

Neurocognitive Linguistics Laboratory User Manual

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1.1.0	October 2010	Rewritten from old Wiki version.	GT

Contents

1	Introduction	1
2	Getting Started	2
2.1	Download and Install	2
2.1.1	Windows	2
2.1.2	Mac OS X	2
2.1.3	Linux	2
3	Main Window	3
3.1	Tool Bar	4
3.2	Property Bar	5
3.3	Network Editing Area	6
3.4	Data Display Area	7
3.5	Other Menu Options	7
4	Creating and Editing a Network	9
4.1	Editing	9
4.2	Labels	10
4.3	Sub-Networks	11
5	Simulating the Network	13
6	Collecting Data	15
7	Available Items	16
7.1	Network	16
7.2	Abstract Notation	17
7.2.1	Abstract AND Node	17
7.2.2	Abstract OR Node	18
7.2.2.1	Choosing Alternatives	18
7.2.3	Abstract Link	19
7.3	Narrow Notation	20

7.3.1	Narrow Node	20
7.3.2	Narrow Oscillator	21
7.3.3	Narrow Excitatory Link	21
7.3.4	Narrow Inhibitory Link	22
7.4	Misc	23
7.4.1	Text Item	23
7.4.2	Sub-Network Item	23
A	Mathematical Details	24
A.1	Automaton Cells	24
A.1.1	Nodes	24
A.1.2	Oscillators	25
A.1.3	Excitatory Links	25
A.1.4	Inhibitory Links	25
A.1.5	Link Learning	25
A.1.6	Node Learning	25
A.2	Implementation Details	25
A.2.1	Narrow Items	26
A.2.2	Abstract Items	26
A.3	AND Nodes	26
A.4	OR Nodes	26
B	License	27
C	References	28

Chapter 1

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:

- [\[Lamb1999\]](#) [Pathways of the Brain: The Neurocognitive Basis of Language](#).
- [\[Lamb2004\]](#) [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 GUI, and record the results of your experiments in tabular form.

Neurocognitive Linguistics Lab is Copyright © 2010 Gordon Tisher, and available under the terms of the [BSD License](#).

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



Chapter 2

Getting Started

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

2.1 Download and Install

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

2.1.1 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 `samples` directory that contains several sample files. If you choose to use the MSI file, simply download and run it. This will install the program to `C:\Program Files` (or wherever else you choose to install it) and create a Start Menu entry for it. The sample files can be located in `C:\Program Files\NeuroLab\samples`.

2.1.2 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.

2.1.3 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` folder containing several sample files.

Chapter 3

Main Window

This is NeuroLab's main window:

