Space Invaders Genetic Algorithm and Neural Network

David Vanderzee and Collin Fingar
Game A.I.
December 10, 2015

Presentation URL:
https://docs.google.com/presentation/d/1U1pmxRb3xXOvuqLrztTexpvtNbONJxUDuwMbMej-vi8/edit?usp=sharing

To run Program:

Hit Play button. The score and generation is displayed on screen. If you are interested in seeing the values to the weights of the inputs, click the NN gameobject while the game is running. You can see the public values there being changed while the program is running. Scores are printed in the Debug Log.

Space Invaders

We made a lightweight version of the classic 1978 arcade game. The player is positioned near the bottom and can only move horizontally and shoot upwards. It's goal is to defeat as many aliens as it possibly can before it is either hit with an enemy shot, or the aliens get low enough and reach their objective. The player receives points for each enemy that it successfully shoots. When all of the on-screen enemies are defeated, they are respawned in their initial position. However, each iteration, their speed increases, thus adding to the difficulty.

After the game was created, we modified the player to be controlled by an A.I. The player mutates and evolves thanks to a genetic algorithm, and it's every move in-game is directly controlled by a neural network.

Genetic Algorithm

Through each generation of our A.I., it learns how to better handle different conditions to improve its fitness. The fitness of our player is calculated primarily on number of enemies defeated and secondarily by the amount of time it's alive.

When the player decides upon and does any action, an integer variable related to that action is increased. At the end of iteration, whether the player is hit with an enemy shot or the aliens reach the bottom, those variables are compared to those of the past iteration. The A.I. compares the fitness levels and modifies weights (and, therefore, likliness) of some actions based on if it did better or worse. The mutation rate for those weights is 10%.

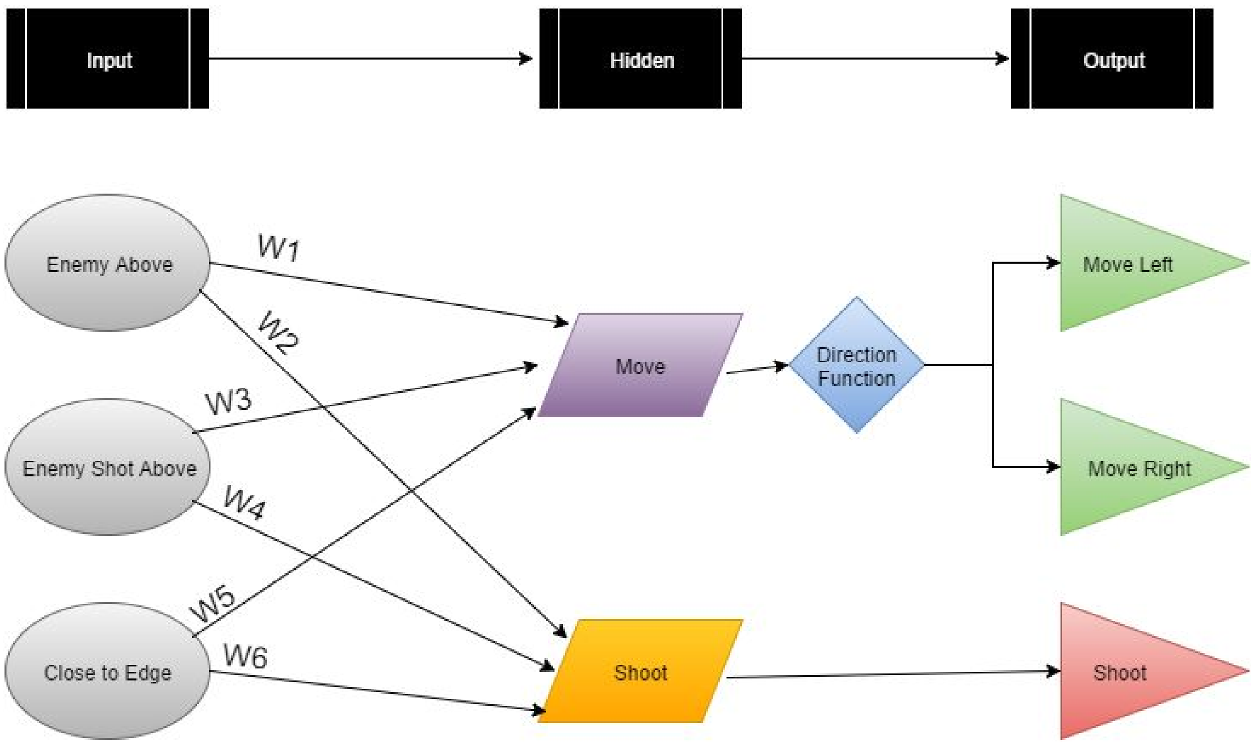
Neural Network

Our A.I. is directly controlled by a neural network. The network receives some inputs and makes a series of calculations based on them. Our inputs are whether or not there is an enemy above the player, whether or not there is an enemy shot above the player, and if the player is close to the edge.

In our hidden layer, we calculate a number of different values based on our input booleans. First, we add weights to two variables, Move and Shoot, which will decide the rest of the process. These weights range from the move weight to being close to an edge to the shoot weight if an enemy isn't above.

If Move is the higher value, we have to determine which direction to move. For instance, if an enemy is NOT above the player, it determines the direction of the nearest enemy. It also determines which edge it's closest to (if close to one). It adds the respective weights to move left or move right, and, depending on which is greater, chooses that as it's action.

