Evolved Dynamical Neural Network

by: Garrett R and Andreea G

Abstract

This work is an extension of the work done by Randall Beer and Brain Yamauchi in the paper Sequential Behavior and Learning in Evolved Dynamical Neural Networks. We implemented our networks in a very similar way with some small modifications, and attempted to scale the result up in order to solve a useful task. The first task we have designed is to solve a non-trivial maze in the least amount of steps.

Dynamical Neural Network

Each brain is made up of N neurons. Each neuron can connect to any other neuron (these connections are called *synapses*) including itself. Each neuron is not necessarily connected to every other neuron.

At any given time, neuron i has an activation $y_i(t)$. The differential equation describing how the activation changes is

$$\frac{dy_i}{dt} = -K_i y_i + \sum_{j=1}^{N} w_{ji}\sigma(y_j - \theta_j)$$

where K_i is the rate of decay, w_{ji} is the strength of the synapse leading from j to i, $\sigma(.)$ is the activation function for neuron j which we have set to be the unit step function, i.e.

$$\sigma(x) \quad = \quad \begin{cases} 1 & , x > 0 \\ 0 & , x \leq 0 \end{cases}.$$

After one time step dt (or "cycle"), the new activation $\widetilde{y_i}$ is

$$\widetilde{y_i} = y_i - K_i y_i dt + \sum_{j=1}^{N} w_{ji}\sigma(y_j - \theta_j)dt$$

Also, we're now setting dt to be 1 in order to get a faster brain. Lastly, if a neuron fired during the current round, it is depleted and gets set to zero at the end of the round. This helped to create a more dynamic brains (i.e. a brain which doesn't get stuck in a steady state solution)

Input and Output to the brain

Each brain goes through a bunch of *decision processes*. In each decision process, the brain is given input (which is dependent on the task) and then output is read from the brain. There may also be deadtime between the decision processes where the brain cycles in the absence of input given or output read.

The input and output of a brain must be in binary form. If it takes 5 bits to encode the input, and suppose the first input the brain receives is $\{0, 1, 0, 0, 1\}$,

then the second and fifth neurons' activations are held at 1 (the maximum activation), while the first, third and fourth neurons' activations are held at 0 (the minimum activation). This input is given over a number of cycles specified by the user.

The output is taken from the neurons immediately after the input neurons in the neuron sequence. So a brain that has 5 input neuron and, say 3 output neurons, will have output neurons at position 6, 7, and 8. The output is read by "listening" to the neurons for a specified number of cycles, i.e. if neurons 6 and 8 fire, then the output is considered to be $\{1, 0, 1\}$. Unsurprisingly, the brain gives many different outputs during the output period. We therefore look at the most common output and if it is valid, that is taken to be the final output of the brain (for that decision process). If that output was invalid we look at the second most common output, and so on. If the brain fails to give any valid decision, then it is automatically assigned the worst fitness score possible (as if it had taken the maximum number of moves allowed).

For the case of the maze task the brain is given 3 bits of input (to the first 3 neurons). The first indicates if a left-turn is possible, the second is for the straight option, and the third is for the right-turn. For example, if the input was [1,1,0]. This means that only a left turn or straight forward are the options.

The output of the brain is 2 bits. The first bit says if the brain wants to go straight, the second decides between right and left turns. So, the possible inputs and their corresponding meanings are: [1,1] and [1,0] -> Stay pointed straight; [0,0] -> turn left, [0,1] -> turn right.

Genetic Algorithm

During each generation, every brain is assesed for its *fitness score*, which we define to be the inverse of the number of steps that brain took to solve the maze task. The next generation is then constructed. Each member of the next generation has a certain probability (specified by the user) of coming from sexual or asexual reproduction.

In the case of asexual reproduction, a specified number of neurons each have a specified number of synapses mutated (i.e. the parameter w_{ji} is reset to a random number).

In the case of sexual reproduction, a random number x between 0 and the number of neurons in the brains is chosen. The child brain then inherits the first x neurons from the first parent and the remainder from the second parent. Note: a synapse is considered to belong to the neuron from which it originates.

Creating Mazes

We wrote a helper script to create mazes (maze_files/png_to_maze.py), which takes a PNG image and converts each pixel into a square on the maze. Type "./png_to_maze.py –help" to see documentation on using this script.

Executing the code

You should execute the helper Bash script which feeds all the necessary parameters to the program. Open this script and edit all the parameters, then

execute it.

Reading the code

For those interesting in reading through our code, here are some helpful tidbits.

In the code: y_i is Neuron::activation_,
w_{ji} is Brain::neurons_[j].synapses[i]),
K_i is Neuron::decay_rate_,
θ_i is Neuron::acitve_threshold_,
dt is globals::TIME_STEP,
$\sigma(.)$ is Neuron::ActivationFunction(.).

Results

We encountered an interesting problem while trying to mutate generations. We found that mutating a single synapse of a single neuron causes a brain which performed very well to suddenly become a poor performer. Obviously, such behavior means that our genetic algorithm would have no hope of working. This suggests that we're missing some very basic feature that real brains have since if an animal has a change of a single synapse, their behavior doesn't completely change.

If anyone has any ideas, please let us know!