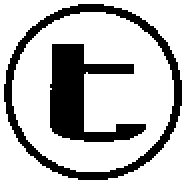

S))))))))))))))))))))))))))))))))))

23A Durham Drive
Amherst, NY 14228

Galileo*ORESME:

User Manual

Rev. June, 1993

S))))))))))))))))))Q

ORESME COPYRIGHT 1990 BY JOSEPH WOELFEL

PLEASE DON'T LOSE THIS PAGE. IT CONTAINS THE REGISTRATION NUMBER YOU WILL NEED TO UPGRADE TO LATER RELEASES OF CATPAC.

Your Name _____
Your Registration Number _____

Version 3.0

The Galileo Company

IMPORTANT!
PLEASE READ CAREFULLY BEFORE USING THE SOFTWARE.

NOTIFICATION OF COPYRIGHT

THIS SOFTWARE IS A PROPRIETARY PRODUCT OF The Galileo Company AND IS PROTECTED BY COPYRIGHT LAWS AND INTERNATIONAL TREATY. YOU MAY MAKE A REASONABLE NUMBER OF COPIES OF THIS PROGRAM FOR BACKUP PURPOSES, AND YOU MAY COPY THE SOFTWARE TO THE HARD DISK OF A SINGLE COMPUTING PLATFORM OF THE TYPE SPECIFIED IN YOUR LICENSE.

YOU ARE PROHIBITED FROM MAKING ANY OTHER COPIES OF THE SOFTWARE FOR ANY OTHER PURPOSE BY COPYRIGHT LAWS. YOU MAY MAKE ONE COPY OF THE WRITTEN MATERIALS ACCOMPANYING THIS SOFTWARE FOR ARCHIVAL PURPOSES.

The Galileo Company

PLEASE READ THIS LICENSE AGREEMENT BEFORE USING THE SOFTWARE. THIS AGREEMENT IS A LEGAL CONTRACT BETWEEN YOU AND The Galileo Company GOVERNING YOUR USE OF THIS SOFTWARE. USING THIS SOFTWARE INDICATES YOUR ACCEPTANCE OF THIS AGREEMENT. IF YOU DO NOT WISH TO ACCEPT THE TERMS OF THIS AGREEMENT, PLEASE RETURN THE UNOPENED SOFTWARE PROMPTLY TO The Galileo Company. IF YOU HAVE ANY QUESTIONS ABOUT THIS AGREEMENT, PLEASE CONTACT The Galileo Company, 615 E. ELEVEN MILE ROAD, SUITE 34, ROYAL OAK, MI 48067. PHONE 313.398.6236.

TERMS OF LICENSE

THIS IS AN EXPERIMENTAL PROGRAM. WHILE The Galileo Company CERTIFIES THAT THE HIGHEST STANDARDS OF DILIGENCE AND SCIENTIFIC INTEGRITY HAVE BEEN APPLIED TO THE DEVELOPMENT OF THIS SOFTWARE, BY ACCEPTING THIS LICENSE YOU AGREE THAT THIS IS EXPERIMENTAL SOFTWARE AT THE CUTTING EDGE OF SCIENTIFIC PROGRESS.

NOT AS MUCH IS KNOWN ABOUT THE PERFORMANCE OF NEURAL NETWORK TECHNOLOGY AS IS KNOWN ABOUT TRADITIONAL COMPUTER SOFTWARE. YOU AS THE END USER AGREE THAT REASONABLE AND PRUDENT CAUTION ABOUT THE APPLICATION OF RESULTS FROM THIS SOFTWARE IS APPROPRIATE, AND The Galileo Company AGREES TO SHARE WITH YOU (THE LICENSEE) RELIABLE ESTIMATES OF THE OPERATING PARAMETERS OF THE SOFTWARE INSOFAR AS THEY ARE KNOWN BY TERRA.

The Galileo Company GRANTS YOU THE RIGHT TO USE ONE COPY OF THE SOFTWARE ON A SINGLE-USER COMPUTER. EACH WORKSTATION OR TERMINAL ON A MULTI-USER COMPUTER SYSTEM OR LOCAL AREA NETWORK MUST BE LICENSED SEPARATELY BY The Galileo Company.

YOU MAY NOT SUBLICENSE, RENT OR LEASE THE SOFTWARE TO ANY OTHER PARTY.

YOU MAY MAKE REASONABLE BACKUP OR ARCHIVAL COPIES IF THE SOFTWARE, BUT YOU MAY NOT DISASSEMBLE, DECOMPILE, COPY, TRANSFER, REVERSE ENGINEER OR OTHERWISE USE THE SOFTWARE EXCEPT AS STATED IN THIS AGREEMENT.