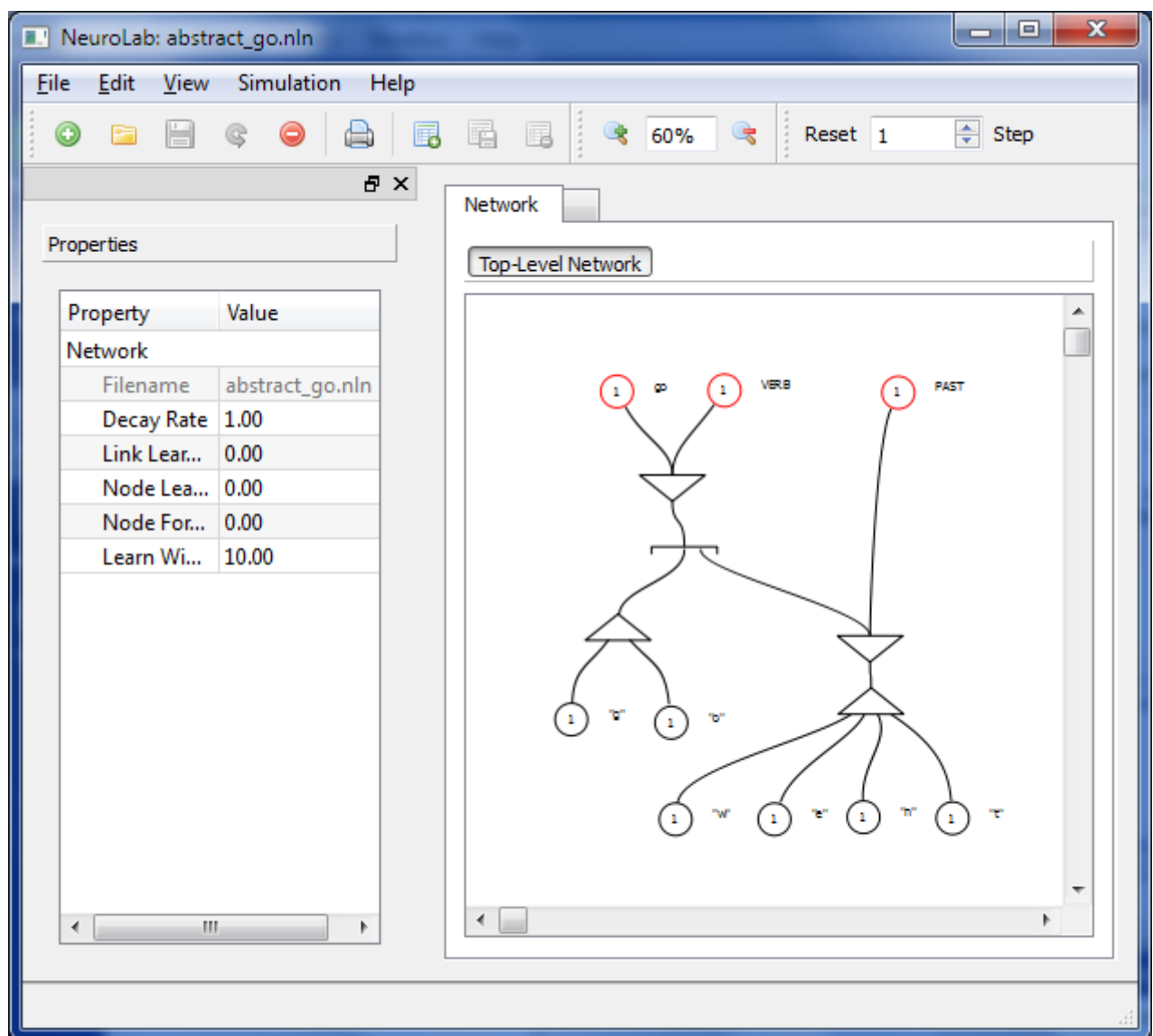


Figure 3.1: Main Window

The program is organized around a Network file, which stores the relational network that you create, and the Data file, which contains data from simulating spreading activation in the network. When a network file is loaded, its filename appears in the title bar of the main window.

There are several different parts to the main screen:

3.1 Tool Bar



Figure 3.2: 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 tooltip 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



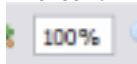
Saves the data file to a Comma-Separated Value (CSV) text file.

Zoom In



Zooms the editing area in.

Zoom Percent



Allows you to directly edit the zoom factor for the editing area.

Zoom Out



Zooms the editing area out.

Reset



Resets the network. This sets all network activation to zero.

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.

3.2 Property Bar

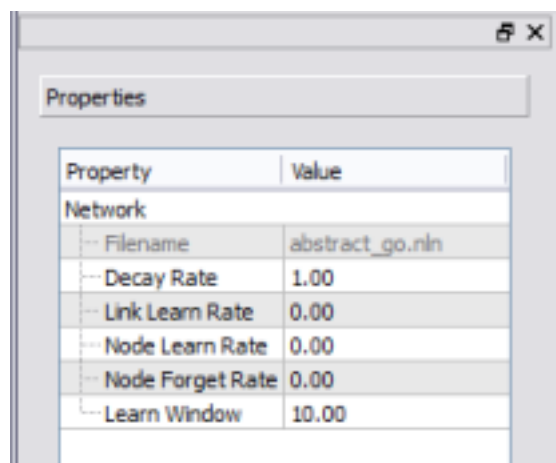


Figure 3.3: Property Bar

The Property 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.

