

Modern Robots: Evolutionary Robotics

Programming Assignment 2 of 10*

Description

In this assignment you will be creating an artificial neural network (ANN). There are many kinds of ANNs, but they all share one thing in common: they are represented as a directed graph in which the nodes are models of biological neurons, and edges are models of biological synapses. Henceforth, the terms *neurons* and *synapses* will be used to describe the elements of an ANN. The behavior, in addition to the structure of an ANN is also similar to biological neural networks: (1) each neuron holds a value which indicates its level of activation; (2) each directed edge (synapse) is assigned a value, known as its *strength*, which indicates how much influence the source neuron has on the target neuron; and (3) the activation a of a neuron i at the time step $t + 1$ is usually expressed as

$$a_i^{t+1} = \sigma\left(\sum_{j=1}^n w_{ij}a_j^t\right) \quad (1)$$

where there are n neurons that connect to neuron i , a_j^t is the activation of the j th neuron at time-step t that connects to neuron i , w_{ij} is the weight of the synapse connecting neuron j to neuron i , and $\sigma()$ is a function that keeps the absolute value of the activation of neuron i from growing too large.

In this project you will create an artificial neural network in C++, simulate its behavior over time, and visualize the resulting behavior.

Tasks

1. Copy all your code for assignment 1 to the folder you will use for assignment 2. Make sure to keep a working backup of assignment 1.
2. Extract `hw_2_code`, and move `Ind_NeuralNetwork.hpp`, `Ind_Neuron.hpp`, and `Ind_Connection.hpp` to your assignment 2 folder.
3. Open `Ind_Neuron.hpp` and `Ind_NeuralNetwork.hpp` and implement all functions marked `TODO: YOUR CODE HERE`. Make sure to test each function after you have implemented it, so you know that it works correctly. The network should be a fully-connected, recurrent neural network, where the activation function is the hyperbolic tangent. Note: you are allowed to extend the `Neuron` and `NeuralNetwork` classes with your own functions for your convenience.

*Original material was graciously provided by Josh Bongard. Jeff Clune slightly modified it. Joost Huizinga then heavily modified it.

4. Comment out, or delete, all the homework 1 code in `main.cpp` (but remember to keep a backup).
5. First, we will check that our neural network is working correctly by creating a benchmark neural network for which we know the correct output. To make our neural networks more general for future assignments, we will use template programming¹. As a result, you will have to define a new network like this: `NeuralNetwork<> myNetwork;`. Now, create a neural network with 10 neurons and initialize it as follows:

Listing 1: Initialization for the benchmark network.

```
for(int i=0; i<numberOfNeurons; ++i){
    for(int j=0; j<numberOfNeurons; ++j){
        network.addConnection(i, j, i%3-1);
    }
}

for(int i=0; i<numberOfNeurons; ++i){
    network.setValue(i, 1);
    network.getNeurons()[i].setBias(i%3-1);
}
```

This creates a network where the connections and biases from each neuron are either -1 , 0 , or 1 . In addition, it initializes every neuron with an activation value of 1 . Write the network to a file with:

```
std::ofstream networkFile("hw2_network_test.csv");
networkFile << network << "\n";
networkFile.close();
```

Plot the network with the provided `plotNetworkCircle.py` Python script. If your network file is called `hw2_network_test.csv`, you can plot your network with:

```
python plotNetworkCircle.py hw2_network_test.csv
```

The result should be identical to Fig. 1. If it is not, there is some error in the way you construct the network, and you should try to fix the issue before moving on. Once the network looks as expected, you should include this figure into your document.

6. Subsequently, run the network for 50 time steps, and write the activation of the neurons to a single file. Call the `step` and `logActivation` methods you implemented to run the network and log its activation. Note: the resulting file should contain 50 rows, where each row consists of 10 numbers. Each row represents the activation of the network at a single time-step.

7. Plot the resulting file with the provided `plotMatrix.py` Python script. This should produce the same image as shown in Fig. 2. Include the image into your document. If your image does not look like Fig. 2, you have an error in your neural network, and you should try to resolve the issue before moving on.

8. Randomize the neural network from above, such that each weight and each bias is assigned a random number between -1 and 1 (inclusive). Write the randomized neural network to a file and plot the result. The resulting image should look like, but *not* be identical to, figure 3. Add this figure to your document.

9. Initialize the current activation value of each neuron to a random number in $[0,1]$, run the network for 50 time-steps, and write the activation of the neurons to a single file. Call the `setValue` method of a neuron

¹If you are unfamiliar with C++ template programming, please consider doing some online tutorial such as this one: http://www.tutorialspoint.com/cplusplus/cpp_templates.htm

to set its current activation value, as shown in listing 1. Once again, write the activation of all neurons to a file, as explained in step 6, and plot the result with the `plotMatrix.py` Python script. This should produce an image similar to that shown in Fig. 4. Include the image into your document. The image does not need to look exactly like that of Fig. 4. In fact, re-run your program several times, and compare the images produced. Notice that the patterns of neuron activation vary greatly from one run to the next.

