

**Neurocognitive Linguistics
Laboratory**

User Manual

Version 1.2.1

Revision History

1.1.0, October 2010: Rewritten from old Wiki version.

1.2.0, March 2011: Added sections on Network Item Palette and Grid items.

1.2.1, July 8, 2011: Added sections on Multi-Links and the Grid Viewer window.

Table of Contents

Introduction.....	4
Getting Started.....	5
Download and Install.....	5
Windows.....	5
Mac OS X.....	5
Linux.....	5
Main Window.....	6
Tool Bar.....	7
Properties Bar.....	8
Network Items.....	9
Network Editing Area.....	10
Data Display Area.....	11
Other Menu Options.....	11
Creating and Editing a Network.....	12
Editing.....	13
Labels.....	13
Sub-Networks.....	14
Simulating the Network.....	15
Collecting Data.....	17
Network Items and Properties.....	18
Network.....	18
Abstract Notation.....	19
Abstract AND Node.....	19
Abstract OR Node.....	20
Abstract Link.....	21
Narrow Notation.....	22
Narrow Node.....	22
Narrow Oscillator.....	23
Narrow Excitatory Link.....	24
Narrow Inhibitory Link.....	24
Grids.....	25
Grid Item.....	25

Grid Viewer.....	27
Edge Connector.....	27
Text IO Item.....	27
Multi-Link.....	28
Misc.....	28
Text Item.....	28
Sub-Network Item.....	28
Appendix A: Mathematical Details.....	30
Automaton Cells.....	30
Nodes.....	31
Oscillators.....	31
Excitatory Links.....	32
Inhibitory Links.....	32
Link Learning.....	32
Node Learning.....	33
Implementation Details.....	33
Narrow Items.....	33
Abstract Items.....	33
Appendix B: License.....	35
Appendix C: References.....	36

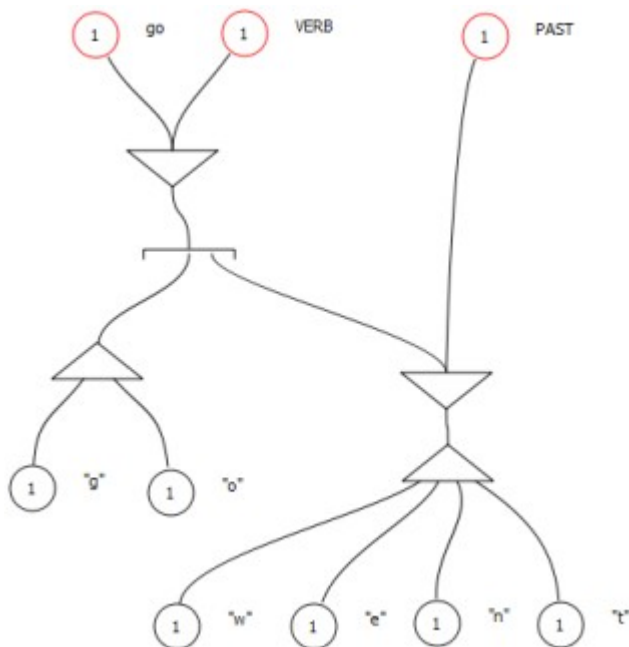


Figure 1: A simple relational network in the Neurocognitive Linguistics model

Introduction

Neurocognitive Linguistics is an approach to linguistics developed by [Sydney Lamb](#) which uses relational networks to model what the brain actually does when it handles language. You can read more about it at the [LangBrain](#) web site and the [Glottopedia](#) wiki, or in Lamb's books [Pathways of the Brain: The Neurocognitive Basis of Language](#) and [Language and Reality: Selected Writings of Sydney Lamb](#).

Neurocognitive Linguistics Lab ("NeuroLab" for short) is a program that allows you to experiment with relational networks using a convenient graphical user interface and record the results of your experiments in tabular form.

You can use NeuroLab to develop template-based networks that contain millions of nodes.

Neurocognitive Linguistics Lab is Copyright © 2010-2011 Gordon Tisher, and available under the terms of the BSD License (see Appendix B: License).



This document is Copyright © 2010-2011 Gordon Tisher, and available under the terms of the [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](#).

Getting Started

You can find this manual, as well as other information, at the [NeuroLab website](#).

Download and Install

The latest distribution of NeuroLab can be found at the [Download Page](#). NeuroLab is available for Windows XP/Vista/7, Mac OS X, and Linux.

Windows

The Windows distribution comes in two forms, a ZIP file and a Windows Installer MSI file. To run NeuroLab from the ZIP file, download it, then right-click it in Windows Explorer and choose "Extract All" from the context menu. This will create a new folder containing the executable program `NeuroLab.exe`, along with a folder named `samples` that contains several sample files.

If you choose to use the MSI file, simply download and open it. This will install NeuroLab to `C:\Program Files\NeuroLab` (or wherever else you choose to install it) and create a Start Menu entry for it. For the MSI installation, the sample files can be located in `C:\Program Files\NeuroLab\samples`.

Mac OS X

The OS X distribution is a DMG disk image file, which, when you download it, should open to reveal the NeuroLab application, which you can drag to your `Applications` folder, if you like, or run directly from the mounted DMG. A `samples` folder containing several sample files is included in the DMG.

Linux

The Linux distribution is a tarball. Download and extract it. The resulting directory contains a script called `neurolab` that launches the application, as well as a `samples` directory containing several sample files.

Main Window

This is NeuroLab's main window:

