Charlie Mistrata

5/1/18

cpm2

Modeling of C. Elegans Nervous System

Overview

For my project I created a model of the connectome of the C. Elegans nervous system in Python. The python code allows you to activate various neurons of the nervous system by name and view the neurons and muscles that fired in response.

Background

Caenorhabditis elegans is an animal with an extremely simple and deterministic nervous system. For this project, the hermaphrodite C. Elegans worm’s nervous system was modeled. Each hermaphrodite C. Elegans grows up to have exactly 959 cells, of which 302 are neurons. The neuronal wiring of the C. Elegans is preprogrammed in its DNA, and the neural connectivity of each hermaphrodite C. Elegans is exactly the same. Because of this, C. Elegans has been an ideal subject of research for neuroscientists. Neuroscientists have been able to completely map out the cell structure of the hermaphrodite C. Elegans, naming every cell and discovering how the cells are connected.1 The C. Elegans is the only animal to have its nervous system completely mapped out,2 and the connectome for the C. Elegans nervous system is what is modeled in this project.

While the C Elegans simplicity makes it a suitable subject for research, its nerves have a key difference from human nervous systems: they do not fire action potentials. While human nerves fire in constant magnitude action potentials that occur in an all or none response to input, C Elegans nerves fire graded potentials that vary with both time and the magnitude of the summation of input.2 Additionally, partly because of its small size, the C. Elegans nervous system is a small world network,2 with almost every neuron being able to be reached by every other neuron by traversing a small number of connections. The project reflects these differences by having signals decay at a constant rate as they travel through the nervous system.

Data Used

This project uses neural connectivity data by L.R. Varshney, B.L. Chen, E. Paniagua, D.H. Hall and D.B. Chklovskii. The data can be found at the [Wormatlas database](http://www.wormatlas.org/neuronalwiring.html). Specifically, the excel files NeuronConnect under section ‘2.1 Connectivity Data’ and NeuronFixedPoints under section ‘2.3 Neuron Connections to Sensory Organs and Body Muscles’. For the project, the excel files were converted into csv files with the header lines removed. Additionally, what I believe to be an erroneous data entry in NeuronConnect (one entry had lower case letters for the neurons and these neurons were only connected to eachother) was removed.

NeuronConnect contains connectivity data for all the neurons in C. Elegans. A more in depth explanation can be found at the website, but the dataset essentially lists the connections of the nervous system in the form [neuron 1, neuron 2, type of connection, number of synapses between pair].

NeuronFixedPoints describes neuronal connections between sensory organs and neurons and between neurons and muscles. Again, more information is at [Wormatlas database](http://www.wormatlas.org/neuronalwiring.html), but the dataset essentially contains the connections as [name of neuron, sensory organ/name of muscle, spatial location of connection on sensory organ/muscle, weight of connection]. NeuronFixedPoints was used in conjunction with NeuronConnect to discover how the neurons were connected to each other and to muscle cells. However, how the neurons were connected to sensory organs was not modeled.

While the data provided gave a comprehensive overview of which neurons were connected to each other, the type of synapses between them, and the number of synapses between them, it did not describe in too much detail the specifics of the connections between the neurons and the firing dynamics between them. Because of this, in the simulation, various neuronal firing parameters (gradient potential decay rate, threshold value, etc.) are left as command line arguments for the user and are set to the same value for each neuron rather than on a neuron by neuron basis. Additionally, factors such as temporal and spatial summation were ignored, as they would be complex to code and could not be coded accurately with the insufficient level of detail in the data given.

What the Simulation Does

The file ‘CESimulation.py’ can be run with a command line argument specifying the neuron you want to have fire, and various program parameters (see the README.txt file). When the first neuron fires, it queues up all the neurons and muscles it has outgoing connections with to fire on the next step. The next step, these neurons fire, and all the neurons and muscles that they have outgoing connections to fire. As output, the program will show which neurons fired during each step of the program. When all the signals decay below a firing threshold, the program outputs some interesting statistics, such as the number of muscles that fired and their names.