LIMITED WARRANTY

The Galileo Company will replace defective diskettes that are returned within 90 days of the original purchase date without charge. The Galileo Company warrants that the software will perform substantially as stated in the accompanying written materials. If you should discover any significant defect and report it to The Galileo Company within 90 days of purchase, and Terra is unable to correct it within 90 days of receipt of your report of the defect, you may return the software and Terra will refund the price of purchase.

SUCH WARRANTIES ARE IN LIEU OF OTHER WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE SOFTWARE AND THE ACCOMPANYING WRITTEN MATERIALS. IN NO EVENT WILL The Galileo Company BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY LOSS OF PROFITS, LOST SAVINGS, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF YOUR USE OF OR INABILITY TO USE THE PROGRAM, EVEN IF The Galileo Company OR AN AUTHORIZED TERRA REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. The Galileo

Company WILL NOT BE LIABLE FOR ANY SUCH CLAIM BY ANY OTHER PARTY.

This limited warranty gives you specific legal rights. Some states provide other rights, and some states do not allow limiting implied warranties or limiting liability for incidental or consequential damages. For this reason, the above limitations and/or exclusions may not apply to you. If any provision of this agreement shall be unlawful, void or for any reason unenforceable, then that provision shall be deemed separable from this agreement and shall not affect the validity and enforceability of the remaining provisions of this agreement. This agreement is governed by the laws of the State of New York.

U.S. Government Restricted Rights

The software and accompanying materials are provided with Restricted Rights. Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3)(ii) of the Rights in Technical Data and Computer Software clause at 252.277=7013. Contractor/manufacturer is The Galileo Company, 615 E. Eleven Mile Road, Suite 34, Royal Oak, MI 48067

TABLE OF CONTENTS

NEURAL NETWORKS 1
Self Organizing Neural Networks 4

ORESME 4

INSTALLING ORESME 6

RUNNING ORESME 7
Hey Boss! How many nodes? 7
Do you want to start a new problem? 7
Do you have a labels file? 8
Where do you want to put the labels? 8
Where should we put the data? 8
Randomizing 9
Where are the data? 9
Where are the labels 9
And where would you like the output, Air Breath? 9
Where would you like the modified weights saved? 9
Care to set any values? 10
Do you wish to set a new threshold? (10); **How about a new decay rate?** (11); **New Learning Rate?** (11); **Care to speculate on a functional form, Chiphead?** (11)
Do you need to see the labels, Chemical Brain? 12
Do you have a training file? 12
Enter concept label (Ctrl z when done) 12
Enter activation value 13
Do you want these values clamped? 13
How many cycles, hysteresis breath 13
Should I learn? 14
Analog? 14
Do you want to go again, Sack of Mostly Water? 15

CREATING A NEW PROBLEM -- AN EXAMPLE 16

OTHER INPUT 16

Appendix 1: Tools 20

NEURAL NETWORKS

The human brain is perhaps the most complicated device we know, and it is folly to believe we understand it fully. Deep questions of consciousness, coordination and control remain unsolved. But it is fair to say that fundamental understandings of how the networks of interconnected neurons in the brain store and retrieve patterns of information in principle are beginning to emerge. A natural neural network (like the brain) consists of neurons, each of which may be connected to many other neurons. (In a human brain, there are about 100 billion neurons, each of which is connected, on the average, to about a thousand other neurons.) When a neuron is stimulated, it becomes "active", and sends signals to all the other neurons to which it is connected.

Neural networks store information as patterns in the same way that a TV screen or theater marquee or electronic scoreboard does: By activating some of the dots or light bulbs and leaving others off, any pattern can be displayed. (Researchers have actually identified more than a dozen maps of the visual field in the human brain.) But because the neurons in a neural network are connected to each other, the neural network can do more than simply display patterns of information: it can store and retrieve those patterns, and recognize patterns it has stored even if they are distorted or incomplete.

Although the actual functioning of a neural network like the human brain can be extremely complicated, in principle the way a neural network works is very simple and easy to understand. A neural network learns by connecting together the neurons which represent any particular pattern. Since they are connected together, *when some of them are activated, they spread their activation to the others connected to them, which turns on the rest of the pattern*. The neurons in the pattern may also be *negatively* connected to neurons not in the pattern, so that when the neurons in the pattern are active, they tend to turn off all those neurons not in the pattern. Thus, when a network sees part of a pattern, it can recall the rest of the pattern, even in spite of incomplete or erroneous information, as long as enough of the pattern is there to activate the rest.

Figure 1 shows a network consisting of six nodes representing the words "Cat", "Dog", "Barks", "Howls", "Meows", and "Purrs". Each of the nodes may take on the value "0" (off), or "1" (on).