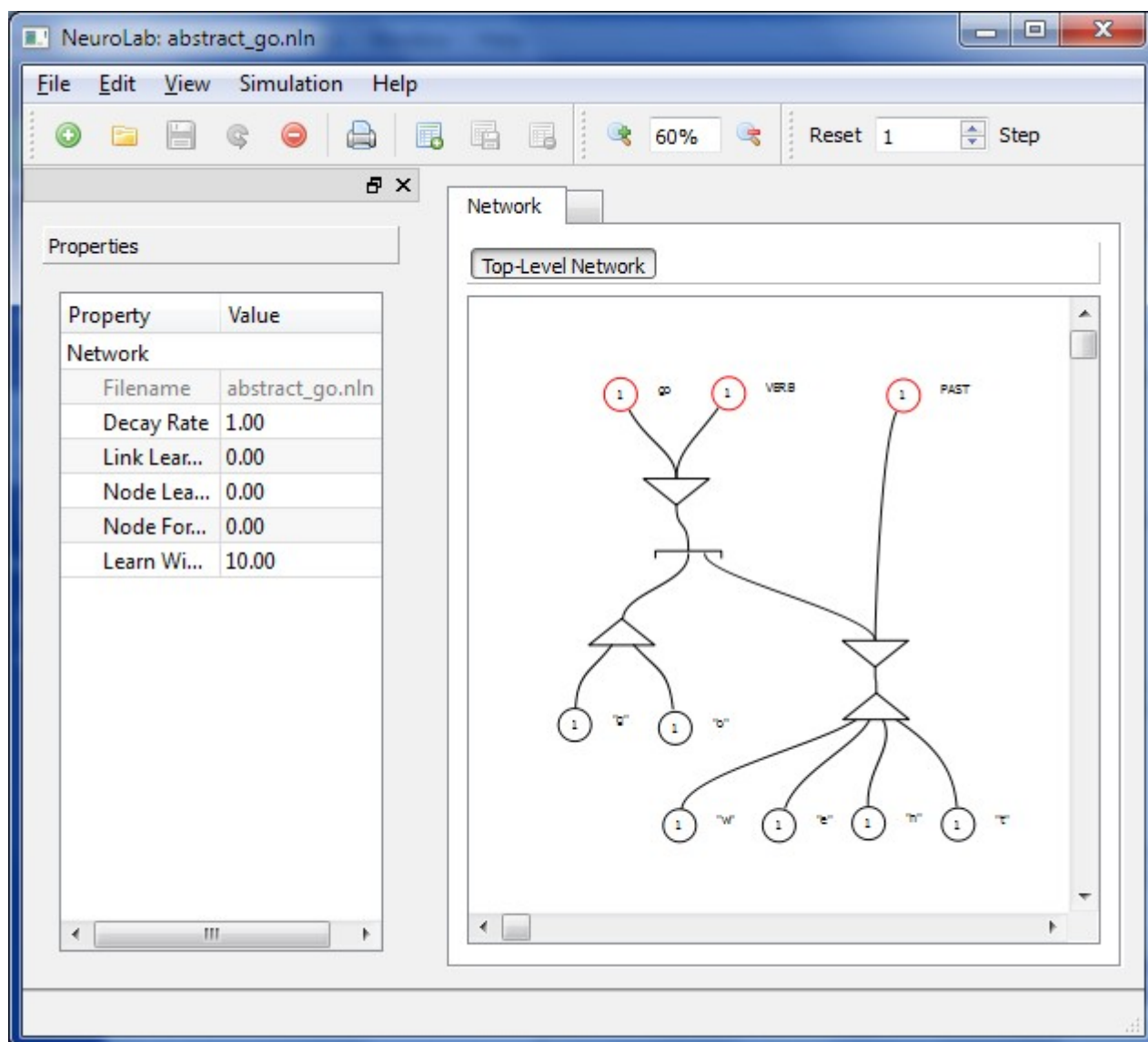


Figure 2: NeuroLab Main Window

The program is organized around a **Network** file, which stores the relational network that you create, and one or more **Data** files, which contain data from simulating spreading activation in the network. When a network file is loaded, its file name appears in the title bar of the main window.

There are several different parts to the main screen:






Tool Bar







Figure 3: Tool Bar




The bar of icons across the top of the screen allows quick access to program functions. All of these functions are also available via the menus. If you move your mouse pointer over one of the buttons and wait a second or two, a “tool tip” will be displayed that describes the button.

The buttons are, from left to right:

-  **New Network:** Create a new network file.
-  **Open Network:** Open an existing network file.
-  **Save Network:** Saves the current network file to disk.
-  **Reload Network:** Discards any changes to the current network file, and reloads it from disk. Since there is no Undo function yet, this is useful for backing out of changes.
-  **Close Network:** Closes the network file.

-  **Print:** Prints an image of the portion of the network that is visible in the editing area.

-  **New Data File:** Initializes a new data file.
-  **Save Data File:** Saves the data file to a Comma-Separated Value (CSV) text file.
-  **Close Data File:** Closes the current data file.

-  **Zoom In:** Zooms in the view of the editing area.
-  **Zoom Percent:** Allows you to directly edit the zoom factor for the editing area.
-  **Zoom Out:** Zooms out the view of editing area.

Toolbar buttons continued...

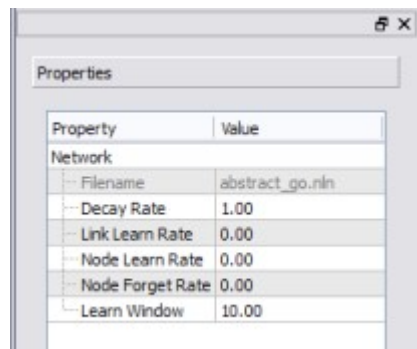
Reset **Reset:** Resets the network. This sets all network activation to zero.

Stop: Allows you to stop the network simulation when you have specified a large number of steps.

Number of Steps: This allows you to set the number of time steps the network will advance when you press the Step button.

Step: Advances the network simulation by the number of time steps you have specified.

Properties Bar



The screenshot shows a window titled 'Properties' with a table of network properties. The table has two columns: 'Property' and 'Value'. The properties listed are: Filename (abstract_go.nln), Decay Rate (1.00), Link Learn Rate (0.00), Node Learn Rate (0.00), Node Forget Rate (0.00), and Learn Window (10.00). Each value in the table is highlighted with a light gray background, indicating it is editable.

Property	Value
Network	
Filename	abstract_go.nln
Decay Rate	1.00
Link Learn Rate	0.00
Node Learn Rate	0.00
Node Forget Rate	0.00
Learn Window	10.00

Figure 4: Properties Bar

The Properties Bar displays a list of properties for the network item that you have currently selected, or the overall network file properties if you have not selected anything. Most of these properties are editable. Left-click on the value of a property to edit it.

Network Items

The network item list contains a list of the available network items:

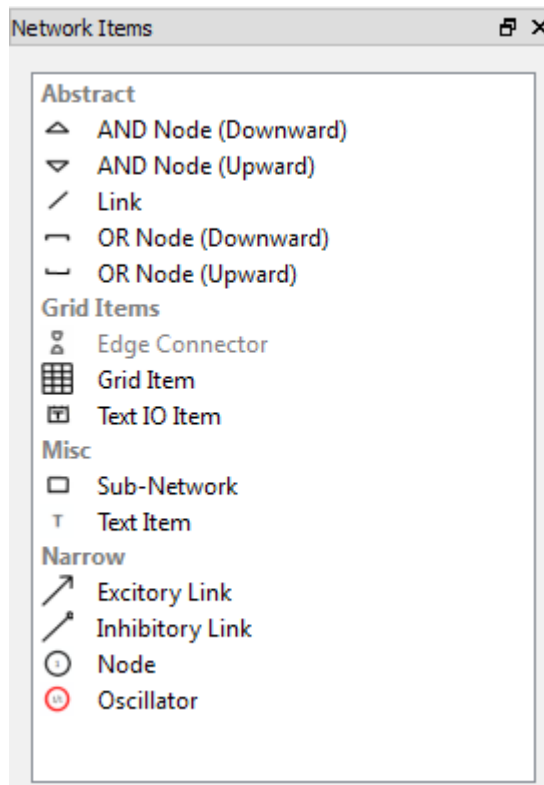


Figure 5: Network Item List

You can drag an item from this list into the main editing area.

