Bug Race using a genetic algorithm

Yannick Daniel Gibson

ČVUT–FIT

gibsoyan@fit.cvut.cz

May 1, 2022

1 Introduction

Genetic Algorithm (GA) is a metaheuristics which is inspired by real world process called natural selection. It belongs to a larger class of evolutionary algorithms one that measures fitness for each genome, which then, depending on the configuration attributes, determines how the next generation is going to look like. In our report we will be focusing on NeuroEvolution of Augmenting Topologies (NEAT).

2 Definition of the problem

The goal of the program is to make a race in a 2D plane, where with the help of *neat-python* (python module) agents will compete at who can drive faster through a road, in the correct direction and without hitting the sides of the road. The program can be run in *AI mode* that simulates the genetic algorithm and *movement mode* which demonstrates car controls.

3 Graphics

We are using *pygame* (python module) to draw necessary components of the simulation. The agent images were drawn in a pixel editor not wider than 50 pixels. Roads were created using a road-creating utility which I have created that is not included in this project version; It created an array of length 2 which contains array of points that define the road as a polygon. Related to that, fitness lines, which define how the fitness should be calculated, they are defined in points of the road.

4 AI Parameters

In a AI simulation we are able to turn on and of a couple parameters

- agent vision
- active fitness lines for each agent
- all fitness lines
- highlight parent
- save best genome
- reverse road direction
- allow user to drive

5 Numerical Parameters

Those are defined for *AI simulation*. The *movement simulation* has only the *road index* parameter.

- road index
- max frames elapsed, to skip to next generation
- no progress frames elapsed, to skip to next generation when no progress is made

6 Algorithms

The following subsections will describe the core algorithms used ...

6.1 Movement

The agent movement is controlled by an acceleration property and a rotation method. The x and y velocity is calculated by the following formula (acceleration is either -1, 0, 1.).

$$xvel = xvel + \cos(angle) * maxspeed * acc$$

The drift angle is calculated by **L**inear Int**erp**olation function (LERP) that is restricted to the interval (-5, 5).

6.2 Vision

We project 8 line segments in appropriate angles to maximize the NEAT algorithm efficiency. The line segments intersect with the car center. Next we check if the line segment intersects the road (which is constructed as a polygon, therefore from lines) and return the length of the line segment which is constructed as (center, intersection), if no intersection is detected, then return a maximal fixed length which is defined as a constant.

$$vision = ||(center - intersection)||$$

6.3 Collision detection

First step is to roughly approximate the car as a rectangle. Next we use the same line intersection algorithm used in 6.2 and run it for each line segment of the rectangle. The rectangle vertices are firstly calculated by not accounting for the rotation, secondly getting the vector which points from center to the not rotated vertices, thirdly rotating the vectors and lastly adding center to the rotated vectors.

6.4 Genetic Algorithm

At the start of the algorithm we will pick n agents, with randomly generated weights. After each generation we pick the most fittest agents, where we clone and mutate their genomes. There are 10 inputs: 8 of them are *vision distances*, the other 2 are: *drift angle* and *speed*. We also have 3 following outputs: *front acceleration*, *back acceleration* and *rotation*. The neural network type is feed-forward. We are using 1 *tanh* activation function.

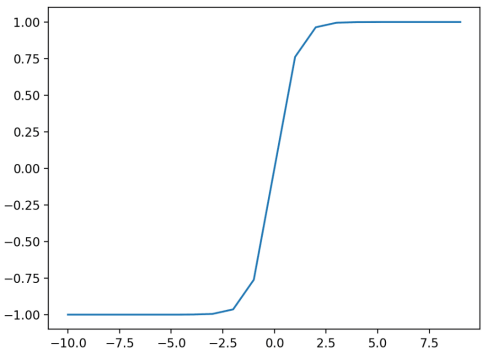

Figure 1: *tanh* activation function

7 Results

We can see that the NEAT algorithm is performing pretty well and in just a couple of generation is able to make agents navigate in the map. It also helps that the NEAT algorithm tries to preserve genetic variety, it is one of the tools that make the algorithm effective. There are also some flaws. If the road is too narrow or the road is created with fitness points too far from each other the net struggles to find the optimal neural net.

8 Conclusion

The program is visually displaying the power of genetic algorithms and how simple rules can have interesting results. The next versions of the project could include a polished subprogram for creating roads (which would include, creating it in a manner that optimizes NEAT performance). Also another subprogram for starting a generation based on a saved genome could be added. Frankly, some optimization would be handy. I believe the program has many uses one of them is the ability to be used in an open day for showcasing.

References

[1] Risto Miikkulainen Kenneth O. Stanley. Evolving neural networks through augmenting topologies. online, 2002. `http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf`.

[2] Lothar M. Schmitt. Theory of genetic algorithms. online, 2004. `https://www.sciencedirect.com/science/article/pii/S0304397503003931?via%3Dihub`.