Of course, if Shoot is higher than Move, our agent then decides to shoot instead of moving.



<u>Recombination</u>

After the A.I. loses the game in the current generation it works on adjusting its weight dependant on how it did in relation to the previous generation and what it did differently. Every time an output is given the NN stores what inputs were active when that output was taken. For example, if the player moved as an output, the NN documents that it moved when there was or was not an alien above the player. In the end we have how many times the player moved when there was an alien above, or a shot above, etc…

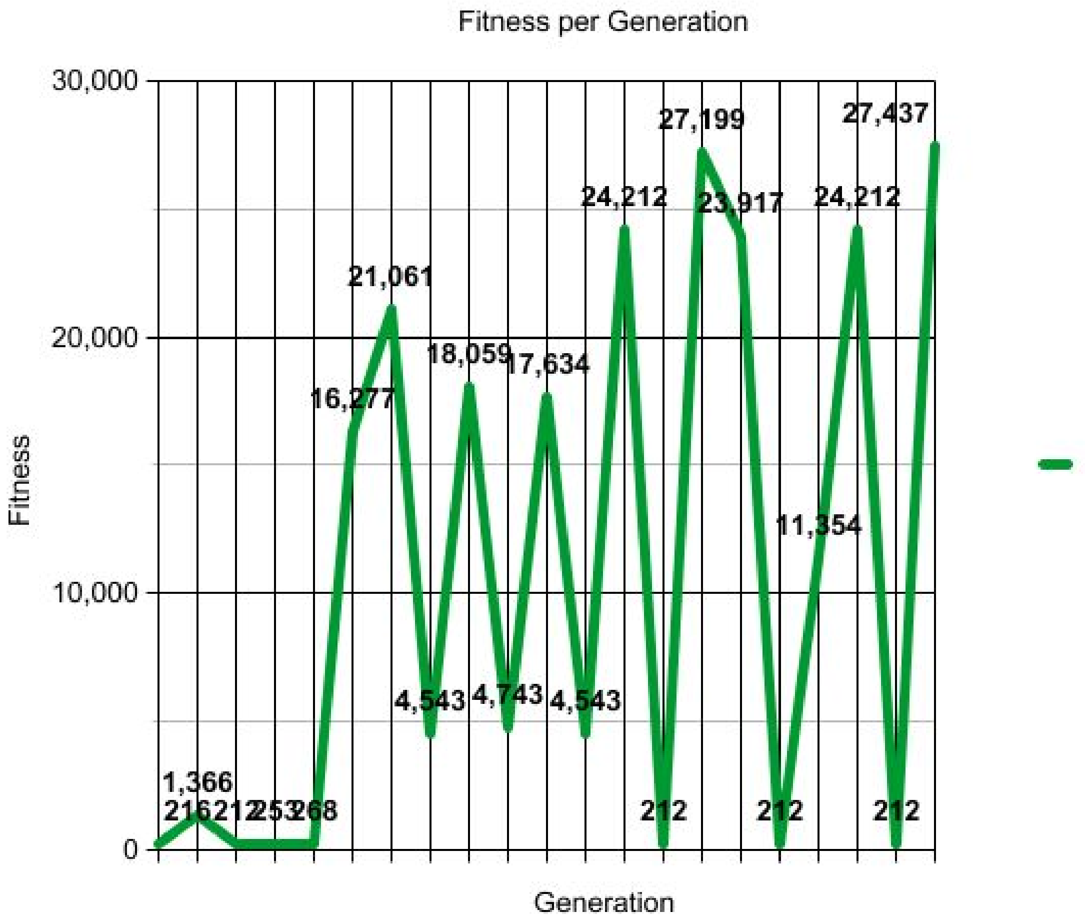
The NN compares all these stored values and the Fitness to the previous generation. If the previous generation moved a lot more when there was an alien shot above it along with living longer then that means the NN should adjust weights accordingly to move more in that occasion. Our player is mutated by 10% per generation.

Our A.I., because of this intelligent design is able to learn from its mistakes and adjust its way to a balanced play style that is able to avoid being shot by aliens and work fast enough to win before the time runs out.

<u>Analysis</u>

The first generation starts with equal weights across its inputs to give no specific advantage. Usually the A.I. will be shot quickly in the first generation. This will increase its weight to move away when there is a shot above it. From here the A.I. becomes very cautious and spends most of its time avoiding all conflict. This will result in the aliens winning due to hitting the bottom of the screen. The A.I. now knows it needs to be more aggressive. It goes back and forth through the generations tweaking all its weights until it gets to a balanced point where it's able to kill all the aliens and go to the next level.

Usually it takes the A.I. around 5 generations to get to the point where it can beat the first level of aliens. By the tenth generation it has found a good place where it doesn't make much improvements. The A.I.'s final weight values are exactly what we expected. Above all, it's the most important to dodge incoming bullets. Failure to do so results in the game being over. Second to that is to shoot at aliens above you when there is no immediate danger. We expected the A.I. to go seek out aliens when there are none available to shoot. Instead it plays it safe and waits for the aliens to come to it since it knows it can be patient.



As you can see from the graph, the A.I. starts off slow and then makes large leaps where it does very well. After that it has to do more experimenting before it gets a new fitness high-score. For 21 generations, it slowly hits a new high-score peak after two or so learning generations, eventually learning better and better every time.