3.3 Network Editing Area

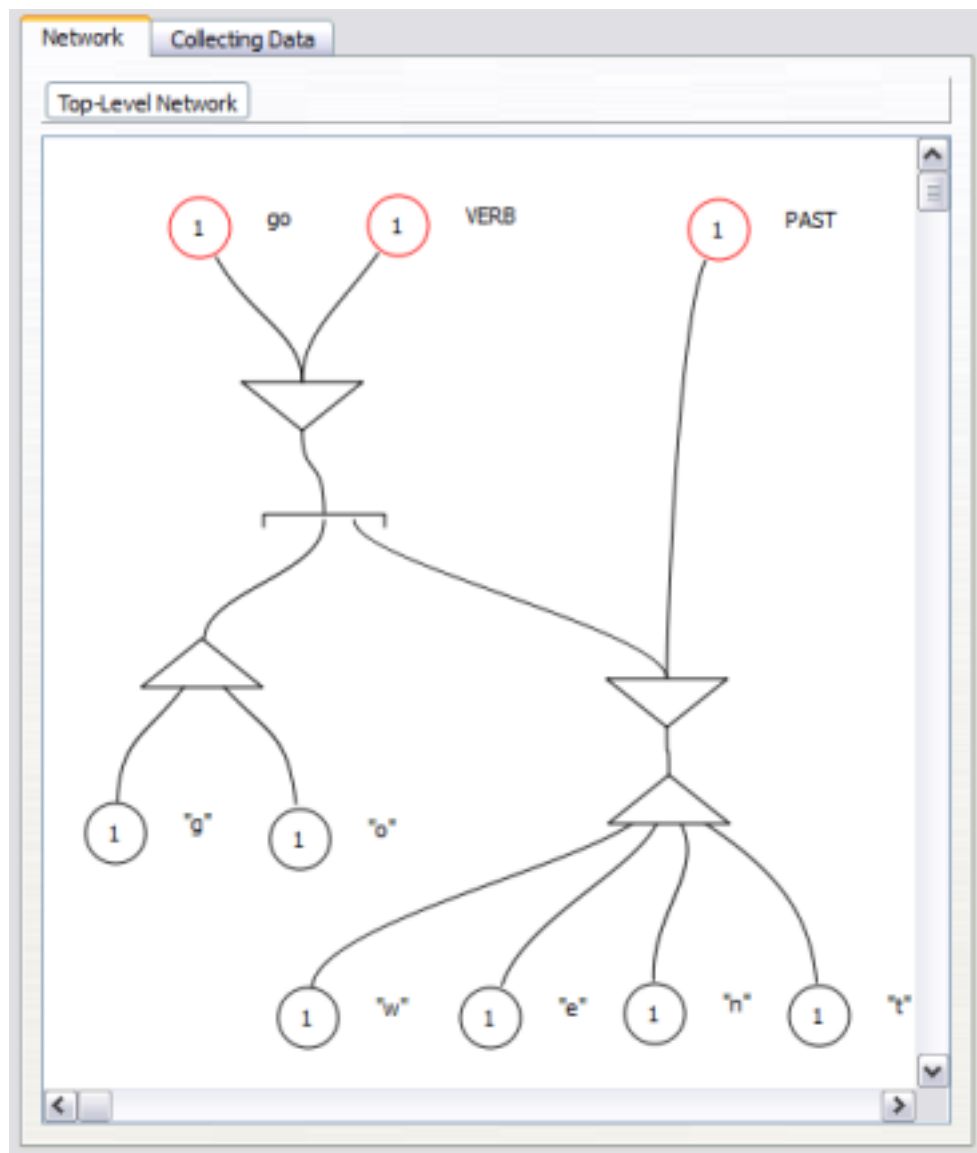
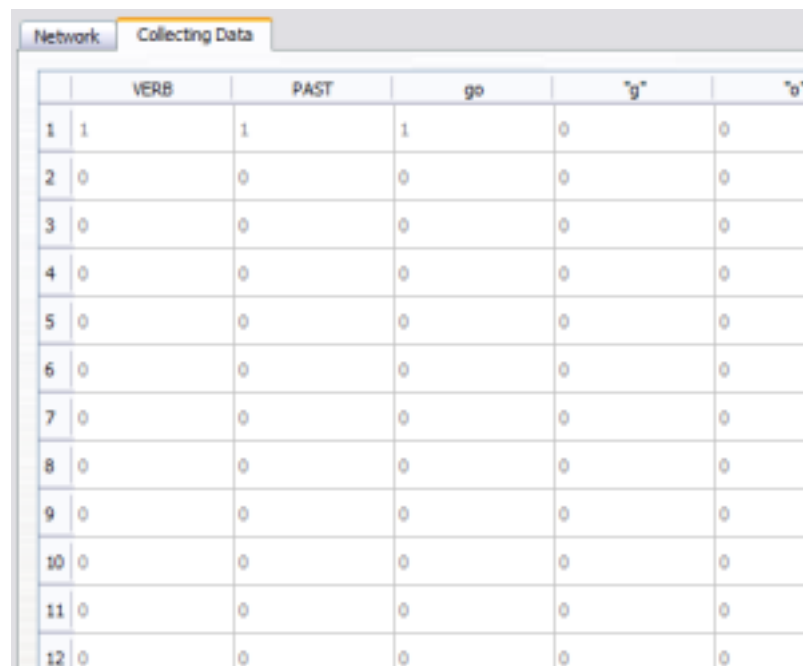


Figure 3.4: Network Editing Area

The network editing area is where you create and connect the items that make up your relational network. See [below](#) for details.

