttons specify the training data presentation order. If **Train sequentially** is checked, then the training data are presented in the order that appears in the **.data/.teach** files. If **Train randomly** is checked then the training data are presented in a random order. In the large **Training Options** dialogue, the **Train randomly** radio button has an extra check box associated with it, namely **with replacement**. If the **with replacement** box is checked then the training data are presented randomly with replacement, which means that for each subsequent presentation of a training pattern, the next pattern to be presented is chosen randomly from all of the training patterns. When the **with replacement** box is not checked then the first pattern to be presented is chosen randomly from the training patterns but is not replaced; so the next pattern is chosen randomly from the remaining training patterns and is also not replaced. This random choosing continues until there are no more patterns to choose from, at which point all the training patterns are put “back in the hat” to be chosen from again. **Train**ing **randomly** without replacement ensures that all of the training patterns are seen at least once each epoch.

Init bias offset

Allows the setting of an initial offset for the connections from the bias (node 0) to any nodes in the network. This offset only takes effect when the initial random weights of the network are calculated, where it is added to all the bias connections.

Use & log RMS error
Use & log X-entropy
Use X-ent; log RMS

These radio buttons give settings for the error function that is used by backpropagation and the error function that is displayed in the error display. Using and logging Root Mean Squared (RMS) error is the default setting. **RMS error** and **Cross entropy** (X-entropy) are often similar in their effects. Cross-entropy is explained in Chapter 9, page 166.

Log error every .. sweeps

Specifies the sweep interval at which error values are calculated and displayed on the error display.

Dump weights every .. sweeps

Specifies the sweep interval at which weight files can be saved.

Load weights File

Allows a saved weight file to be used instead of generating initial random weights.

Halt if RMS error falls below ..

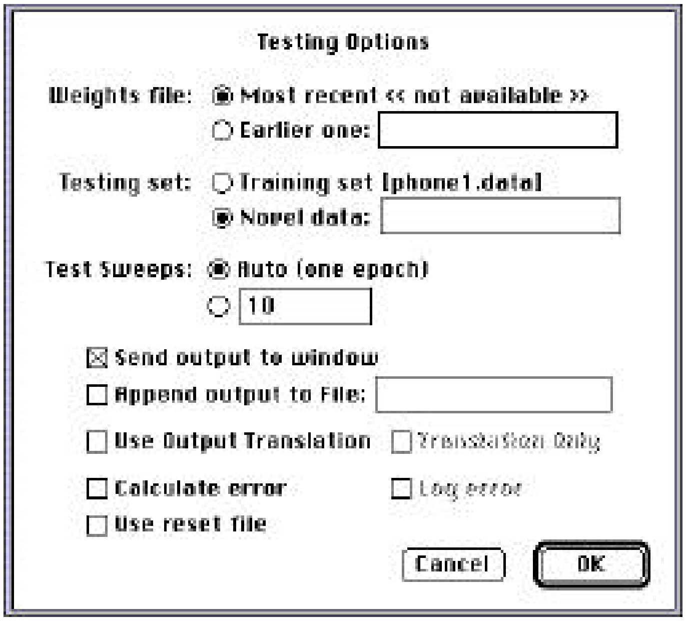
Specifies the error criterion for which training stops.

Back prop thru time w/ .. copies This option is used for training of recurrent networks using the Backpropagation Through Time (BPTT) procedure. Each unit in the network is represented by *n* multiple copies. Each of the *n* copies records that unit's activation at time *n*. Thus, propagating error back through these multiple copies is equivalent to propagating the error back in time. This allows the network to relate error information which comes in at a later time to network states earlier in time.

Update weights every .. sweeps Specifies the sweep interval between weight updates. Online learning occurs when weight updates occur at every sweep. Batch learning occurs when the training presentation is sequential and the update interval is equal to the number of training patterns.

Teacher forcing If set, when output units are fed back (as in Jordan, 1986 networks), the teacher pattern (target output) will be fed back instead of the actual output. This speeds learning.

Use reset file Causes the use of a **.reset** file during training. A **.reset** file is used for simple recurrent networks to reset the context unit activations.