Details of Implementation

The program parses the NeuronConnect and NeuronFixedPoints csv files to create a list of Neuron objects and Muscle objects. Each Neuron object has a name and a list of outgoing Connection objects. A Connection object consists of the name of the neuron on the receiving end of the connection, the type of connection (electrical, chemical, etc.), the number of synapses in the connection (although this value is ultimately not used in the program), and the weight of the connection, which is only a non-1 value for neuromuscular junctions between a Neuron and Muscle

When the program is ran, whatever Neuron the user specifies to fire will fire a gradient potential by calling its fireGP method with a specified signal strength as its parameter. This method will send signals to all the Neurons in the firing Neuron’s outgoing Connection list with the strength (strength of original signal \* decay rate of synapse type), with chemical and electrical synapses having different rates of signal decay. Although the amount of synapses between the sending and receiving neurons should affect the received signal strength, the received signal strength is already so simplified that this was ignored. Additionally, because the connectome is a small world network, multiplying the signal strength by the amount of connections led to infinite recursive loops. After the Neuron is done sending its signals, each Neuron that received a signal will call its fireGP method with the signal strength received as its parameter (as long as the signal received was above the user specified firing threshold). This continues until all the signals decay below the threshold.

The program outputs some statistics based on the parameters you input. If you do not tell the program to fire all the neurons, the program will print info about which neurons fired each round and which muscles were activated. If you tell the program not to have signals decay and also have each neuron fire a maximum of one time, the program will print which neurons are not reachable from the original firing neuron. If you tell the program to fire all the neurons, it will print info about which neurons fired the most and least. Finally, the program will always print info about the amount of neuron fires and muscle activations that occurred over the course of the program.

Conclusions from program

Looking at the neural network as a directed graph, almost every neuron in the C. Elegans connectome is reachable from every other neuron. However, the neurons IL2DL, PVDR, IL2DR, PLNR, and VC06 only receive input from sensory organs and so aren’t reachable from any other neuron but can reach every other neuron. Additionally, neurons VC06 and DD06 only synapse onto muscle cells, so themselves cannot reach any other neuron but are reachable from every other neuron. (Run the program with python CESimulation.py ALL TRUE to see, and CESimulation.py DD06 TRUE and CESimulation.py VV06 TRUE to verify)

With varying firing parameters, AVAL is usually the neuron to fire the most often when ALL the neurons are fired. (To clarify, firing ALL the neurons fires the first neuron, waits until all the signals die out, and then fires next one and so on) This implies AVAL, looking at the neural network as a directed graph, is highly connected and very close on average to any other neuron (where closeness is defined as how many edge traversals it takes to reach one neuron from another). Additionally, IL2DL, AINL, and BDUL usually fire the least (with ties resulting in the first alphabetically to display as the least). IL2DL must be the least, as it is not reachable from most other neurons and is in the first alphabetically in the group of non-reachable neurons. However, when the electrical decay rate is set to 0 (signals instantly decay when sent through electrical synapses), AINL is the least, showing that it is only receives electrical synapses, and, when the chemical decay rate is set to 0, BDUL fires the least, showing it only receives chemical synapses.

References

1.“Post-Embryonic Cell Lineages of the Nematode, Caenorhabditis Elegans.” *Egyptian Journal of Medical Human Genetics*, Elsevier, 1 Dec. 2004, [www.sciencedirect.com/science/article/pii/0012160677901580?via%3Dihub](http://www.sciencedirect.com/science/article/pii/0012160677901580?via%3Dihub).

2.Watts, Duncan J., and Steven H. Strogatz. “Collective Dynamics of 'Small-World' Networks.”*Nature News*, Nature Publishing Group, 4 June 1998, www.nature.com/articles/30918.

3. “Voltage-Gated Sodium Channels as Therapeutic Targets.” *Egyptian Journal of Medical Human Genetics*, Elsevier, 13 Nov. 2000, [www.sciencedirect.com/science/article/pii/S1359644600015701?via%3Dihub](http://www.sciencedirect.com/science/article/pii/S1359644600015701?via%3Dihub).