3.4 Data Display Area



		VERB	PAST	go	"g"	"a"
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 3.5: 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 labeled items in your network.

3.5 Other Menu Options

There are a few other menu options to note:

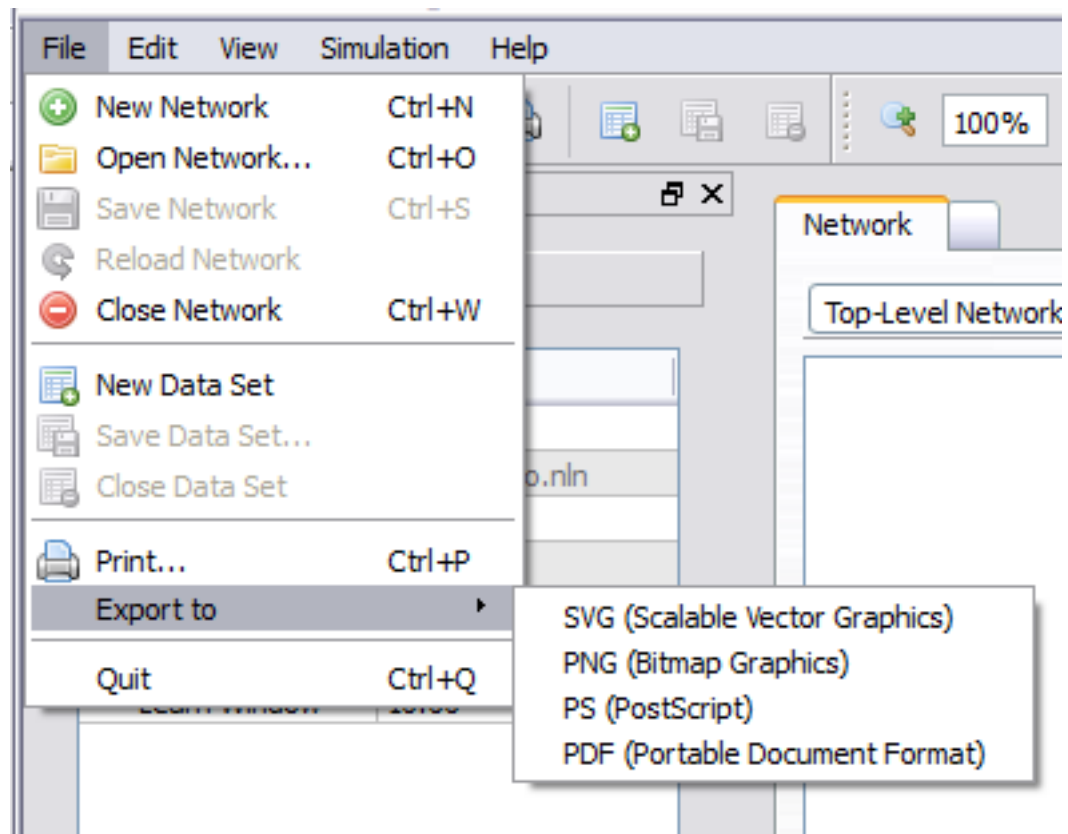


Figure 3.6: Export Menu

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.

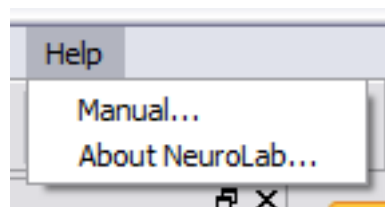


Figure 3.7: Help Menu

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

Chapter 4

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

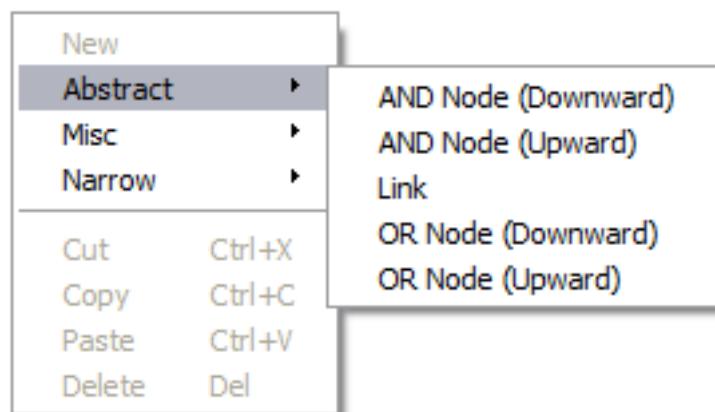


Figure 4.1: New Item Menu

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 [below](#) for a list of the various items you can create, and their properties.