Network Editing Area

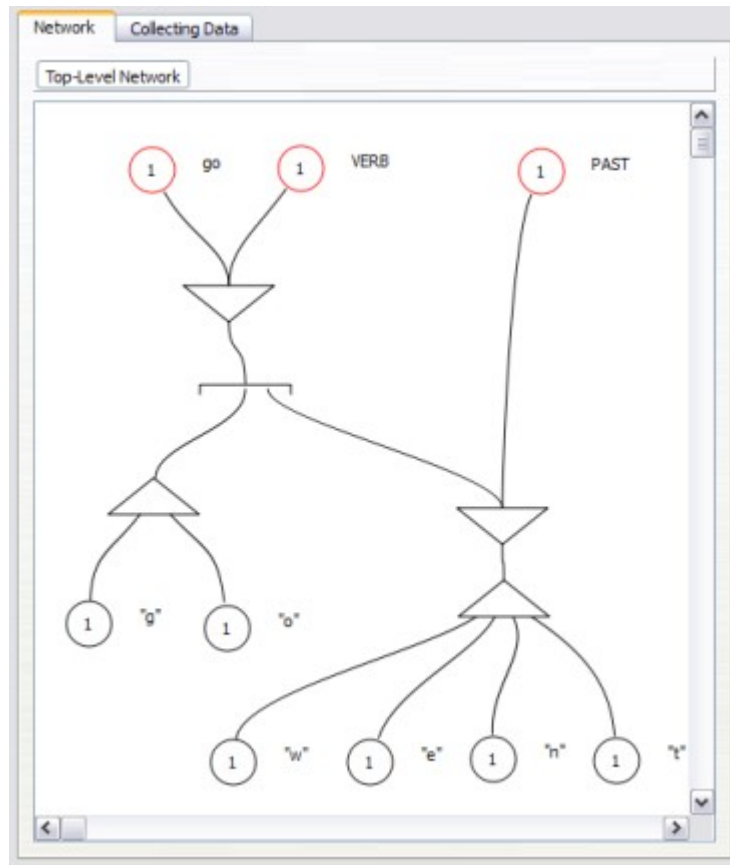
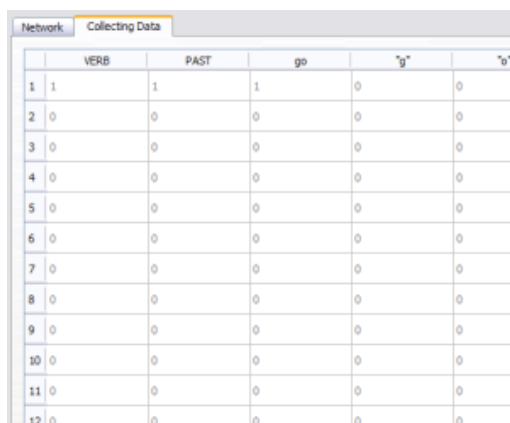


Figure 6: Network Editing Area

The network editing area is where you create and connect the items that make up your relational network. See **Creating and Editing a Network** below for details.

Data Display Area



	VERB	PAST	go	'g'	'o'
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0

Figure 7: Data Display Area

If you have currently created a data file to record the results of your simulations, you can view the data in the data display area. The numbers along the left-hand side are the time steps, and the columns record the values of any labelled items in your network.

Other Menu Options

There are a few other menu options to note:

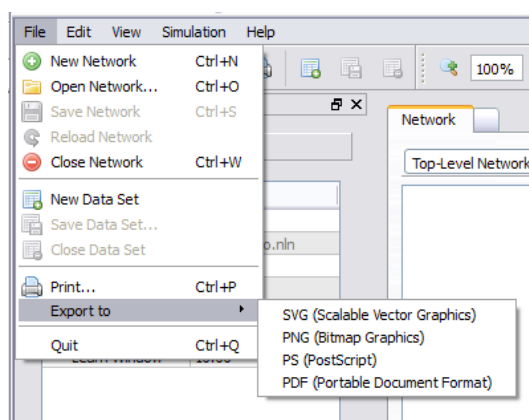


Figure 8: Export Menu

The **Export** menu allows you to save an image of your network (actually, the part of your network that is visible in the editing area) to a number of different file formats.

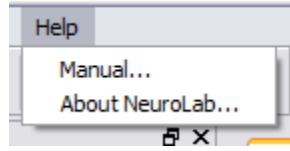


Figure 9: Help Menu

The **Help** menu allows you to view this manual, or display some information about the license and version of NeuroLab that you are using.

Creating and Editing a Network

When you create a new network, you are presented with a blank editing area. In order to add new items to your network, you can drag network items from the Network Item sidebar into the editing area.

You can also right-click in the blank white space of the editing area (hold down Control and click on a Mac with only one mouse button).

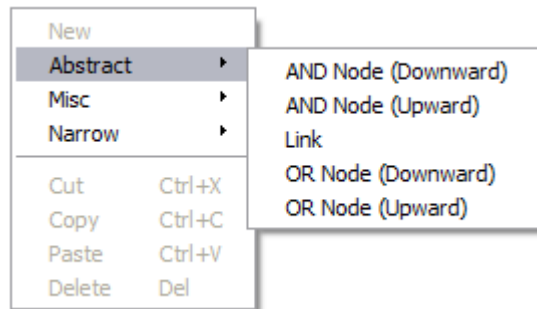


Figure 10: New Item Menu

Choose one of the available items from the menu that pops up, and it will appear in the network editing area. It will be automatically selected when you first create it, so you can immediately modify its properties if you need to.

See **Network Items and Properties** below for a list of the various items you can create, and their properties.

Editing

After you create a new network item, you can left-click and drag it to wherever you like in the editing area. You can also left-click and drag in the empty white space of the editing area to select multiple items, which you can then move, delete, or copy and paste.

