Applying Neural Networks and Neuroevolution of Augmenting Topologies to play Super Mario Bros

Vivek Verma

7 June 2020

1 Abstract

This paper describes the background and implementation behind a project that uses Neroevolution of Augmenting Topologies (NEAT) to play Super Mario Bros. It's implementation is different from classic applications of NEAT since the training process was heavily optimized using multithreading and downsampling. As a result, the training process can be run on underpowered CPUs without the help of an external GPU. The neural network successfully completed level 1-1 of the game.

2 Background

Given a dataset with coordinate pairs (x, y), linear regression can be applied to find the best fit line and extrapolate values not within the dataset. Such a model can take the form of equation 1.

$$\hat{y} = b_0 + b_1 x \tag{1}$$

In equation 1, the explanatory variable is of dimension 1. For example, the explanatory variable of the model in equation 2 has dimension 3.

$$\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 \tag{2}$$

This can be generalized with the help of vectors and linear algebra. Define a vector **x** that contains every input and 1 for the constant term, and a vector b that contains every coefficient.

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \tag{3}$$

$$b = \begin{bmatrix} b_0 \\ b_2 \\ \dots \\ b_n \end{bmatrix} \tag{4}$$

Linear regression with multiple variables, multiple regression, can be generalized as a dot product between **x** and b.

$$\hat{y} = b \cdot \mathbf{x} \tag{5}$$

In matrix notation, this is a transpose of b times **x**

$$\hat{y} = b^T \mathbf{x} \tag{6}$$

To predict categorical variables, a sigmoid function (Fig-1) can be applied over the generalized multiple regression equation (6).

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

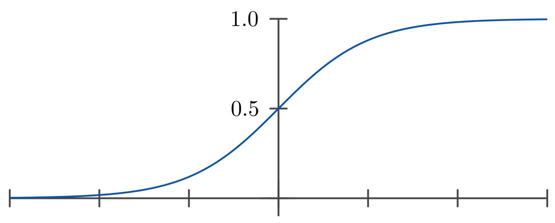
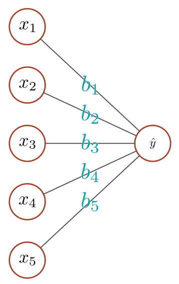


Figure 1: The Sigmoid Function

This results in the generalized logistic regression equation (7)

$$\hat{y} = \sigma(b^T \mathbf{x}) \tag{7}$$

Categorical variables with more than two possible values can be predicted with the use of multiple logistic regression models, however this project does not take advantage of this. Instead, the complexity of the model is increased by adding multiple logistic regression layers to create a neural network.



$$y = \sigma(b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_4)$$

Figure 2: Going from one logistic regression layer to another

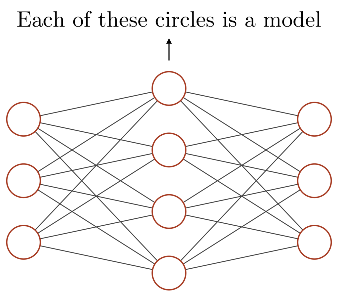


Figure 3: Diagram of a Neural Network

The coefficients b are typically found using backpropagation or gradient descent on a large dataset. However, this project utilizes a genetic algorithm to evolve the coefficients by maximizing a fitness function. Here, the fitness function is the distance covered by each of the networks on a simulation.

The way neuroevolution works is by initializing a population of random networks, then running each of them through a simulation (in this case, Super Mario Bros), choosing the top few networks from the population, and breeding a new generation using mutations. This process is repeated until the desirable network is achieved.

NEAT features two other optimizations: Augmenting Topologies and Speciation. The word "Augmenting Topologies" implies that the structure of the neural network, or the topology, evolves along with the coefficients.

Speciation is the set of processes by which NEAT creates, maintains and uses several disjoint groups of similar genomes for guiding reproduction.

3 Implementation

To create an interface between the game and the python code running Neuroevolution, a Lua script was used that provided 16x13 grids that represent the game (Fig-4).

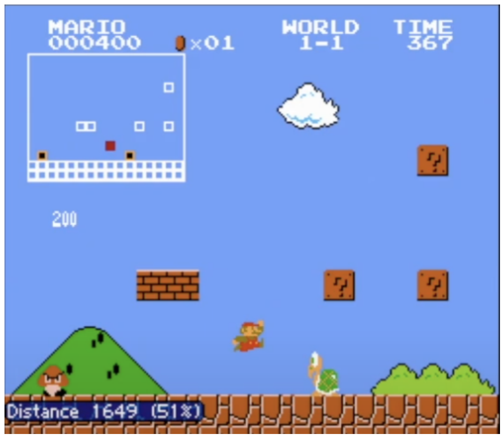


Figure 4: An example frame from the game and its compressed version in the top left

The compression of the image to a 16x13 grid reduced 61,440 inputs to 208. This enables it to run on lower end computers, but the training process would take a few weeks without any further optimization.

This problem was solved through taking advantage of multithreading. Using a python library called "multiprocessing", multiple networks of a population could be run at once.

For the training process of this project, 4 networks were run at once, to take advantage of the 4 threads on my laptop.

4 Results

After 2 weeks of training, the network was successfully able to complete level 1-1.

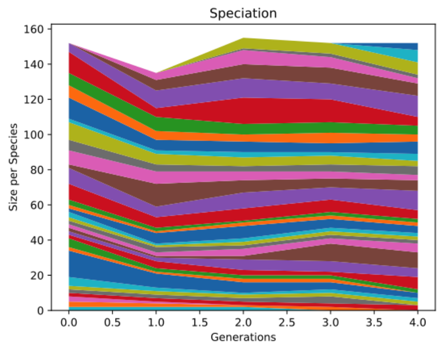


Figure 5: Speciation over generations

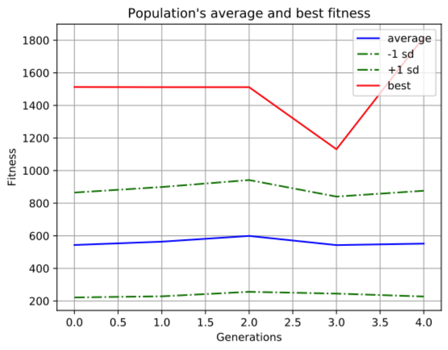


Figure 6: Evolution of fitness over generations

5 Conclusion

The results of this project are quite successful. Although it is unable to complete other levels, it is able to complete level 1-1. However, random movements within the game, like the movement of enemies causes some inconsistencies in the results.

Video explanation can be found at `https://youtu.be/JcxhGDdjCZ8` and source code here `https://github.com/vivek3141/super-mario-neat`