4.1 Editing

After you create an 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.

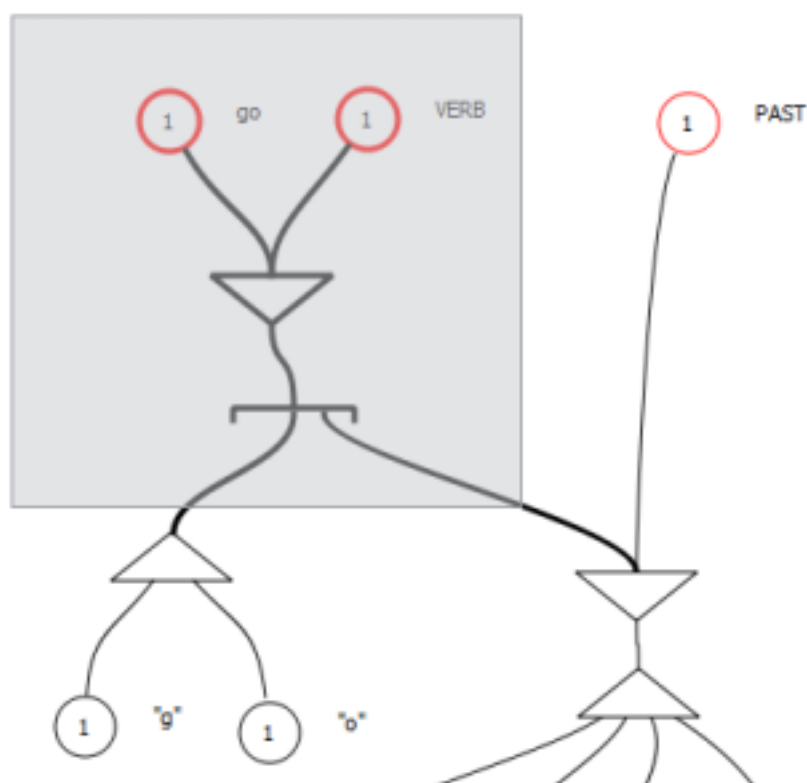


Figure 4.2: Multiple Selection

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 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.

4.2 Labels

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

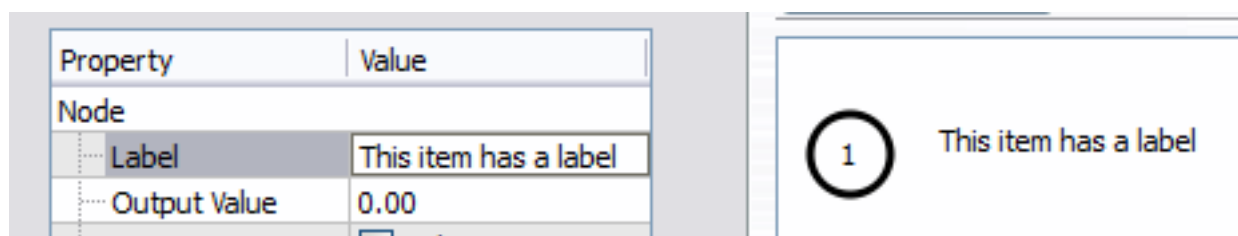


Figure 4.3: Labeled Item

Collecting Data

If an item is labeled, its output values will be recorded in the data file.

4.3 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:

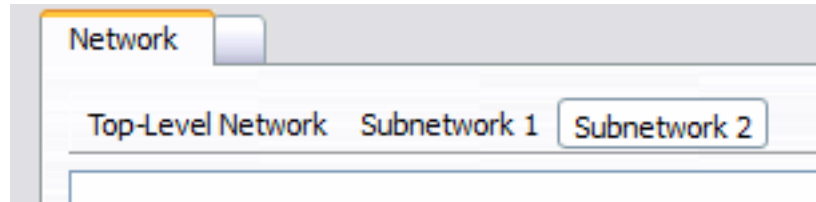


Figure 4.4: Subnetwork Buttons

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

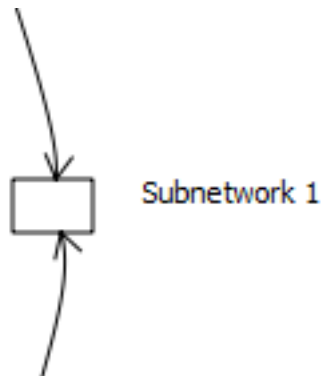


Figure 4.5: Link to Sub-Network

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

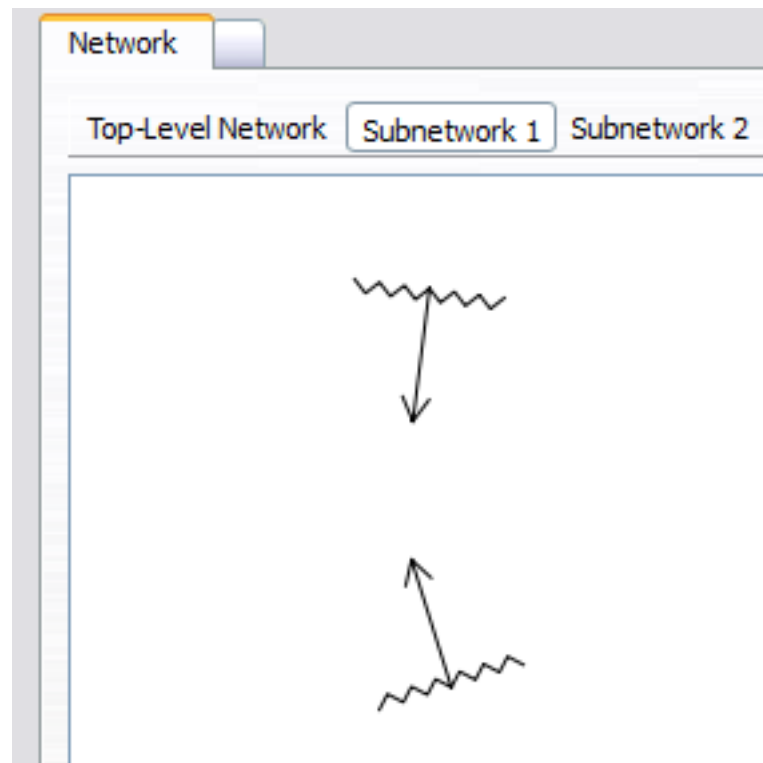


Figure 4.6: Links Inside a Sub-Network

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.

Chapter 5

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.

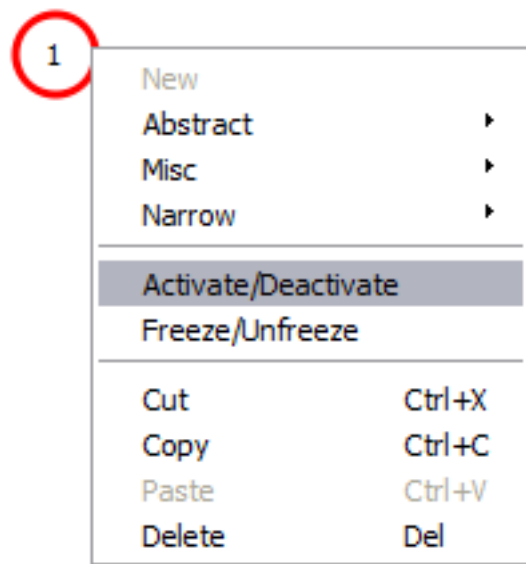


Figure 5.1: Item Activation

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

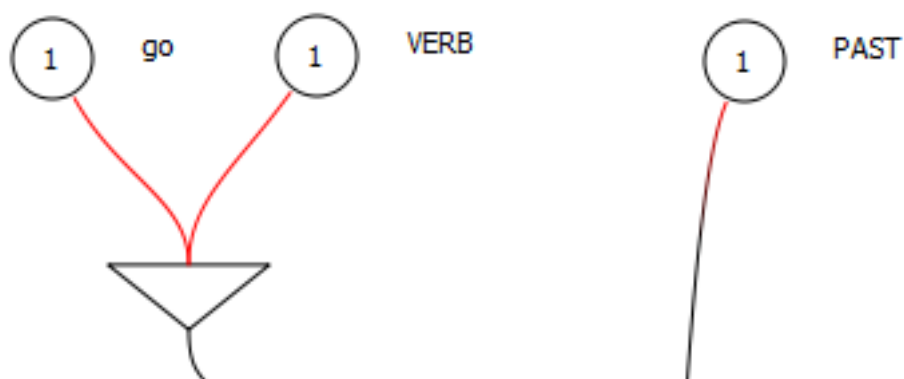


Figure 5.2: Spreading Activation

Output Value

Note that the output value of an item depends on the values of its inputs *at the previous time step*.