Testing Options

Weights file: ● Most recent << not available >>
○ Earlier one:

Testing set: ○ Training set [phone1.data]
● Novel data:

Test Sweeps: ● Auto (one epoch)
○ 10

☒ Send output to window
□ Append output to File:
□ Use Output Translation □ Translation Only
□ Calculate error □ Log error
□ Use reset file

FIGURE B.22 The **Testing Options** dialogue box

The Testing Options dialogue

The **Testing Options** dialogue box is shown in Figure B.22 It is used to change options related to the testing actions (**Verify the network has learned**, **Probe selected nodes**). A description of the options on the dialogue is given in the following.

Weights file These radio buttons allow the selection of the **Most recent** set of network weights or an **Earlier one:** to be used for testing. The name of the desired weight file can be typed into the text entry box next to the **Earlier one:** choice, or, if the user double-clicks on the text box, a file selection box can be used to select the appropriate file.

Testing set These radio buttons select the data (and teach) files to be used for testing. Either the **Training set** or a **Novel data:** set can be chosen. As for the earlier weight file selection above, the name of the data set can be typed into the text entry box, or the user can double-click on the text box, and a file selection box can be used to select the **.data** file.

Test Sweeps The testing action can be set to run for one epoch (that is, a number of sweeps equal to the number of training patterns in the data file) or a specified number of test sweeps entered into the text entry box.

Send output to window/Append output to File Specify where the output of the testing actions (and also the text output of clustering and **PCA** actions) is sent. The output can be sent to either an **Output** window or appended to a file, specified in the text entry box next to the **Append output to File:** check box, or both.

Use Output Translation Causes the **Verify** action to use and print the **Output Translation** defined in the **Output Translation** dialogue. The **Translation Only** check box causes the **Verify** action to only output the translation rather than the translation and the output unit activations.

Calculate error Causes the testing actions to produce network error calculations which appear in the error display . For the calculation of error a **.teach** file is required. For the project's normal training set defined by the **.data** file, there is already a corresponding **.teach** file, but for a novel data set, if an error calculation is required, then a novel **.teach** file must also be present for the testing actions to produce a valid error calculation.

Log error Causes the error calculation if specified by the **Calculate error** check box to be written to a file. The name of the file is **<project_name>.err**

Use reset file This check box causes the use of a **.reset** file during testing actions. A **.reset** file is mostly used for simple recurrent networks which require the activations of context units to be reset.

Command key and shortcut reference

Menu command keys

Menu keys can be seen in the appropriate menus where, if the menu item has an associated command key, it is shown at the right of the menu item.

Windows	Mac	Action
Ctrl+A	**⌘-A**	Edit/Select All
Ctrl+C	**⌘-C**	Edit/Copy
Ctrl+E	**⌘-E**	Search/Enter Selection
Ctrl+F	**⌘-F**	Search/Find…
F3	**⌘-G**	Search/Find Again
F4	**⌘-H**	Search/Replace & Find Again
Ctrl+J	**⌘-J**	Network/Testing Options…
Ctrl+K	**⌘-K**	Network/Verify the network has learned
Ctrl+L	**⌘-L**	Network/Probe selected nodes
Ctrl+N	**⌘-N**	File/New
Ctrl+O	**⌘-O**	File/Open…
Ctrl+P	**⌘-P**	File/Print…
Alt-F-x	**⌘-Q**	File/Quit
Ctrl+R	**⌘-R**	Network/Resume training
Ctrl+S	**⌘-S**	File/Save
Ctrl+T	**⌘-T**	Network/Train the network
Ctrl+U	**⌘-U**	Network/Abort
Ctrl+V	**⌘-V**	Edit/Paste
Alt-F-C	**⌘-W**	File/Close
Ctrl+X	**⌘-X**	Edit/Cut
Ctrl+Y	**⌘-Y**	Network/Training Options…
Ctrl+Z	**⌘-Z**	Edit/Undo
Ctrl+H	**⌘-=**	Search/Replace
Ctrl+G	**⌘-`**	Search/Go To Line…

Editor quick keys

Shift-arrow-keys allows the selection of text (or, if text is already selected, the extension or reduction of the text selection) to be done from the keyboard rather than with a click-drag action of the mouse.

Ctrl-left, **Ctrl-right**	moves the cursor to the next or previous word boundary
Ctrl-up, **Ctrl-down**	moves the cursor up/down a screenful of text
⌘-left, **⌘-right**	moves the cursor to the beginning/end of a line
⌘-up, **⌘-down**	moves the cursor to the beginning/end of a document

Troubleshooting

This section lists common problems and solutions.