The nodes are

Input = "Meows"

		Cat	Dog	Barks	Howls	Meows	Purrs
	Cat		-.8	-.9	.2	.8	.9
	Dog	-.8		.9	.3	-.8	-.7
	Barks	-.9	.9		.5	-.3	-.9
	Howls	.2	.3	.5		-.2	-.1
+1	Meows	.8	-.8	-.3	-.2		.8
	Purrs	.9	-.7	-.9	-.1	.8	
		on	off	off	off	on	on

FIGURE 1

connected to each other by weights which represent their relative "closeness" in the network.[1] They communicate with each other by a simple linear threshold rule: the signal sent from any node *i* to any node *j* equals the product of the activation value of *i* and strength of the connection between *i* and *j*. Thus the total signal received by any node *j* will be the sum of the signals received from all the other nodes, or

$$net_j = \sum_{i=1}^{N} w_{ij} a_i$$

The way a node responds to the set of signals it receives is determined by its activation function; in this case we adopt the rule that the node will be activated if the sum of its input signals is positive; otherwise it will be turned off, or

+1 if x > 0
a_i = unchanged if x = 0
-1 if x < 0

Following this rule, we assume the network receives the input "Meows" from its environment (i.e., the node which represents "Meows" has been activated). This sets the activation value of "Meows" at +1, and the activation values of the other nodes at 0. Multiplying the weights in each column by the activation values of the corresponding rows, then summing within each column shows that the activation of the node "Meows" will "spread" to the nodes "Cat" and "Purrs", setting their activations to 1, but will leave the nodes "Dog", "Barks" and "Howls" off. Figure 2 shows that

Input = "Howls"

		Cat	Dog	Barks	Howls	Meows	Purrs
	Cat		-.8	-.9	.2	.8	.9
	Dog	-.8		.9	.3	-.8	-.7
	Barks	-.9	.9		.5	-.3	-.9
+1	Howls	.2	.3	.5		-.2	-.1
	Meows	.8	-.8	-.3	-.2		.8
	Purrs	.9	-.7	-.9	-.1	.8	
		on	on	on	on	off	off

FIGURE 2

[1] In the present example, the weights are essentially the correlations between frequencies of occurrence of the various words. Thus "Meows" and "Cat" tend to "go together", with a weight of .8, while "Meow" and "Dog" have a negative coefficient of -.8.

activating the node "Howls", will also activate the nodes "Cat", "Dog" and "Barks". Figure 3 shows that activating both the nodes "Barks" and "Howls" will also activate "Dog", but will leave "Cat", "Meows" and "Purrs" off.

Input = "Howls" and "Barks"

		Cat	Dog	Barks	Howls	Meows	Purrs	
	Cat		-.8	-.9	.2	.8	.9	.1
	Dog	-.8		.9	.3	-.8	-.7	.9
+1	Barks	-.9	.9		.5	-.3	-.1	1
+1	Howls	.2	.3	.5		-.2	.1	1
	Meows	.8	-.8	-.3	-.2		.8	-.8
	Purrs	.9	-.7	-.9	-.1	.8		-.1
		off	on	on	on	off	off	

FIGURE 3

This example shows clearly that communication among the nodes of the network produces an apparently qualitative change in the pattern recognition and storage capabilities of the network. When the nodes do not communicate, the network can represent a pattern of virtually any complexity when activated directly by the environment, but the complete input is required to produce the complete pattern. When the nodes communicate, however, the complete pattern can be produced with only a partial input. When a sufficient subset of the nodes in a stored pattern is activated, the activation of those nodes will spread through the links and in turn activate the rest of the nodes in the pattern.

It is worth emphasizing the fundamental role communication as it has been defined here plays in this process. A pattern is stored by "connecting" its elements together. Things that "go together" are "close". Nodes or elements in turn *communicate* their activation values to other nodes in proportion to their closeness in the communication network. If a node is "on", it will tend to transmit that "on-ness" to other nodes through the links between them, so that the "on-ness" will spread to other nodes which represent the other elements in the pattern. Similarly, if a node is "off", it will tend to communicate its "off-ness" to other nodes through the links between them. *The entire pattern is encoded in the pattern of communication among the nodes as connections or weights, and can be recovered by the activation of any suitable subset of nodes.*

Self Organizing Neural Networks

All of a network's "memory" is stored in the weights or connections among the neurons. A network learns by setting these weights. One way self-organizing neural networks (often called "unsupervised" networks) learn patterns is by a simple Pavlovian conditioning rule: When two or more neurons are simultaneously active, the connection among them is strengthened. This means, quite simply, that neurons that have behaved similarly in the past are likely to behave similarly in the future. Self-organizing networks receive information in the form of patterns, which they learn to recognize, and which they can recall later. Self-organizing networks develop an internal representation

of the information to which they have been exposed. They are useful because one can enter fragments of a pattern the network has learned, even in somewhat distorted form, and the network can recover the original pattern.