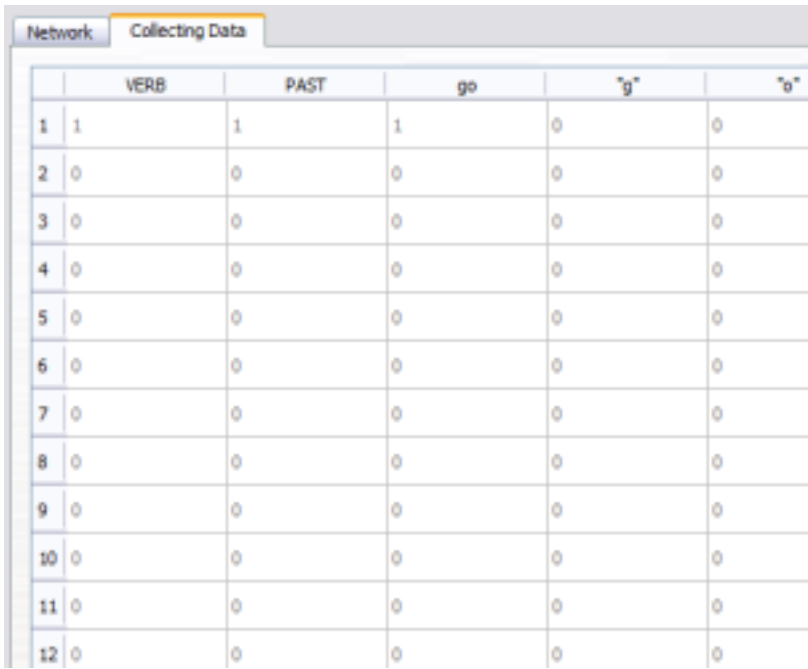
Decay Rate

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.

Chapter 6

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"	"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 6.1: 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.

Chapter 7

Available Items

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

7.1 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 7.1: 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 [below](#) for mathematical details.

Node Learn Rate

A factor that determines how much nodes’ thresholds will increase if they are activated after a period of inactivity. See [below](#) for mathematical details.

Node Forget Rate

A factor that determines how much nodes’ thresholds will decrease if their activation level has not changed over a period. See [below](#) for mathematical details.

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.

7.2 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.

Mixing Abstract and Compact Items

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.

7.2.1 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.

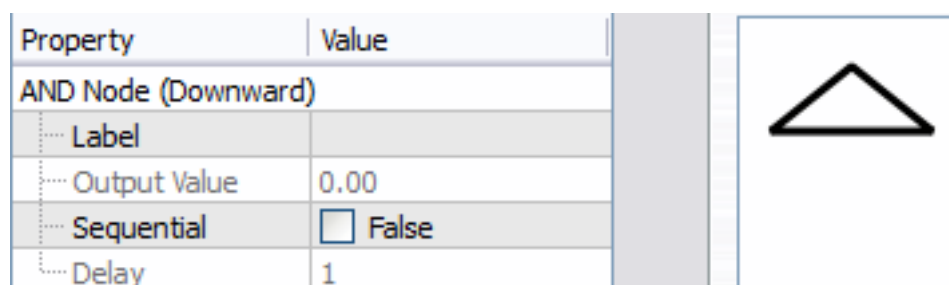


Figure 7.2: Abstract AND

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.

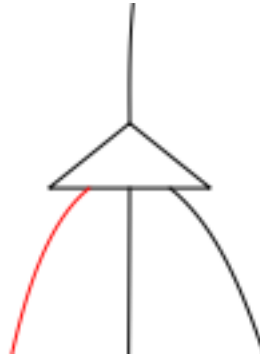


Figure 7.3: Sequential Abstract AND node

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

7.2.2 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 7.4: 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.

7.2.2.1 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 7.5: 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.

7.2.3 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 7.6: 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.

Backward Output Value

The output value of the node in the other direction.

7.3 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.

7.3.1 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 [below](#) for more mathematical details.

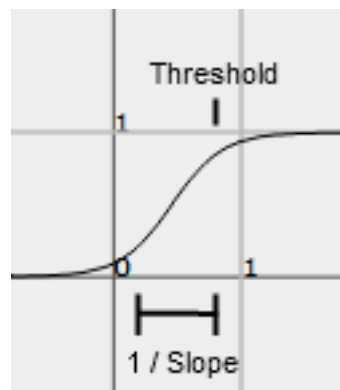


Figure 7.7: Sigmoid Function

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 7.8: 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.

7.3.2 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 respectively.

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




Figure 7.9: Narrow Oscillator

Label

The oscillator's label.

Output Value

The oscillator's output value. 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.

7.3.3 Narrow Excitatory Link

A narrow excitatory link transmits activation in one direction.

