Patrick Farley
Project 6

Detecting Cycles in a Simple Neural Network

I came across a data source listing all of the connections in the neural network of a *C. Elegans* roundworm. I was very interested in the topic, and came up with the question of whether a cycle could exists in a neural network. I don't know much about neuroscience, but I know that neurons send "chain reactions" of electrical signals to the other neurons they're connected to. So, I wondered if a simple cycle in a neural network would actually be possible, or if it would cause a perpetual firing of a particular group of nerves (which would probably be detrimental). If I did find cycles, I would try to learn more about what they mean and why they occur. The data itself, although not very large, was far too large to visually check for cycles, so I felt that this would be a reasonable and valid application of graph theory.

The first task was to create a graph from the given data. The data was originally found on a webpage, but I opted to download it as a .txt file for simplicity. The file listed 3 space-separated values per line: the starting vertex, the vertex it pointed to, and the number of connections there (essentially the weight of the edge, but weight was not relevant for my particular focus). I parsed this data into an array of ArrayLists (adjacency lists), storing each vertex id as an integer (as in the original data). Because of its standard space-separated format, and because I made the third-column input optional, I was satisfied with the versatility of my graph-building code to be able to run on other data sources.

Next, I make a "CycleDetector" class and wrote a basic method for detecting whether a cycle exists in the graph reachable from a given vertex. It did a depth-first (recursive) search into the graph reachable by a given input vertex. It used a tracker array to mark which vertices had been visited before. Also, since this was a directed graph, I learned that I needed a second tracker to march which vertices were a part of the *current call* on the method (otherwise, a vertex that is visited twice could

falsely trigger a cycle detection (pointed to by two different verts: not a cycle). Cycles were detected, for many different vertices, in fact. So, I concluded that cycles *are* allowed to exist in a neural network (knowing as little as I do about neuroscience, that was noteworthy).

This was still very little information, however, so I set to rewriting the method to tell me more. I wanted a function that met the following three specifications:

1) Only detect cycles that *contain* the given node (not merely reachable by it).
2) Only list the first cycle found.
3) Keep track of the path around the cycle.

To handle 1) and 2), I had to change the structure a little. Originally, when any vertex had shown up as "already marked", it was then looked for in the "in current call" array. If found there, it would trigger "cycle detected." I wanted it to only detect a cycle when the *original vertex* was found again. So, I added a second argument and updated the method accordingly.

For requirement 3), I experimented with a few options before deciding it would be easiest to display the cycle sequence as a string, built by appending the vertex identities at each level of the recursion stack. The end result was a space-separated string of integer values that traced a cycle backwards from the last vertex in the cycle to the original vertex.

I then refactored the code a little so that it could be more simply controlled from the "Driver" class. The only user input is the "original vertex." So, for a given neuron id, the user could easily find out whether it is part of a cycle and see the path of such a cycle.

I then set out checking the cycles for different vertices, looking for patterns. There was a *lot* of information available to me now, so I tried to keep my questions modest for the sake of time. I found that most neurons were part of cycles, but a few were not. I examined these and tried to do a little research on their function. Two such "terminal" neurons were directly involved in receiving signals from outside the organism (Oxygen levels in the air). Another two terminal neurons were said to take

input from a number of skin-layer “touch-sensing” neurons and integrate these multiple signals into a chosen behavior (determining which direction to propagate the signal, I guess). As a side note, the reason they came in pairs was because the roundworm has mostly symmetrical neurons on its left and right side. I eventually realized I was not going to find an easy pattern for why some neurons are in cycles and some are not. Still, I’m happy with the role my program played in making all of this information available to me.

Neuron Information Resource:

Altun, Z. "Individual Neuron List." Individual Neuron List. WormAtlas, n.d. Web. 28 Apr. 2015. <http://www.wormatlas.org/neurons/Individual%20Neurons/Neuronframeset.html>.