ORESME

ORESME is a self-organizing neural network which simulates the cognitive processes of individuals or groups of people, such as markets or market segments. ORESME represents objects, products, attributes, people or any other concept as neurons in a network. Mentioning one or more of these objects (as one would in an advertisement) activates the neurons which represent those objects. These activated neurons in turn activate those other neurons to which they are closely connected, while turning off those neurons to which they are negatively connected. This *interactive activation and competition network* thus simulates the process by which one or more ideas stimulates still other ideas.

Figure 4 illustrates an example that shows how **ORESME** might be used to test a particular advertisement for an automotive vehicle.

Threshold = .000 Restoring Force = .250 Damping Factor = .000

Concept	Cycles X 1									
	1	2	3	4	5	6	7	8	9	10
SPORTY LOOKING	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FUN TO DRIVE	1.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FAMILY CAR	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD VALUE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
PRACTICAL	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
AFFORDABLE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
EXCITING	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
APPEALS TO OLDER PEO	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
LUXURIOUS	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
RELIABLE	1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
HONDA ACCORD	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
SUBARU LEGACY	1.0	1.0	1.0	1.0	.0	.0	.0	.0	.0	.0
FORD TEMPO	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
TOYOTA CAMRY	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
NISSAN STANZA	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
CHRYSLER LEBARON GTS	.0	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
PONTIAC GRAND AM	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
YOURSELF	.0	1.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

An ORESME analysis indicates that advertising SUBARU LEGACY as "fun to drive" and "reliable" might increase its appeal in the short run, but might eventually result in the decision to buy a PONTIAC GRAND AM or a CHRYSLER LEBARON GTS in the longer term.

FIGURE 4

ORESME can be helpful in alerting advertisers to the potential problems which might arise from unexpected connotations of otherwise useful message strategies. **ORESME** can accept inputs from CATPAC or GALILEO, or can develop its own network interactively.

INSTALLING ORESME

- Place the diskette in the A: or B: drive.

- Type **INSTALL <diskette drive> <target drive>** and press Enter.

  For example to install the system on your C: drive with the diskette in the A: drive you would type:

  INSTALL A: C:

That's it. The install program will take care of everything.

The following directories will be created:

\GALILEO\RUNNER	Contains the executable programs
\GALILEO\HELP	Contains the help files
\GALILEO\DOC	Contains all available Galileo Documentation in WordPerfect 5.0 format
\GALILEO\DATA	Contains sample data sets
\GALILEO\TOOLS	Contains a text editor and several utility programs

RUNNING ORESME

To run **ORESME**, change directories so that you are in the RUNNER sub-directory and type **ORESME.** If you have edited your path, you need only type **ORESME**. (If you are running **ORESME** as a part of the complete Galileo system, you can select ORESME from the Galileo Menu and press **[ENTER]**.

ORESME will then ask you a series of questions to determine the type of analysis you want to perform, and you need only type the answers to set-up your run. Here's what **ORESME** will ask:

Hey Boss! How many nodes?

The basic input into **ORESME** is CATPAC output. If you are inputting the output from a CATPAC analysis, the number of nodes corresponds to the number of unique words CATPAC generated from its analysis.

Essentially, *Node* is another name for *neuron*, and **ORESME** needs to know how many neurons to create. Each neuron corresponds to one concept or word. Presently, **ORESME** can handle up to 160 neurons.

Do you want to start a new problem?

ORESME can read networks made by other programs, such as **CATPAC** or **GALILEO**, or the output from a previous (ORESME) run. **ORESME** can also create a new network on the fly. That is, you may enter a network by hand at the terminal. If you are entering output from another program (like CATPAC) type **NO** at this prompt.

If you wish to create a new network on the fly, type **YES**. If you type YES **ORESME** will ask you the following questions:

Do you have a labels file?

You can save some time if your labels are already listed in a file, one label per line. If so, just say yes and the program will ask you later for the name of that file. If you haven't done this, the program will give you the opportunity to enter them here.

Where do you want to put the labels?

Each of the neurons in the network stands for some concept or word; these words are called "labels." **ORESME** wants you to tell it the name of a file where it can store the labels you are about to give it. Just enter the path of any file where you would like to store the labels. If the file does not already exist, **ORESME** will automatically create it. (You can name this file anything you want, but at **Terra** we end all labels files names with the suffix **.LBL**.) After you've named a file to store the labels, **ORESME** will prompt you for each of them:

Please enter label 1
Please enter label 2
.
Please enter label n

Then **ORESME** will ask you:

Where should we put the data?