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

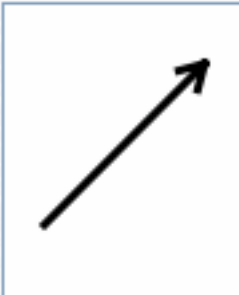


Figure 7.10: Narrow Excitatory Link

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.

7.3.4 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, excitatory link, abstract link, or another inhibitory link.

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




Figure 7.11: 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.

7.4 Misc

7.4.1 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.

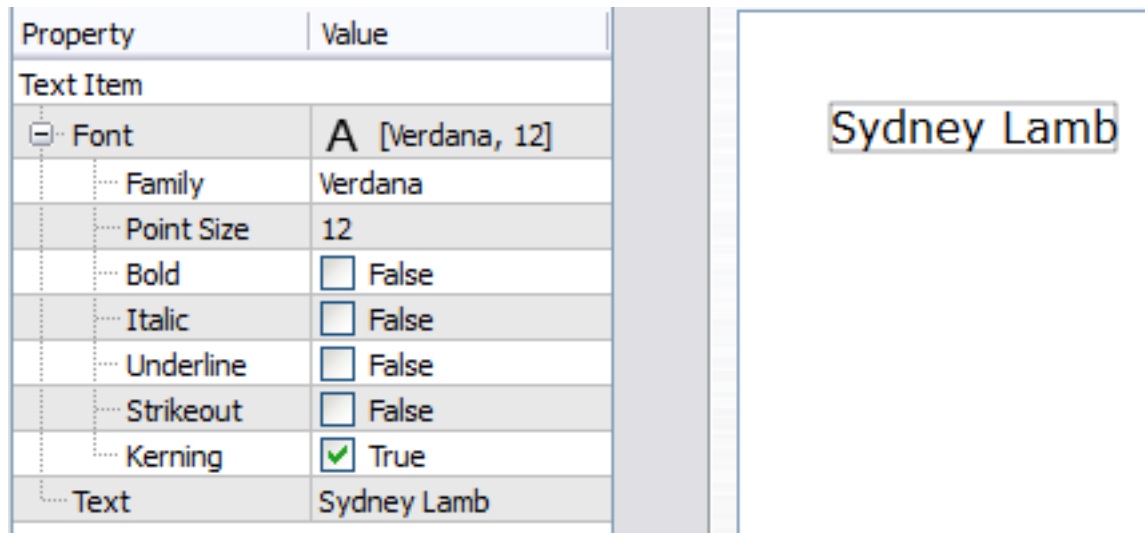


Figure 7.12: Text Item

Font

The font in which to display the text.

Text

The text to display.

7.4.2 Sub-Network Item

A sub-network item

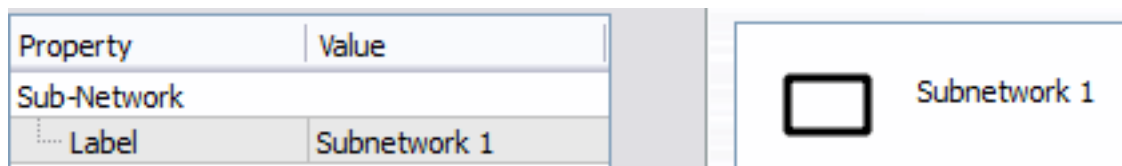


Figure 7.13: Sub-Network Item

Label

The subnetwork'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, but connected in a directed graph).

The automaton is updated using Nehaniv's asynchronous algorithm [Nehaniv2003]. 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.

A.1 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:

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

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

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

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:

A.1.1 Nodes

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

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

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

- $s = \frac{6 - \ln(1/0.99 - 1)}{r}$

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 **threshold** d is the point at which the output reaches close to 1.

A.1.2 Oscillators

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

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

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

A.1.3 Excitatory Links

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

$$x = hclip\{0, vw, 1.1\}$$

where w is the weight of the link.

A.1.4 Inhibitory Links

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

$$x = hwclip\{0, v, 1\}$$

where w is the weight of the link, which is always -1 for inhibitory links.

A.1.5 Link Learning

After each time step, the weight of a link is modified using Hebbian learning using the covariance hypothesis [Haykin1998], p79:

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

The change in weight Δw is equal to the **link learning rate** η , multiplied by the difference between the link's output x and its running average, multiplied by the difference between the link's target's output y and its running average. 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.

A.1.6 Node Learning

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

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

where l is the **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.

A.2 Implementation Details

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

A.2.1 Narrow Items

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

A.2.2 Abstract Items

Abstract items contain two paths of automaton cells, one for each direction of activation.

A.3 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 excitatory 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.

A.4 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 Lab

Copyright (c) 2010, 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. *Purennet: 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.
-