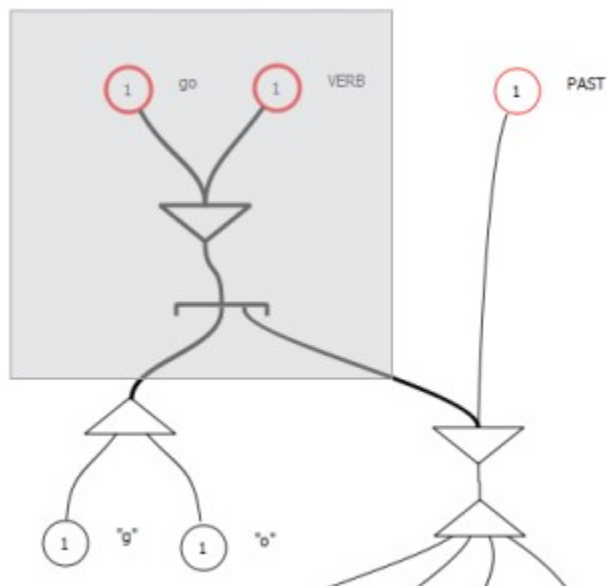


Figure 11: Multiple Selection

After you create a link, you can drag either end of it close to a compact or narrow node, and it will connect itself to the node. You can tell when an item is connected because it will “snap” to a new position.

You can pick up any item, whether a node or a link or otherwise, and move it without changing its shape by holding down the Alt key while you drag the item.

Some items (Narrow Nodes, for example), will allow you to create new links directly on top of them. The new links will be created already connected to the node.

Labels

The first property of all network items is its label. If you give an item a label, the label will be displayed beside the item.

Property	Value
Node	
Label	This item has a label
Output Value	0.00

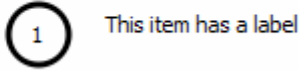


Figure 12: A Network Item with a Label

NOTE: If an item is labelled, its output values will be recorded in the data file. Items without labels will not be recorded.

Sub-Networks

One of the items you can create is called a Sub-Network item (Misc/Sub-Network in the new item menu). After you create a sub-network item, you can double-click on it, and the editing area will display a new blank network. You can navigate back to the top-level network and the sub-networks that you have opened by means of the buttons at the top of the editing area:

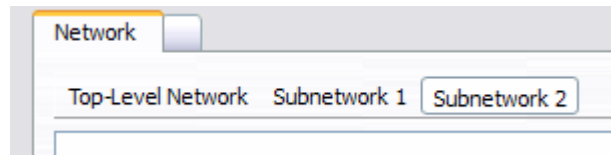


Figure 13: Sub-Networks

You can connect links to the outer sub-network item:

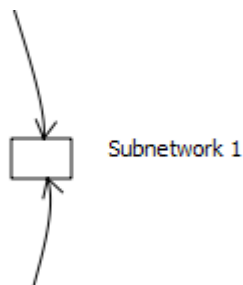


Figure 14: Link to Sub-Network

When you connect a link to a sub-network item, the ends of the links will appear inside the sub-network itself:

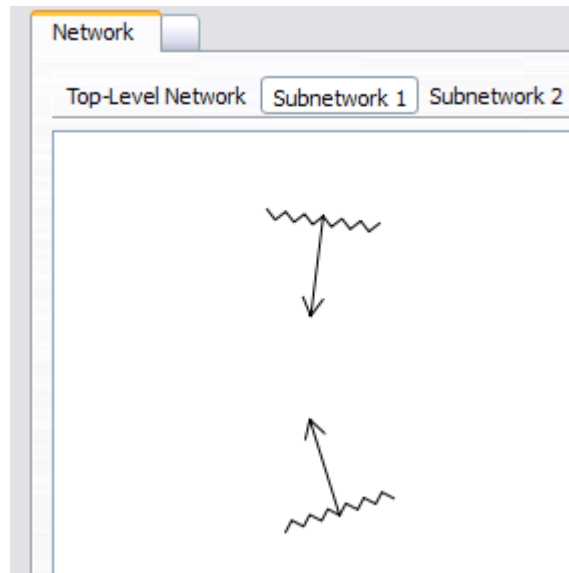


Figure 15: Links Inside a Sub-Network

You can connect these inner links to other items inside the sub-network, and they will transmit activation from the outer network to the inner (and vice versa) just as you would expect.

Simulating the Network

When you wish to simulate spreading activation in your network, you can right-click on one or more nodes or links in the network and choose "Activate/Deactivate" from the menu. When an item is activated, it will turn red.

To simulate one time step in the network, and allow the activation to spread, press the Step button.

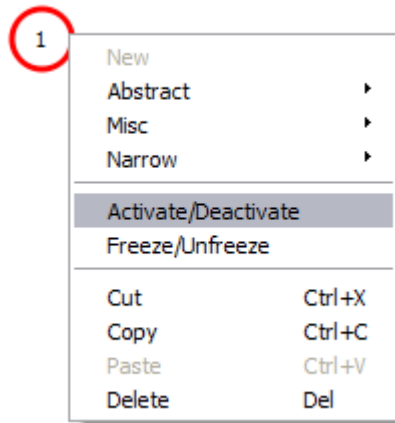


Figure 16: Activation

If connections have been set up from that item, when you press the Step button, the activation will begin to spread to those other nodes.

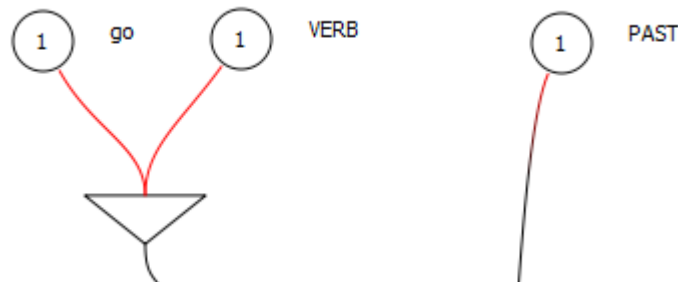


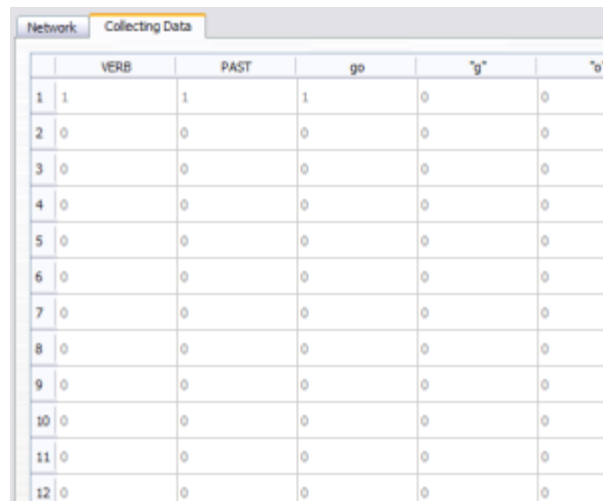
Figure 17: Spreading Activation

NOTE: the output value of an item depends on the values of its inputs *at the previous time step*. This can be a source of some confusion, as by default activated nodes lose their activation within one time step, and it is not always obvious where the source of activation for a given node was, especially if you reload a file that has activation present.