Problem: Can’t lesion weights file. **Lesion** command is disabled.
Solution: Before the lesioning action can be performed, a project must be specified. The reason for this is that the network configuration must be known for lesioning to be performed.

Problem: I have specified an **Output Translation** file in the **Special** menu, but when I probe or verify the output translation doesn’t appear.
Solution: In the **Testing Options** dialogue the **Use Output Translation** check box must be set for the **Output Translation** to be used. If you only want the **Output Translation** and not the output node activations, then the **Output Translation Only** box should be checked.

Problem: I want to display principal components other than the first two or three. How do I do this?

Solution: Specify a subset in the appropriate part of the **Principal Components Analysis** dialogue box.

Problem: Why doesn't the project change even though I close the old **.cf**, **.data**, **.teach** files and open the new files?

Solution: To use a different set of project files you need to create or open the project using the **New Project…/Open Project…** commands in the **Network** menu. The **Status** Display shows the name of the current project.

Problem: The **Translate…** action translates too much and translates things I don't want translated.

Solution: The order of things in the translate file is important. Specifically, care must be taken that a later translation doesn't inadvertently retranslate a previous translation. This could occur if letters were translated to numerical vectors, and then a later translation action translated digits to letters. Unless this was a desired effect, this double translation will possibly cause havoc to the translation as intended. One piece of advice here is to ensure that no strings on the left of a translation rule appear on the right of a translation rule.

Problem: When I run **tlearn** on several projects, I'm told that there is not enough memory (for **data** or **teach** files, for example).

Solution: After running consecutive projects, **tlearn** may fail to release all the memory associated with the files it has used. Quitting **tlearn** and restarting should solve the problem.

APPENDIX A

Installation procedures

This appendix describes the contents of the Macintosh and Windows **95 tlearn** distribution disk and how to install **tlearn. tlearn** is a connectionist modelling simulation package that provides a graphical user interface (GUI) to backpropagation neural networks. There are versions of **tlearn** for Macintosh and Windows 95 and XWindows/Unix.

Mactintosh installation

The Macintosh distribution disk contains the Macintosh **tlearn** (version 1.0.1) executable and associated exercise files. The tlearn executable file is a FAT executable which means that it can be run on both PowerPC and 680x0 Macintosh machines. The disk also provides a set of Chapter folders which contain the project files corresponding to the exercises in this book.

To install **tlearn,** copy the compressed file called **mac_tlearn** to the folder on your Macintosh where you want the application to reside. This file is a self-extracting-archive (SAE) which means that you only need to double-click on the file to begin the installation. (This installer was created usingstuffit InstallerMaker. © 1990-96 Aladdin Systems, Inc.) You will be asked where you want to place the programme and assciated exercise files. The executable is called **tlearn** and can be launched without any other changes to the machine. Note that for simulation of large networks, it may be necessary to increase the memory of application through the **Get Info...** window available within the finder.

Windows 95 installation

The Windows 95, distribution disk contains the Windows 95 tlearn (version 1.0.1) executable and associated exercise files. To install tlearn, copy the compressed file called win_tlearn(.exe) (using Windows Explorer) to the folder on your PC where you want the application to reside. This file is a self-extracting-archive (SAE) which means that you only need to double-click on the file to begin the installation. The executable is called tlearn and can be launched without any other changes to the machine. The disk also provides a set of Chapter folders which contain the project files corresponding to exercises in this book. We do not recommend that you use tlearn running under Windows 3.x (even with Win32s extensions installed).

Updates and bug reports

We do not guarantee that the software provided with this book is free of bugs. In fact, we guarantee that if you try bard enough you will find some situations where tlearn will break! Please tell us about any software problems you experience by emailing us at
innate@crl.ucsd.edu.
We may not respond to your queries immediately but we will fix the bugs as time permits. You can obtain thelatest versions of tlearn via anonymous ftp at crl . ucsd. edu (in the pub directory) or on the world wide web at http: //crl .ucsd.edu/innate. This site is situated in San Diego. Users from the Old World may find it easier to obtain the laeest versions of tlearn via anonymous ftp at f tp . psych. ox. ac. uk (also in the pub directory). This site is situated in Oxford, UK.