Once again, **ORESME** needs to know the name of a file, this time to put the network of connections or weights that it will build. This file will be in the form of a matrix of weights, where each weight represents the strength of communication between two of the neurons in the network. When you are starting a new network, these weights will initially be random numbers; later the program will give you the opportunity to output a new set of weights after **ORESME** has learned them.

Randomizing

When **ORESME** first constructs the network, it randomizes the connections among all the neurons. You don't have to respond to this; it's just informing you of this.

Where are the data?

If you answered *NO* when **ORESME** asked if you wanted to start a new problem, **ORESME** will need to know where the previously made network is stored. Answer with the complete path to the previously made weight input network (.WIN) file. (See **INPUT** below.)

Where are the labels?

In every network, the neurons represent some words or ideas. The labels (.LBL) file contains

the names of each neuron. Tell **ORESME** the exact path to the file containing the labels for this network.

And where would you like the output, Air Breath?

ORESME keeps an exact record of what appears on your screen during your conversation, and stores it on a file of your choice. You can specify the name of any file whatever, and **ORESME** will write a copy of your conversation to that file for saving or printing.

Where would you like the modified weights saved?

When **ORESME** learns, it does so by modifying its weights. Rather than changing the original weight input network (.WIN) file, **ORESME** makes a new matrix with the changed weights in it. That way you can keep **ORESME** as it was, and still have a modified matrix as well. Just tell **ORESME** the path of the file on which you'd like the modified weights saved. (At **Terra** we use the extension **.WGT** to denote a file that contains modified weights. But you can call it whatever you want.)

Care to set any values?

ORESME can simulate four different kinds of neurons, and the overall performance of **ORESME** depends on three parameters. The most generally useful neuron and some reasonable values for the three general parameters have been chosen as defaults in **ORESME**. But you can change them if you wish, and none of these neuron types or parameters are sacred, even those selected by Terra as defaults. You might well find **ORESME** performs better for some tasks with a different choice of neurons and/or default parameters. In order to change any defaults, just say yes. If you say no, you will get the defaults. If you say yes, you will be asked four questions:

Do you wish to set a new threshold?

Each neuron in **ORESME** is either turned on by you assigning it a value, or else it receives inputs from other neurons to which it is connected. These inputs are transformed by a *transfer function*. **ORESME** can use one of four transfer functions: a linear function varying between -1 and +1, a logistic function ranging between 0 and +1, a logistic function varying between -1 and +1, and a hyperbolic tangent function varying between -1 and +1.

After the inputs to any neuron have been transformed by the transfer function, they are

summed, and, if they exceed a given threshold, that neuron is activated; otherwise it remains inactive. The default threshold is 0.0, which is appropriate for three of the four transfer functions (.5 would be a more reasonable value for the logistic varying between 0 and +1.) By lowering the threshold, you make it more likely for neurons to become activated; by raising the threshold, you make it less likely for neurons to become activated.

How about a new decay rate?

When you see an object, neurons which represent that object are activated. When the object is gone, the neurons (fortunately) turn off again. (If they didn't, you'd be seeing everything you ever saw all the time.) The decay rate specifies how quickly the neurons return to their rest condition (0.0) after being activated. The default rate is .9, which means that each neuron, if not reactivated, will lose 90% of its activation each cycle. Raising the rate makes them turn off faster; lowering the rate means they are likely to stay on longer.

New Learning Rate?

When neurons behave similarly, the strength of the connection between them is strengthened. The learning rate is how much they are strengthened in each cycle. Default is .001. Increasing this rate makes **ORESME** learn faster. Faster is not always better, though, since too high of a rate can make **ORESME** oscillate back and forth as new information is read. No one knows the optimum rate, or even if there is an optimum rate, however, so feel free to experiment.

Care to speculate on a functional form, Chiphead?

This option allows you to try different transfer functions. You can choose from four: a logistic varying between 0 and +1, a logistic varying between -1 and +1, a hyperbolic tangent function varying between -1 and +1, and a linear function varying between -1 and +1. Some writers speculate that different functions are better for different kinds of task, but no one knows for sure at this time.

The default threshold is 0.0. If you choose the logistic function that varies between 0 and 1, you might want to change the threshold to .5 or thereabouts (see *Do you wish to set a new threshold?* above.) If you'd like to experiment with different transfer functions, just say yes, and **ORESME** will prompt you to select the transfer function you want.

A Chiphead is a person with an exceptional commitment to computing. If you plan to do basic research on various transfer functions, you are one.

Do you need to see the labels, Chemical Brain?

ORESME works by allowing you to turn on or off some or all of the neurons in the network, and then operates by communicating that pattern of activation throughout the network, turning other neurons on or off. Each neuron represents some idea or concept; the labels remind you of which is which. If you can remember which is which, you don't need to see the labels; if you don't, just say **YES** and **ORESME** will remind you.

