**Goal**

Use a genetic algorithm (GA) to create a population of neural networks (NN) that will play snake. The NN’s objective is to score as many points as possible.

**Game parameters**

* Snake will start with length 2.
* Snake will start in a random position that is more than 1 position away from any wall.
* Snake will start with a random direction.
* Fruit will spawn in a random location that is not occupied by the snake’s body.
* Snake will have a timer measured in ticks.
* When the timer reaches 0, snake dies.
* If snake head collides with its body, it dies.
* Timer starts at 500.
* Timer increases by 200 when fruit is eaten.
* Timer has an upper limit of 10,000.
* Board will be 30x30.

**Inputs relative to head (0 for N/A)**

* X distance to fruit
* Y distance to fruit
* +X distance to wall
* -X distance to wall
* +Y distance to wall
* -Y distance to wall
* +X distance to body
* -X distance to body
* +Y distance to body
* -Y distance to body

**Outputs**

* Up
* Down
* Left
* Right

**Hidden layers**

* 2 layers of 6 neurons

**Initialization**

* All weight values will be between -2 and 2.
* All bias values will be between -5 and 5.

**Definitions**

Neuron - A node in the network.

Weight - The influence a neuron has to another connected neuron activating.

Bias - An adjustment of how much the value of a neuron requires before activating.

**Define variables**

y - Desired output

a - Neuron output

z - Neuron output before squash

w - Weight

b - Bias

C - Cost

L - Layer

j - Neuron index

nL - Number of layers

nj - number of neurons in a layer

F - Fitness

P - Points

T - Ticks passed to get point value

r - Run number

**Define functions**

***Fitness***

This is meant to encourage obtaining the most points within the least amount of time.

***Sigmoid***

This function is meant to squash the input into a useable neural value between 0 and 1.

***Neuron activation***

This function calculates the weight and bias of a neuron to a single variable.

Calculates the neuron output given all neurons attached to it.

This will use the sigmoid function to squash the result to a value between 0 and 1.

**Procedures**

***Population***

* Each generation will have a population of 1000.

***Training***

* Run 5 games and average the fitness values for an overall fitness.

***Crossover***

* Select the 15% fittest members, and 5% randomly from the population for crossover.
* Randomly select 1 neuron, swap all weight/bias values associated with the opposite parent.
* Crossover 10 children, train them, and only select the fittest.

***Mutation***

* Mutation rate is 2%.
* Weight mutation adds a random value between -1 and 1.
* Bias mutation adds a random value between -1 and 1.