Application of Neuro-Evolutionary Augmented Topology:

Using Genetic Algorithm to Evolve Neural Networks

Si Yuan Fang (u5838661)

Abstract –

This report assesses the value of combining Genetic Algorithm with Neural Networks using a method known as Neuro-Evolutionary Augmenting Topologies. This allowed minimal (no hidden layers) neural networks to gradually grow according to the environment they interact with. Eventually they will become complex enough to be useful for applications.

Specifically this report will evolve a neural networks with their performance accessed by pitting individuals against a traditional AI in a simple Pong game.

The initial attempt was unsuccessful, but with the use of several optimising strategies,
Through this experiment the effectiveness of the NEAT technique was shown.

Introduction –

Neural Networks has been proven to be an effective problem-solving tool. However their weakness is apparent. They require a substantial amount of data to train, and due to the nature of the gradient descent algorithm, cannot be trained unsupervised. In comparison, Genetic Algorithms are effective at interaction with an established environment. With its population-based search method it's less likely to be trapped inside a local minima (or local maxima). They are good for when the error rate cannot be seen immediately and require the individual to interact with the environment in real time. However their structure is often chaotic, and they can have difficulty reaching the absolute maxima due to the fact that mutation will change individuals in a random way, which can compromise its efficiency when not handled with care.

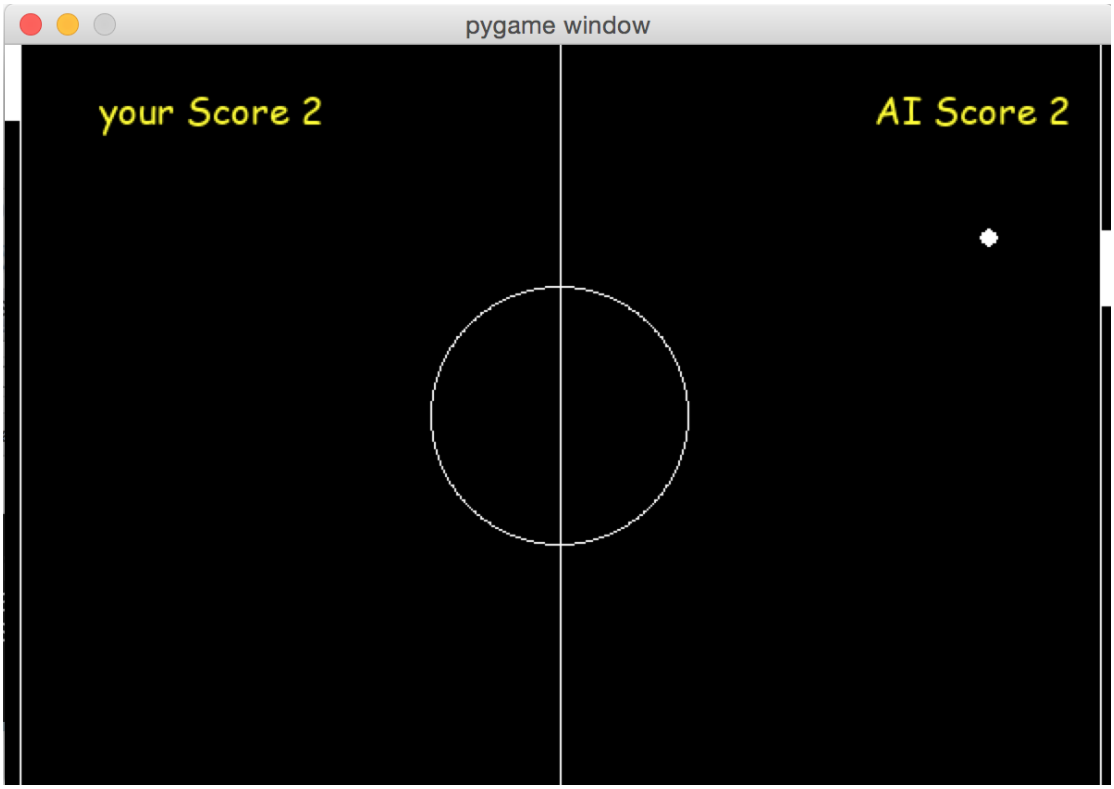
Neuro-Evolutional Augmenting Topologies attempts to bring the best qualities of both techniques together by unifying them into a single structure. Unlike neural network, where it was inspire by the biological structure of neurons, but eventually deviates from this origin and was developed mostly through principles of calculus. NEAT draws further inspiration from the biological principles of gene cross over, mutation, and natural selection. By essentially treating a group of neural networks as the brains of organisms and simulate the process of evolution on them to encourage growth towards a certain objective. Their effectiveness in achieving this objective are evaluation as fitness function, which will determine whether an individual lives and reproduces.

This report aims to explore that effectiveness through an experiment in which traditional AIs will compete with evolved NNs as plays in a computer game.

Aim, Approach, and Method

The goal was to have the algorithm evolve an individual that can at least compete with the AI to a standstill.

The game selected was Pong, where two paddles each controlled by a player. A player receives a point each time the ball enters opponent's goal.



This game was chosen for its simplicity. Each instance of game lasts less than 20 seconds, and the fitness function was easy to compute. In this experiment, the left paddle will be controlled by a neural network while the right one by a traditional AI. The fitness evaluation will simply be how much the NN scored more than the AI when the game ends. A game will end once any side scores 5 points.