Do you have a training file?

ORESME looks at words (labels) that occur together in the same "window." A window is any arbitrary set of words. You can build a training file which lists windows of words or labels, one per line, with each window separated by a -1 in columns 1 and 2 of the line following that window. **ORESME** will then read that file, learn which words "go together," and revise its understanding according to those new patterns. If you haven't made such a file, you can enter the data live and on line. If you have a prewritten training file, say "yes." If not, say "no" and you will be given the opportunity to enter the windows of labels live.

Enter concept label (Ctrl z when done)

Just enter the name of the neuron you want to activate. **ORESME** will keep on asking you for concept labels until you enter a [CONTROL] Z code, so you can turn on as many as you like.

Enter activation value

You may not only activate any neuron or neurons you wish, but you can set an activation value for each. You can enter any real number whatever, positive or negative.

Do you want these values clamped?

Clamping the value of a neuron means that you turn it on and make it stay on. Not clamping means that you assign a value to the neuron, but that value is free to change in the next cycle. Basically it's the difference between sending a message at one time, and sending the same message continuously.

How many cycles, hysteresis breath?

When words are present in the scanning window, the neurons assigned to those words are active, and the connection among all active neurons is strengthened. But the activation of any neuron travels along the pathways or connections among neurons, and can in turn activate still other neurons whose associated words may not be in the window. These neurons can in turn activate still other neurons, and so on.

In an actual (biological) neural network, these processes go on in parallel and in real time, so that the signal coming into the network is spreading at different rates of speed throughout the network, and neurons are becoming active and inactive at different times. (This process of delay is called *hysteresis*.)

In a serial computer like yours, however, this is extremely difficult to model, and so the network is updated periodically all at once. Each update is called a cycle. Letting **ORESME** cycle two or three times allows second and third order relationships among the words to be considered.

Very little cycling (or especially none at all like the concurrence model) tends to find only very superficial associations. Too much thinking, however, is not always a good thing, and **ORESME** can tend to see things as all pretty much alike if its allowed to cycle too many times. Experiment.

Should I learn?

Unlike human beings, who are always being influenced by their surroundings, **ORESME**'s learning can be turned on or off. When learning is on, the weights of the connections among the neurons are allowed to change in response to the patterns of activation that are cycling through **ORESME**. The old weights, in any case, are saved and left in their original file unchanged; the new revised weights are written out to a new file which you named earlier. (See *Where would you like the modified weights saved?*)

Analog?

ORESME operates in either digital or analog mode. In digital mode, if the inputs to a given node exceed an arbitrary threshold (see above), the node is set to +1. In analog mode, the neuron just emits the actual value of its activation. These two kinds of networks work quite differently. Experiment.

Shall I think it over 1 more time?

When **ORESME** studies word connections, it take notes of words that associated with each other, and displays them for you. After its initial analysis, if you type **YES** at this prompt, **ORESME**

will re-adjust the connection weights among words, strengthening some, weakening others, and again display the word associations it uncovered. If the network has stabilized, NO new words will appear, and none will be deleted from your original list. On the other hand, you may see that some words which were on the "fringe of association" have now been included, and/or some words that were "barely associated" have now been included. **ORESME** can do this type of "re-thinking" up to ten times.

Do you want to go again, Sack of Mostly Water?

ORESME is just asking you if you want to run through the program again. If not, it will terminate and put all your files in the places you told it.

CREATING A NEW PROBLEM -- AN EXAMPLE

Figure 5 shows an example of a new problem created using ORESME. After answering "yes" to the question *Do you want to start a new problem?*, the files PLANES.LBL and PLANES.WIN were created. In PLANES.LBL, the names of 10 World War II aircraft were listed: 6 fighters and 4 bombers. **ORESME** assigned random weights to PLANES.WIN.

As Figure 5 shows, during the first pass through **ORESME**, all the nodes representing fighter planes were turned on, while all those representing bombers were turned off. During the second pass, all the bombers were activated, while the fighters were turned off.