The network's **Decay Rate** property governs how fast an item will lose its activation. If the decay rate is 1, items will lose 100% of their activation in the time step after they become active. If you set the decay rate to 0.5, they will lose half, etc.

Collecting Data

In order to collect data from your simulations, create a new data file, and give labels to the network items that you wish to record data for. Then, when you step through the simulation, the output values of nodes that have labels will be recorded. When you are done you can save the data to a text file in Comma-Separated Value (CSV) format.



	VERB	PAST	go	"g"	"g"
1	1	1	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0

Figure 18: Data Display Area

The rows of the data file correspond to the time steps of your simulation, and the columns contain the labels of the items that you are recording.

NOTE: only network items that have labels will be recorded in the data file.

Network Items and Properties

This section contains a list of the network items that are available by default in NeuroLab.

Network

The network itself contains a number of properties that control how the simulation works:

Network	
Filename	link_learning.nln
Decay Rate	1.00
Link Learn Rate	0.50
Node Learn Rate	0.00
Node Forget Rate	0.00
Learn Window	10.00

Figure 19: Network Properties

- **Filename:** The file name of the current network.
- **Decay Rate:** The rate at which items will lose their activation in a given time step. If the rate is 1, then items will lose 100% of their activation in the next time step. If the rate is 0.5, they will lose half of their activation, etc.
- **Link Learn Rate:** A factor that determines how much links will "learn" during the simulation. If this is greater than zero, links whose activation causes their targets to become active after a period of inactivity will have their weights increased. See Appendix A: Mathematical Details for details.
- **Node Learn Rate:** A factor that determines how much nodes' thresholds will increase if they are activated after a period of inactivity.
- **Node Forget Rate:** A factor that determines how much nodes' thresholds will decrease if their activation level has not changed over a period.
- **Learn Window:** Link and node learning depends on a running average of the links' and nodes' values over a period of time steps. The Learn Window specifies the number of time steps over which this running average will be calculated.

Abstract Notation

These items allow you to build networks using Sydney Lamb's Abstract (formerly "Compact") notation. Abstract nodes and links are bidirectional; they allow activation to be transmitted in two directions. Activation in one direction does *not* affect activation in the other.

NOTE: You can freely mix abstract and narrow items. I.e. you can connect uni-directional narrow links to abstract nodes, and bi-directional abstract links to narrow nodes. You can even use narrow inhibitory links to inhibit abstract bidirectional links.

Abstract AND Node

The **Abstract AND** item takes one link to the upward (or downward) tip of its triangle, and one or more links to the base of the triangle. Activation from the top link will spread to all of the bottom ones, or conversely, if all of the bottom links transmit activation, the top link will be activated.

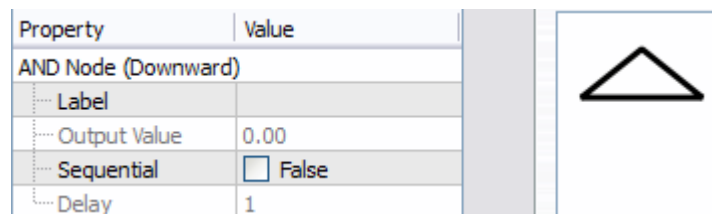


Figure 20: Abstract AND

- **Label:** The node's label.
- **Output Value:** The node's current output value. This is a convenience property which is set to the maximum of its upward and downward activation values.
- **Sequential:** Specifies whether or not the node is **sequential**. A sequential node will spread activation to its bottom links one at a time from left to right, rather than all in the same time step. Conversely, it will only activate its top link if it receives input activation to its bottom links in a precise sequence from left to right.
- **Delay:** For a sequential node, this determines the number of time steps it will take for activation to be transmitted (or needed to be received) by its bottom links. For a value of 1, the bottom links will be activated one after the other. For a value of 2, the first bottom link will be activated, then the second two time steps after that, and so on.

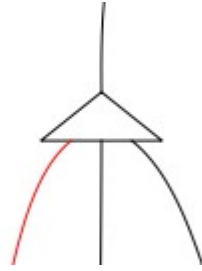


Figure 21: Sequential Abstract AND

You can create both upward- and downward-facing Abstract AND nodes.

Abstract OR Node

The **Abstract OR** item takes one link to its upper side (or lower, if it is an upward-facing node), and multiple links to the center of its lower side. If its upper link transmits activation, then all the lower links will be activated. Going the other way, only one of the lower links needs to be activated for the top link to be activated.

Property	Value
OR Node (Downward)	
Label	
Output Value	0.00



Figure 22: Abstract OR

- **Label:** The node's label.
- **Output Value:** The node's current output value. This is a convenience property which is set to the maximum of its upward and downward activation values.

Choosing Alternatives

If there are links connected to the "arms" of the lower side of the OR node, they function in a special manner. These links are called "alternative" links:



Figure 23: Abstract OR Alternatives

When activation is transmitted downwards through an OR node that contains alternative links, the simulation checks to see if the nodes that the links are connected to are going to activate their targets. If so, the alternative link is allowed to remain active, and the links that are connected to the center of the OR node *are inhibited*, i.e. their activation level is set to zero. If the target node of an alternative link would *not* be activated despite the link being active, the alternative link is itself inhibited, and the central links allowed to remain active.

It's easier to try it out than describe; it works like you would expect it to work.

Abstract Link

An abstract link is bidirectional; it allows activation to be transmitted along it in both directions. Activation in one direction does not affect the other direction.

Link	
Label	
Output Value	0.00
Length	1
Weight	1.10
Frontward Out...	0.00
Backward Outp...	0.00




Figure 24: Abstract Link

- **Label:** The node's label.
- **Output Value:** The node's current output value. This is a convenience property which is set to the maximum of its activation values in both directions. Setting this property will set the output value in both directions.

- **Length:** The number of time steps it will take to transmit activation along the node. The node's appearance will reflect how far along the activation is.
- **Weight:** The maximum of the weights of its links in both directions. The weight of a link is a factor that is multiplied by the activation level of its input to calculate its output value. The default is 1.1, because most nodes will return an output value of slightly less than 1, so a link weight of 1.1 will maintain the activation signal rather than allow it to die out.
- **Frontward Output Value:** The output value of one of the link's directions (it's impossible to tell from the display which is which).
- **Backward Output Value:** The output value of the node in the other direction.