The algorithm was written in three python modules. A pong.py file that has all the requirements to run the game, a da.py file that contains the standard neural network structure, but with GA functions such as crossover and mutate that can operate on any given NN, as well as a build function that can turn genomes into neural networks. These functions are implementations from 'Evolving Neural Networks through Augmenting Topologies' (Stanley, Miikulainen, 2002)

The experiment will have the sss3.py run the game pong.py, while using da.py to evolve a suitable NN, in an attempt to beat the traditional AI.

For each neural network, it will have 6 inputs: its vertical position, opponent's

vertical, position, position of the ball, and velocity of ball. The output will be a tuple of numbers, each representing a decision of going up, going down, or staying put. The option of only having 1 output representing where the paddle needs to go was considered, but ultimately discarded since the control scheme for the game was in arrow key.

The mutation rate will gradually decrease as the generation grows, making it less likely for a NN to grow any further in terms of size. This is done to stabilize the fitness of later generations, preventing already fit individuals from changing needlessly, and to avoid oversized networks.

The cross over is done by elimination, when a child is made, it will replace the less fit parent, where the other parent has a chance of surviving into the next generation.

Due to the nature of genetic algorithm no training dataset was needed.

The standard number of generations was set to 24.

Results and Discussion –

The initial test left much to be desired.

```
generation  24  individuals won:  1  out of  120
generation  24  best individual with fitness:  1
```

It was realized that some mistakes were made in the coding, and their resolution drastically improved the result. For example, the 'stay put' decision was not linked to the paddle's movement. This means once the paddle begins to move it will not stop. The crossover algorithm was also changed so that the fit parent is paired randomly with either the second most fir or the least fit. This prevents cliquing, a problem that prevents not yet fit individuals that have the potential to grow from surviving.

Another key improvement made was normalisation of the inputs. Restraining them to a number between 0 to 1. Doing this allowed greater control over the mutation operator as no input was prioritised over the other.

```
generation  20  individuals won:  17  out of  60
generation  20  best individual with fitness:  3
code ------------------------------------
```

The above accounts for most of the inefficiencies. However the some problem still remained.

The resulting individuals can be placed into two groups according to their behaviour: responsive and unresponsive. Unresponsive individuals wait in a specific place and hope the ball will collide with the paddle, while responsive ones will move round, actively attempting to hit the ball. Unresponsive ones

dominate the population at early generations, while responsive ones begin to out number them in later generations, which was to be expected since intuitively, the fittest individual should be a responsive one as the optimum strategy to win this game is to follow the ball. However, as generation progresses, the responsive population began to shrink until they were driven to extinction. Intuitively, the fittest individual should be a responsive one since the optimum strategy to win this game is to follow the ball, thus this should not have happened.

Further investigation revealed that it was an exploitation of the game mechanic. When a goal is scored, the ball will regenerate at the entre of the stage and move towards the player who scored the goal in a random direction, which sometimes can be too fast for the AI to respond to. The NN exploited this by letting the AI score first, which often leads to the AI missing the next round. This luck based strategy was easier to evolve since its only standing still, and has an advantage over the responsive ones, which takes time to become effective and as a consequence, resulting them being driven to extinction.

As luck based strategies cannot reflect the effectiveness of the algorithm, the rule of the game was changed so that points are gained through contact with the ball, and the winning point was raised to 7 from 5 to decrease the chance of lucky wins.

After the problem was rectified the survival rate of the responsive ones increased, and they became the fittest individual for most of the iterations as predicted.

In the final attempt of the experiment, the fittest NN evolved was:

```
genome([node(0), node(0), node(0), node(0), node(0), node(0),
node(2), node(2), node(2), node(1), node(1)], [connection(2, 7,
-0.045410890906228574,      False),      connection(5,      8,
4.926661849333149,      True),      connection(2,      6,      -
0.09729588519510113,       True),       connection(4,       7,
0.6258867536731088,     False),     connection(0,     6,     -
3.0614284653224164,       False),       connection(2,       9,
1.1286724497269474,     False),     connection(5,     10,     -
0.6115037228770681, True)])
```

To see how this particular individual functions, replace startinGenome() in the sss3.py file with it and run the program.

Conclusion and Future Work –

The Pong game was originally chosen for its simplicity for coding alone. However

it appears that simple games often have the benefit of having an optimum strategy, which can prove challenging for the GA.

Though some improvements can still be made:

The generation count was set to 24 due to time constraint. A more reasonable number would be 120, where the fittest individuals have time to refine its behaviours for better performance.

The speciation technique was originally implemented to counter the mechanics exploitation problem, but not used due to time constraint. Utilising it's function will likely increase the emergence of fit individual since it gives potential fit individuals room to grow. Escpecially in this case, where responsive individuals will take a long time to surpass the unresponsive ones, preventing them from getting extinct. This may have been the better solution instead of changing the rules of the game.

In the end, regardless of whether or not the individual was able to defeat the AI, the neural networks was still able to adapt to the environment, showing that the algorithm the algorithm is effective at creating artificial decision making mechanics that can improve overtime without supervision. This can serve as proof of concept so that the same algorithm can be used for other more complex mechanics, not limited to just video games, but also any interactions with an established internal logic that requires decision-making. And due to its bio-inspired nature, the decisions made by an evolved NN can in turn inspire people to realise how their own decision making method can be improved.

References –

http://nn.cs.utexas.edu/downloads/papers/stanley.ec02.pdf
'Evolving Neural Networks through Augmenting Topologies'

https://www.youtube.com/watch?v=qv6UVOQ0F44&t=121s
MarI/O - Machine Learning for Video Games

https://www.youtube.com/watch?v=Ipi40cb_RsI
MariFlow - Self-Driving Mario Kart w/Recurrent Neural Network