During both passes, learning was activated, so the weights connecting nodes which were simultaneously active (first fighters, then bombers) were strengthened. By the third pass, **ORESME** has learned to associate the fighters with each other, since activating any one of them (in this case, the Zero) activates all the remaining fighters but none of the

Fighters and Bombers

Threshold = .000 Restoring Force = .100 Learning Rate = .050

Concept	1	2	3	4	5	6	7	8	9	10
				Cycles X 5						
P38	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MUSTANG	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
CORSAIR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B25	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B26	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B17	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B29	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
WARHAWK	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SPITFIRE	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ZERO	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Threshold = .000 Restoring Force = .100 Learning Rate = .050

Concept	1	2	3	4	5	6	7	8	9	10
				Cycles X 5						
P38	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MUSTANG	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
CORSAIR	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B25	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B26	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B17	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B29	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WARHAWK	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
SPITFIRE	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
ZERO	-1.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshold = .000 Restoring Force = .100 Learning Rate = .050

Concept	1	2	3	4	5	6	7	8	9	10
				Cycles X 1						
P38	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
MUSTANG	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
CORSAIR	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B25	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B26	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B17	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B29	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
WARHAWK	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SPITFIRE	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ZERO	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Threshold = .000 Restoring Force = .100 Learning Rate = .050

Concept	1	2	3	4	5	6	7	8	9	10
				Cycles X 1						
P38	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MUSTANG	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
CORSAIR	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
B25	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B26	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B17	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B29	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WARHAWK	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
SPITFIRE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
ZERO	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

FIGURE 5

bombers. And, by fourth pass, the bombers have also been classified as a category by **ORESME**, since activating one of them (the B26) activates all the other bombers, but none of the fighters. These patterns that **ORESME** has learned are written out to the modified weight matrix, PLANES.WGT

OTHER INPUT

Regardless of how complicated a neural network may be in nature, in principle a network consists solely of a set of neurons, each with its characteristic activation function, and a set of connections or weights linking the neurons to each other. In principle. this set of connections can be described completely by a square matrix of numbers, n x n, where n is the number of neurons in the network, and each entry w_{ij} represents the strength of the connection between the i_{th} and the j_{th} neuron. In **Terra** terminology, such a matrix is called a *weight input matrix*, or .WIN matrix. Any square matrix which meets these formal requirements will suffice as input to **ORESME**.

Typically, .WIN matrices most frequently come from either **CATPAC** or **GALILEO**, but any covariance, correlation, co-occurrence matrix or other square matrix can be read easily by **ORESME**. (This is not to say that any square array of numbers will give a reasonable output. There is -- prophets to the contrary -- no mathematical technique whatever that can turn useless inputs into useful outputs. But, formally speaking, a wide array of analytic procedures yield data that is appropriate input to **ORESME**.

Threshold = .000 Restoring Force = .100 Learning Rate = .050
Concept Cycles X 1

	1	2	3	4	5	6	7	8	9	10
LITTLE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
CAESAR	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
DOMINO	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
INEXPENSIVE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
PIZZAHUT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
TWO	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
FAST	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
LIKE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
DELIVERY	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
YOU	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ONE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
QUALITY	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
WANT	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
PIZZA	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
FASTER	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
PLACE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0

Threshold = .000 Restoring Force = .100 Learning Rate = .050
Concept Cycles X 1

	1	2	3	4	5	6	7	8	9	10
LITTLE	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
CAESAR	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
DOMINO	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
INEXPENSIVE	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
PIZZAHUT	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
TWO	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
GOOD	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FAST	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
LIKE	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
DELIVERY	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
YOU	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
ONE	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
QUALITY	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
WANT	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
PIZZA	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FASTER	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
PLACE	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

FIGURE 6

Figure 6 shows an analysis of several interviews about pizza.

The text of these interviews was analyzed by **CATPAC**, which output the weight input network PIZZA.WIN. This file served as input to **ORESME**. When the neurons which represent *fast* and *delivery* are activated, **ORESME** responds *Domino you want faster*. When *Pizzahut* is activated, the network responds with *quality*, and also with *Little Caesar two one inexpensive place*.

ORESME can also accept data directly from the **GALILEO** program. **GALILEO** accepts data about the perceived similarity among concepts, objects, words, products, attributes and the like, and represents these perceptions as objects in a multidimensional space.

In Figure 7, a group of people who planned to buy a Pontiac Grand Am filled out a complete paired comparisons questionnaire reporting their perceptions of the differences among all the cars and attributes listed in Figure 7.

Figure 7 shows that, when *YOURSELF*, the concept which represents the respondent's own position, is activated, many attributes are immediately activated, but ultimately the system settles down until only the attributes *SPORTY LOOKING*, *FUN TO DRIVE*, *EXCITING*, and *LUXURIOUS*, are left active, along with *YOURSELF*, *PONTIAC GRAND AM*, (the car the people in this group plan to buy) and *CHRYSLER LEBARON GTS*. Running the program again in analog mode shows that the *PONTIAC* is more highly activated than the *CHRYSLER*.