10. Things to think about: Why do some of the neurons oscillate like this? What happens when all of the synaptic weights are set to -1, or to zero, or to 1? After you think about it, try it and see if you were right! You do not need to include your answers in the document you submit for grading.

11. More things to think about: what is lacking in this visualization? Are all of the connections shown? Which types are not? Can you think of a better way to show all the connections? Can you implement it? This is not required, but if you do implement a nice way to show all the connections, please include that as an **additional** plot in your submitted PDF and I'll give you extra credit if I like it. Make sure to include a caption that describes how the information is shown.

Deliverables

A pdf document containing the figures resembling figures 1 to 4, and a zip file containing your modified source and header files.

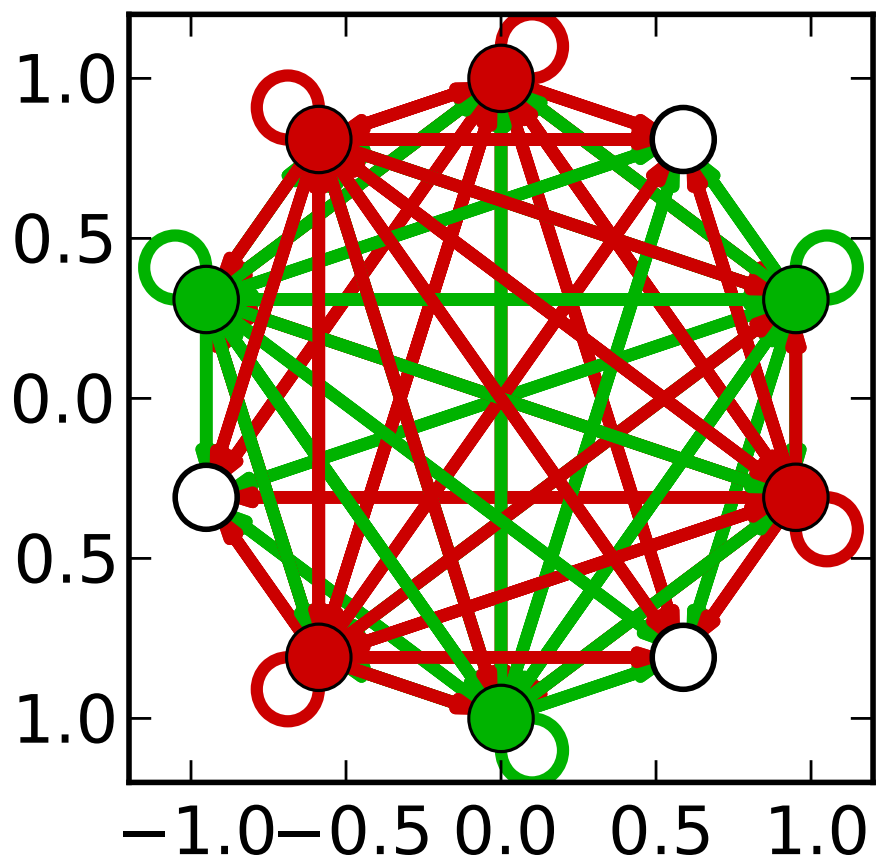


Figure 1: The benchmark ANN, where red and green lines indicate synapses with negative and positive weight, respectively. Your figure should be identical to the one shown.

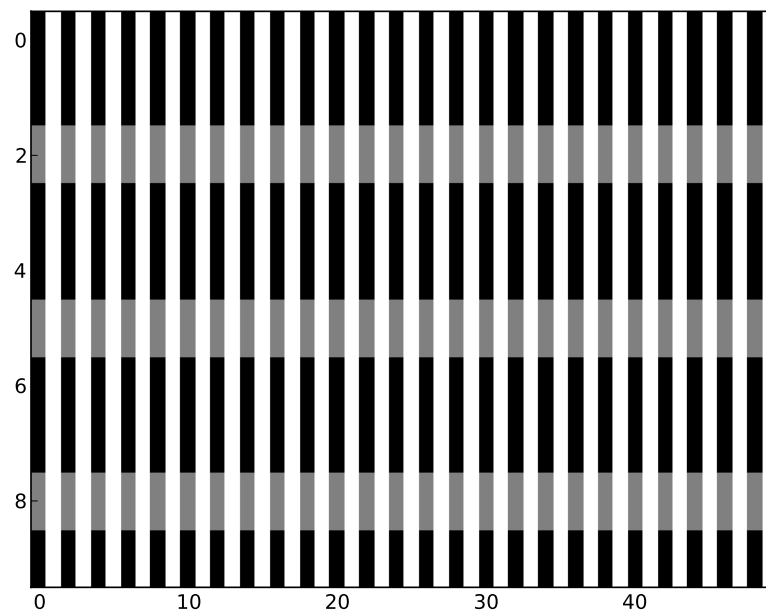


Figure 2: The activation of the benchmark ANN. Your figure should be identical to the one shown.