Narrow Notation

These items allow you to build networks using Sydney Lamb's Narrow notation. Narrow links are unidirectional; they will only transmit activation in one direction.

Narrow Node

A narrow node sums up the activation level of all its inputs from the previous time step, and then calculates its output value using a sigmoid function that constrains its output to between zero and one. See Appendix A: Mathematical Details for mathematical details.

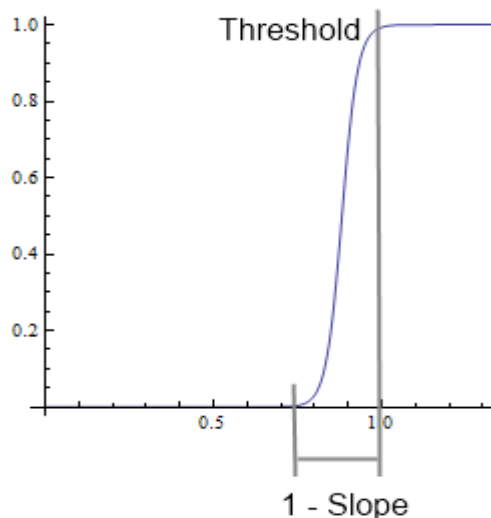


Figure 25: Node Output Sigmoid Function

The number displayed in the middle of the node is its **Input Threshold**.

Property	Value
Node	
Label	
Output Value	0.00
Frozen	<input type="checkbox"/> False
Input Threshold	1.00
1 / Slope	0.10




Figure 26: Narrow Node

- **Label:** The node's label.
- **Output Value:** The current output value of the node.
- **Frozen:** If a node is frozen, its output value will never change. This is for convenience in providing a steady source of activation.
- **Input Threshold:** The threshold at which the sum of the node's inputs will cause the node to have an output value close to 1. You can think of this as the number of input links that must be active for the node to become active.
- **1 / Slope:** This is the range over which the output value will increase from close to zero to close to one. The default value of 0.1 causes nodes to have an abrupt cut-off; if the sum of their inputs is less than 0.1 less than their input threshold, they will not have much output. A larger value will make the nodes less sensitive.

Narrow Oscillator

A narrow oscillator provides an alternating source of activation. The numbers displayed in the middle of the node are its **Spike** and **Gap** properties.

Property	Value
Oscillator	
Label	
Output Value	1.00
Phase	0
Spike	1
Gap	1




Figure 27: Narrow Oscillator

- **Label:** The oscillator's label.

- **Output Value:** The oscillator's output value. Will always be either 0 or 1.
- **Phase:** The number of time steps that the oscillator will wait at the beginning of a simulation before starting to be activated.
- **Spike:** The number of time steps for which the oscillator will then be activated.
- **Gap:** The number of time steps the oscillator will then be deactivated.

Narrow Excitory Link

A narrow excitory link transmits activation in one direction.

Property	Value
Excitory Link	
Label	
Output Value	0.00
Weight	1.10
Length	1

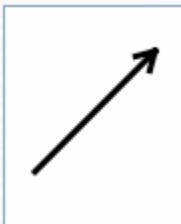


Figure 28: Narrow Excitory Link

- **Label:** The link's label.
- **Output Value:** The output value of the link.
- **Weight:** The weight of the link. This number will be multiplied by the link's input to calculate the link's output value. If the network's *Link Learn Rate* is non-zero, the weight of a link can change depending on whether or not the link's target node gets activated by the link.
- **Length:** The number of time steps the link takes to transmit its activation. The link's appearance will reflect how far along the activation is when activation is present.

Narrow Inhibitory Link

A narrow inhibitory link *inhibits* the activation of its target. If the inhibitory link's input is 1, its target's output value will be 0. An inhibitory link can be connected to a narrow node, narrow oscillator, excitory link, abstract link, or another inhibitory link.

Property	Value
Inhibitory Link	
Label	
Output Value	0.00
Weight	-1.10
Length	1




Figure 29: Narrow Inhibitory Link

- **Label:** The link's label.
- **Output Value:** The link's output value. Will be negative if the link is active.
- **Weight:** The link's weight. You cannot modify the weight of an inhibitory link.
- **Length:** The number of time steps the link takes to transmit its inhibition.

Grids

Creating and simulating relational networks using the standard network items can be interesting for very small networks, but these are too simple to be really representative of anything that might be actually going on in the brain. The nodes and links of the Neurocognitive model are intended to represent *cortical columns* rather than individual neurons in the brain, but there are still hundreds of millions of these.

In order to create networks that contain large numbers of nodes and links, NeuroLab provides a special kind of network item: the Grid.

Grid Item

A Grid item allows you to create a small relational network template which is then copied over and over many times to create a larger network.

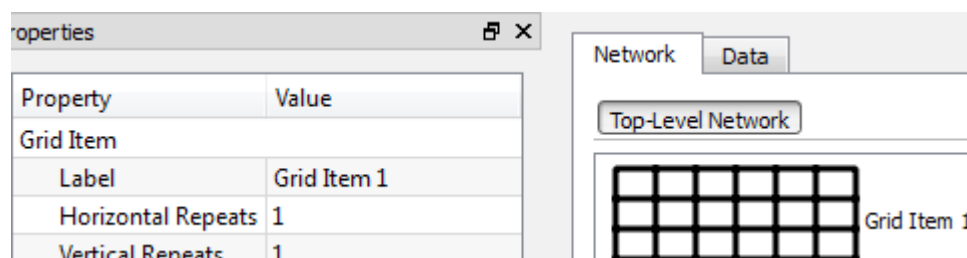


Figure 30: Grid Item

- **Label:** The Grid Item's label.
- **Horizontal Repeats:** The number of times the template network will be repeated horizontally to create the resulting network grid.
- **Vertical Repeats:** The number of times the template network will be repeated vertically to create the resulting network grid.