PONTIAC INTENDERS

Threshold = .000 Restoring Force = .100 Learning Rate = .050

Concept	Cycles X 1									
	1	2	3	4	5	6	7	8	9	10
SPORTY LOOKING	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FUN TO DRIVE	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
FAMILY CAR	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
GOOD VALUE	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
PRACTICAL	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
AFFORDABLE	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
EXCITING	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
APPEALS TO OLDER PEO	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
LUXURIOUS	.0	1.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
RELIABLE	.0	1.0	.0	.0	.0	.0	.0	.0	.0	.0
HONDA ACCORD	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
MAZDA 626	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
FORD TEMPO	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
TOYOTA CAMRY	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
NISSAN STANZA	.0	.0	.0	.0	.0	.0	.0	.0	.0	.0
CHRYSLER LEBARON GTS	.0	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
PONTIAC GRAND AM	.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
YOURSELF	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Threshold = .000 Restoring Force = .100 Learning Rate = .050

Concept	Cycles X 1									
	1	2	3	4	5	6	7	8	9	10
SPORTY LOOKING	.0	.2	.3	.5	.8	1.0	1.1	1.1	1.1	1.1
FUN TO DRIVE	.0	.2	.2	.5	.7	1.0	1.1	1.1	1.1	1.1
FAMILY CAR	.0	-.1	-.2	-.4	-.7	-.9	-1.0	-1.1	-1.1	-1.1
GOOD VALUE	.0	.1	-.1	-.2	-.5	-.8	-1.0	-1.0	-1.0	-1.0
PRACTICAL	.0	.1	-.2	-.3	-.7	-.9	-1.1	-1.1	-1.1	-1.1
AFFORDABLE	.0	.1	-.1	-.2	-.6	-.9	-1.0	-1.1	-1.1	-1.1
EXCITING	.0	.2	.2	.5	.8	1.0	1.1	1.1	1.1	1.1
APPEALS TO OLDER PEO	.0	-.3	-.2	-.5	-.7	-1.0	-1.1	-1.1	-1.1	-1.1
LUXURIOUS	.0	.0	.0	.3	.5	.8	1.0	1.1	1.1	1.1
RELIABLE	.0	.0	-.1	-.2	-.4	-.6	-.8	-.9	-.9	-.9
HONDA ACCORD	.0	-.1	.1	-.1	-.2	-.5	-.7	-.8	-.8	-.8
MAZDA 626	.0	-.2	.0	.0	.2	.2	.3	.3	.3	.3
FORD TEMPO	.0	-.2	-.2	-.3	-.5	-.8	-.9	-1.0	-1.0	-1.0
TOYOTA CAMRY	.0	-.2	-.1	-.2	-.3	-.5	-.7	-.7	-.8	-.8
NISSAN STANZA	.0	-.1	-.1	-.1	-.1	-.2	-.3	-.4	-.4	-.4
CHRYSLER LEBARON GTS	.0	-.1	.1	.1	.3	.5	.8	.9	.9	.9
PONTIAC GRAND AM	.0	.0	.3	.3	.7	.9	1.1	1.1	1.1	1.1
YOURSELF	1.0	.1	.4	.2	.5	.6	.8	.8	.9	.9

FIGURE 7

Appendix 1: Tools

Your Galileo installation includes a directory called GALILEO\TOOLS. On this directory Terra has supplied three helpful DOS tools. First is a simple read only editor called **LOOK**. **LOOK** is a public-domain program which allows you to examine the contents of any file interactively. It is convenient since you can page up and down or scroll up, down, left and right in the file using the cursor control keys. You can also easily read the 132 column format files that V55 writes. And, since **LOOK** is a read only editor, you don't run the risk of altering important files.

To use **LOOK**, simply enter the command

LOOK [filename]

at the DOS prompt. To leave **LOOK**, press [ESC].

Also included is a very powerful ASCII editor, **EDWIN**. **EDWIN** is a public domain program which follows the formats of WORDSTAR, and can be very helpful in modifying files produced by V55 for use in the other Galileo programs and vice versa. **EDWIN** has complete online help, accessed by pressing **F2** once in the program. To start EDWIN, simply enter the command

EDWIN

at the DOS prompt. You can also enter a file directly with EDWIN by entering the command

EDWIN [filename] .

If you already have an ASCII editor you favor, you may use that instead of **EDWIN**. For more information on installing and using EDWIN, consult the documentation provided on the \GALILEO\TOOLS directory.

The last tool provided is called **UP**. Up lets you climb up your directory tree in only three keystrokes. If your default directory, for example, is GALILEO\DATA, then issuing the command

UP

at the DOS prompt will set your default directory to \GALILEO. Issuing the command again will move you to the root directory.

All three of these utilities are public domain software and are neither warranted nor supported by The Galileo Company, Terra Research and Computing or any of their agents. The are provided at

no charge as a convenience for the user.

Note that authors of public domain software sometimes request voluntary payments from users for the use of their programs. No such payments have been made on your behalf by Terra, Galileo or any of their representatives.