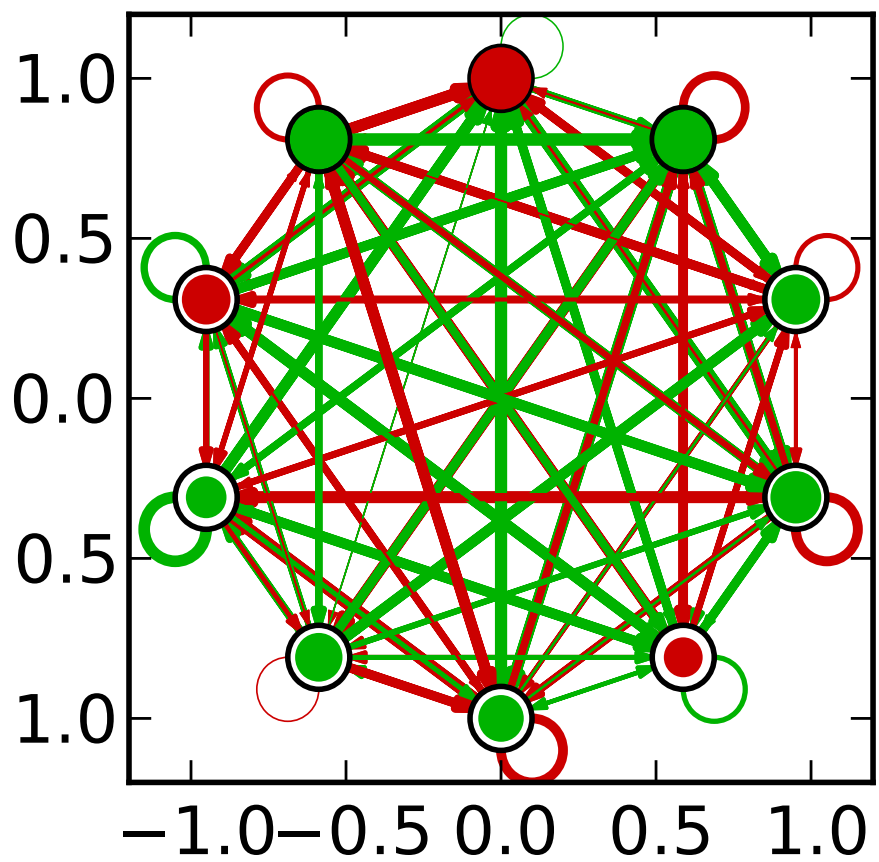


Figure 3: An ANN with 10 neurons and $10 \times 10 = 100$ synapses, where red and green lines indicate synapses with negative and positive weight, respectively.

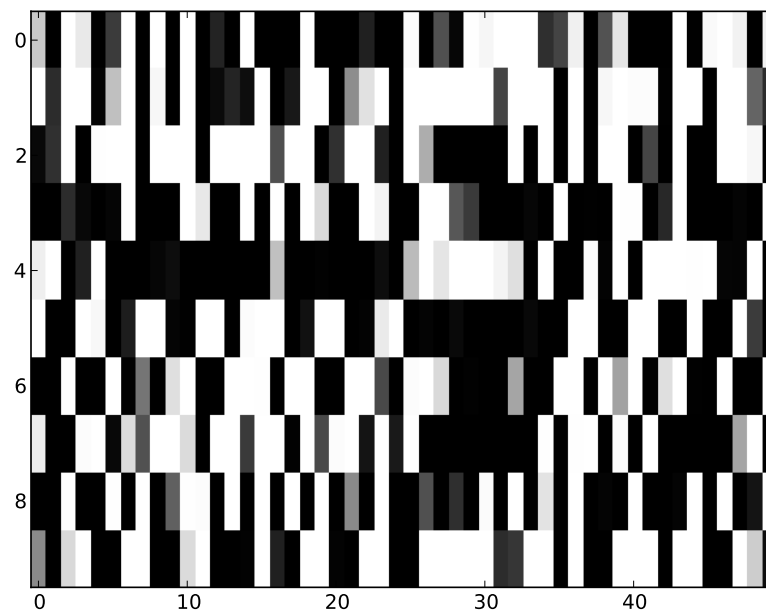


Figure 4: The activation of a network with 10 neurons over 50 time-steps.