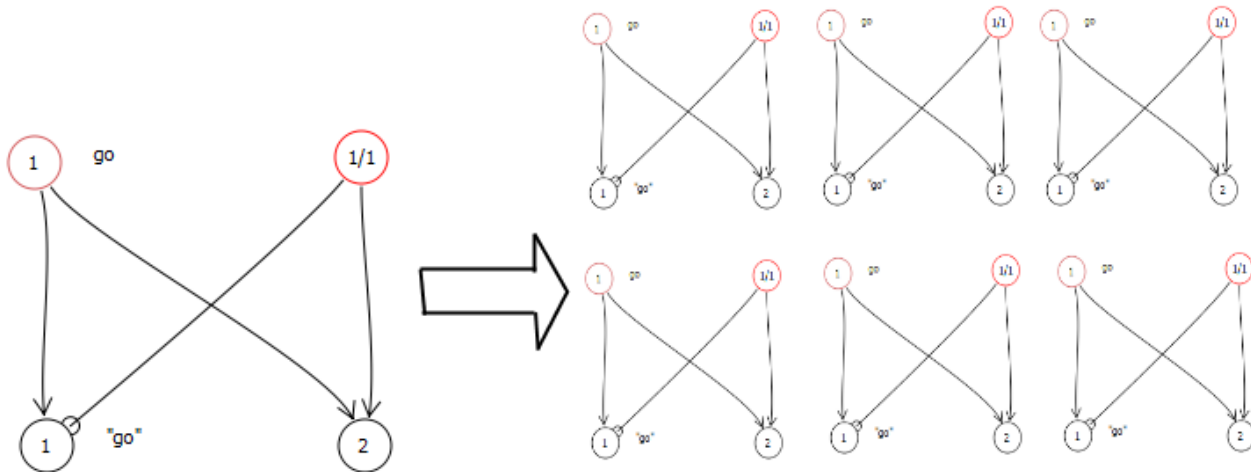


Figure 31: Grid Item Templates -> Network Grid

You can double-click on a Grid Item the same as a Sub-Network item, but the network that you edit inside a Grid Item is not actually part of the outer network. Rather, it is a template. When you start simulating the network, the template will be repeated as many times vertically and horizontally as the Grid Item's properties specify.

You will notice that you can only connect links to the top and bottom of a Grid Item, and connecting links to the Grid Item does not create inner links inside it. A repeated template is no use if each instance is not connected to another part of the network, however. So you can use Edge Connectors in the template network.

Edge Connectors provide a notional link between the top and bottom (and left and right) edges of the grid network template. When the actual grid is constructed, items that are connected to the edge connectors in the template will be connected to their corresponding neighbors in the grid. Left-to-right edge connectors will be connected in a loop, but top-to-bottom connectors will be connected to the top and bottom of the grid item itself, to allow for input and output. You can visualize the resulting grid as being wrapped around a cylinder.

NOTE: there is some special behavior when you connect a link to the top or bottom of a Grid Item. The link itself will not be connected to the grid inside (and thus its length is ignored); what happens is that the item on the other end of the link (if any) will be connected to *all* the

connections at the top (or bottom) of the network grid. This behavior does not apply to Text IO Items (see below).

Grid Viewer

You can view what is happening inside a grid by opening the Grid Viewer from the View/Toolbars menu. The Grid viewer presents a 3D view of the cylindrical grid which you can rotate by holding down the left mouse button in the viewer window and dragging.

Edge Connector

An edge connector creates a notional link between neighbor template instances in a network grid.

Text IO Item

A Text IO item is designed to work with Grid Items that have a value of 256 for their Horizontal Repeats property. They allow you to feed in text and display text output from a grid. The reason that they are designed to work with a grid that is 256 templates wide is that they feed their input text into the grid as bytes of UTF-8 encoded text, and receive their output in the same way. The top connections of the grid cells at the indices of the UTF-8 byte will be successively activated when the network is stepped.

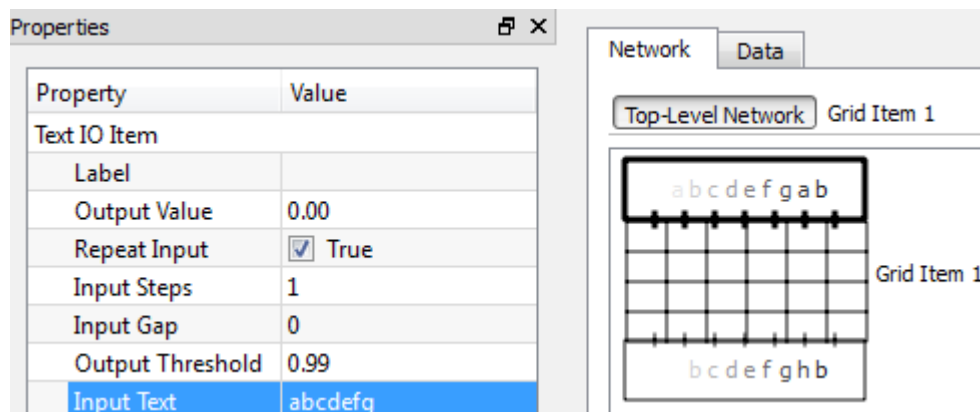


Figure 32: A Grid Item with two Text IO Items

Label: The item's label.

Output Value: Not used.

Repeat Input: Whether or not to repeatedly send the input text.

If you assign a label to a Text IO Item and open a new data file for recording your simulation results, you will capture the text that is being output from the Text IO Item.

Multi-Link

A multi-link is used for connecting two grids. If you set the multi-link's Width property to be the same as the grids that it is connecting, it will transmit activation directly from each output of the first grid to the corresponding input of the other. If the widths don't match, then the inputs and outputs will be grouped together and function unpredictably.

Misc

Text Item

A text item is a convenient way to place arbitrary text in your network editing area. Text items do not interact with any other network items. Currently only one line of text is supported.

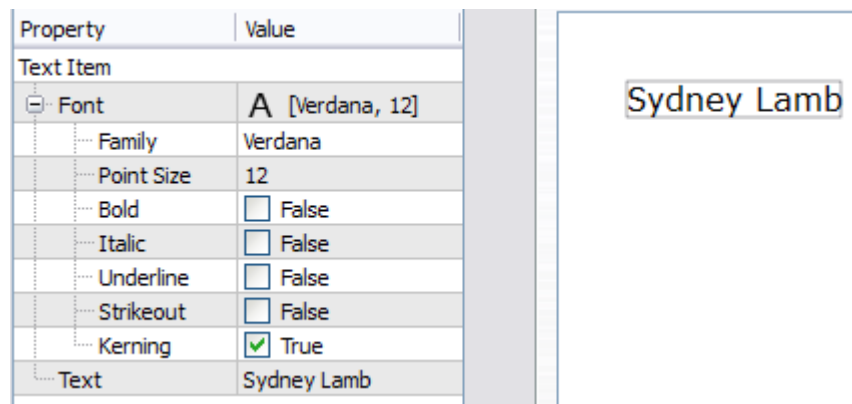


Figure 33: Text Item

- **Font:** The font in which to display the text.
- **Text:** The text to display.

Sub-Network Item

A sub-network item encapsulates a portion of the network.

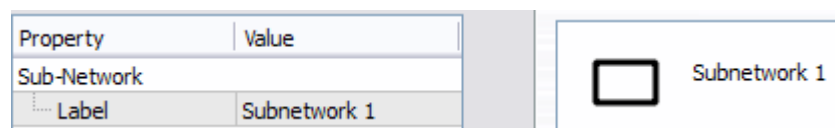


Figure 34: Sub-Network Item

- **Label:** The sub-network's label. This is assigned automatically.

Appendix A: Mathematical Details

Spreading activation is simulated in NeuroLab using an asynchronous network automaton (a network automaton is like a cellular automaton, but a cell's neighbors are not in a fixed grid, they are connected in a directed graph).

All of the node types, whether narrow or abstract, are implemented by combinations of these basic automaton cells.

The automaton is updated using Nehaniv's (Nehaniv2003) asynchronous algorithm. This allows us to use the Qt library's concurrent functionality to update different parts of the network asynchronously on different cores, while guaranteeing that the final state of the network is the same as it would have been had the automaton been updated synchronously.

In the relational network, a cell's inputs are its neighbors in the automaton. As the automaton is a directed graph, the neighbor relationship is not reciprocal.

Automaton Cells

Updating the cells of the automaton happens as follows:

If the cell is frozen, its output does not change.

Otherwise, before updating the cell the following values are calculated:

$$v = \sum_{j=1}^n x_j | x_j > 0$$

Equation 1: Input Sum

The **input sum** v is set to the sum of the values of the cell's inputs, if they are positive.

$$h = 1 - \text{clip}\left\{0, \sum_{j=1}^n x_j | x_j < 0, 1\right\}$$

Equation 2: Inhibition Factor

The **inhibition factor** h is 1 minus the absolute value of all the cell's inputs, if they are negative.

There are four types of cells in the network, each with a different update rule:

Nodes

The **output value** y of a node is calculated using a sigmoid function similar to a generalized logistic function with asymptotes at 0 and 1:

$$y = \frac{h}{1 + e^{6-s(v-(d-r))}}$$

Equation 3: Output Value

where the “**slope**” s of the curve is:

$$s = \frac{6 - \ln(1/d - 1)}{r}$$

Equation 4: “Slope”

and the “**inverse slope**” r is the distance in the input over which the output goes from close to 0 to close to 1, and the **input threshold** d is the point in the input at which the output reaches close to 1.

Oscillators

The output value y of an oscillator is calculated using the function:

$$y = h \begin{cases} 1 & | ((\phi + t) \bmod (g + p)) < p \\ 0 & | \text{otherwise} \end{cases}$$

Equation 5: Oscillator

The **output value** y is set to 1 (times the inhibition factor h) when the **phase** ϕ plus the **timestep** t , modulo the **gap** g plus the **spike** p , is less than p , otherwise it is set to 0.

Excitatory Links

The output value y of an excitatory link is calculated using the function:

$$y = h \text{ clip}\{0, v w, 1.1\}$$

Equation 6: Excitatory Link

where w is the **weight** of the link. The output is clipped to between 0 and 1.1, in order to preserve the output of a node, which may be slightly less than 1.

Inhibitory Links

The output value y of an inhibitory link is calculated using the function:

$$y = h w \text{ clip}\{0, v, 1\}$$

Equation 7: Inhibitory Link

where w is the weight of the link, which is always -1 for inhibitory links. Note that if an inhibitory link is itself inhibited, its output will be 0, and thus its target link's inhibition factor h will be 1. Thus inhibiting an inhibitory link prevents it from inhibiting its target.

Link Learning

After each time step, the weight of a link is modified using Hebbian learning using the covariance hypothesis (Haykin1998, 79):

$$\Delta w = \eta (x - \bar{x})(y - \bar{y})$$

Equation 8: Link Learning

The change in **weight** Δw is the **link learning rate** η , multiplied by the difference between the link's output x and its running average \bar{x} , multiplied by the difference between the link's target's output y and its running average \bar{y} . This ensures that a link's weight will be strengthened if its sudden activation activates its target node, and weakened if the target node loses activation.

Node Learning

After each time step, the input threshold of a node is modified using a learning function similar to that used by Colin Harrison (Harrison2000):

$$\Delta d = l(\text{clip}\{0, y - \bar{y}, 1\}^3 - f)$$

Equation 9: Node Learning

where l is the network's **node learn rate** and f is the **node forget rate**.

This ensures that if a node is continually activated, its input threshold is gradually increased, and if it is not, its threshold gradually decreases.

Implementation Details

The network items that are manipulable in NeuroLab are implemented in the underlying network automaton in various ways:

Narrow Items

Narrow items are implemented using their corresponding automaton cells. Links contain a chain of zero or more excitory link cells, and either an excitory or inhibitory link cell at the end.

Abstract Items

Abstract items contain two paths of automaton cells, one for each direction of activation. Activation travelling in one direction does not affect the other direction.

AND Nodes

Simple AND nodes are implemented using a single automaton node. In the upward direction, its input threshold is modified to equal the number of incoming base links, so all of them must be activated to activate the AND node.

Sequential AND nodes link each base link to a chain of excitory link cells, sized so that incoming activation will arrive at the node at the same time, and outgoing activation will arrive at the base links in sequence.

OR Nodes

OR nodes are implemented using a single automaton node with threshold 1.

For each alternate link, the OR node cheats. It makes a copy of the portion of the network automaton connected to the link's target, and steps the new network one step in the future. If the target becomes active, then the alternate link is allowed to remain active and the main links are inhibited, and vice versa.

Appendix B: License

Neurocognitive Linguistics Laboratory

Copyright © 2010-2011, Gordon Tisher

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Neurocognitive Linguistics Lab nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Appendix C: References

- Harrison2000: Harrison, Colin. *Purenet: a modeling program for neurocognitive linguistics*. Ph.D. Thesis, Rice University. 2000.
- Haykin1998: Haykin, Simon. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall. 1998.
- Lamb1999: Lamb, Sydney. *Pathways of the Brain: the Neurocognitive Basis of Language*. John Benjamins Publishing Co. 1999.
- Lamb2004: Lamb, Sydney. *Language and Reality: Selected Writings of Sydney Lamb*. Continuum International Publishing Group Ltd. 2004.
- Nehaniv2003: Nehaniv, Chrystopher L. Asynchronous automata networks can emulate any synchronous automata network. *Journal of Algebra*, December 2003.