Project: Genetic Algorithm (Neuro Evolution)

Bouncing Ball Game

Developer Name: Partha Mehta

Overview:

In this project I have implemented AI bouncing ball game using Neural Network whose weights are controlled by Genetic Algorithm. (Neuro Evolution). The Bouncing ball game is contain paddle and the ball and the aim is to avoid ball to hitting the ground. Paddle can move in 2 directions (left or right) and it is controlled by the output of fully connected neural network. Since, we don't have specific training data to train neural network, we use streaming data as input to neural network and its weights is updated using genetic algorithm (using score as fitness). Runing instruction is given at the end.

Implementation:

This project contains 4 py files –

- Game.py – contains bouncing ball game logic.
- Genetic_Algorithm.py – to generate the agents to play bouncing ball game.
- Neural_Network.py – to control the paddle.
- Dynamic_plot.py – used to plot the dynamic graph of fitness vs generations.

Details:

Game.py: This file contains logic to play bouncing ball game. Used as GUI (TkInter library in python). Contains randomly moving ball and the paddle to bounce the ball. Objective here is learn the movement of paddle using NN and GA. Which will be explained further below.

Genetic_Algorithm.py:

Steps:

- Generate a random population of weights for neural network. Neural network architecture used is 2 input node, 1 hidden layer with 2 nodes and one output node. So **chromosome length is 6 (Bias is handled separately)** with random floating numbers. **Population size is 10.**
- **Genotype and phenotype mapping** here is random floating numbers as weights which controls output of neural network which in return controls the movement of paddle.
- **Evaluation of fitness:** Each Agents (population) is allowed to play a game and its score (no of times successful hits) is recorded.

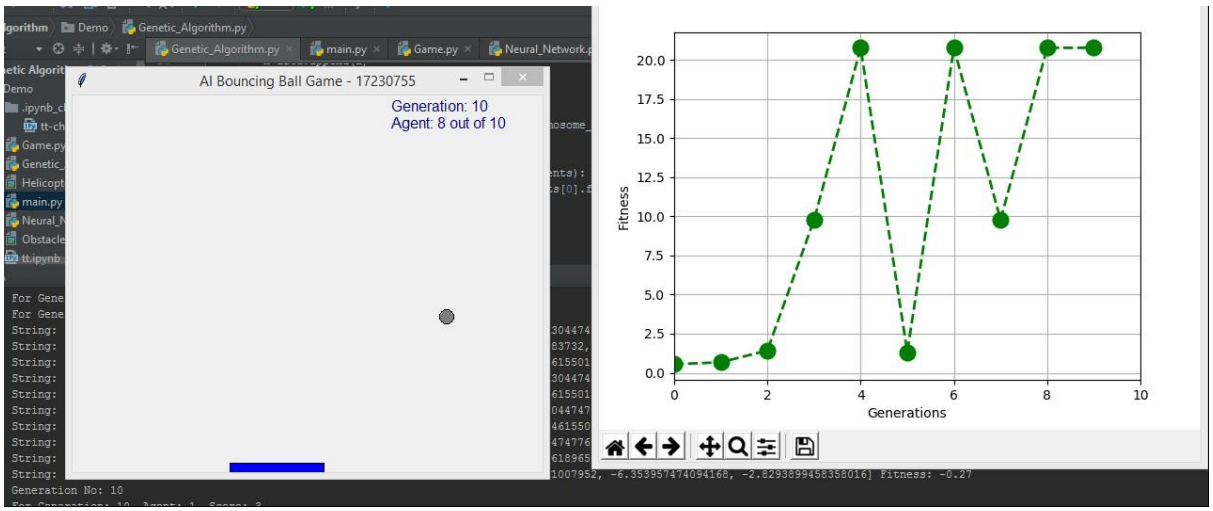
  Fitness function = Score – abs(x_position of ball - x_position of paddle)/(total window length i.e. 600)

  -abs(x_position of ball - x_position of paddle)/(total window length i.e. 600) is used to determine how far is the ball from paddle just before ball hits the ground. This ensures the paddle come near ball at each generation i.e. near miss is better.

- **Selection:** Top 20% Agents (based on fitness) are selected.

- **Cross Over –** 2 Random choice of selected agents (20%) for crossover and mutation. Child1 and child2 from parents are created hoping better solution found and increase the diversity.
- **Mutation:** Mutation rate used is 0.3. Randomly select the single index of the chromosome and previous weight + randomly generated number between (-2, 2).
- This process is repeated for 100 generations (excepts the first step).

Observation: Can obtain score of 25 in around 5-6 generations. Can obtain more if we train more. However, its is likely to get stuck at local maxima. (stuck at score 25 can see in below graph)



Neural Network.py: Structure of neural network implemented here is 2 input node, single hidden layer with 2 nodes and one output node.

Input to NN contains 2 features –

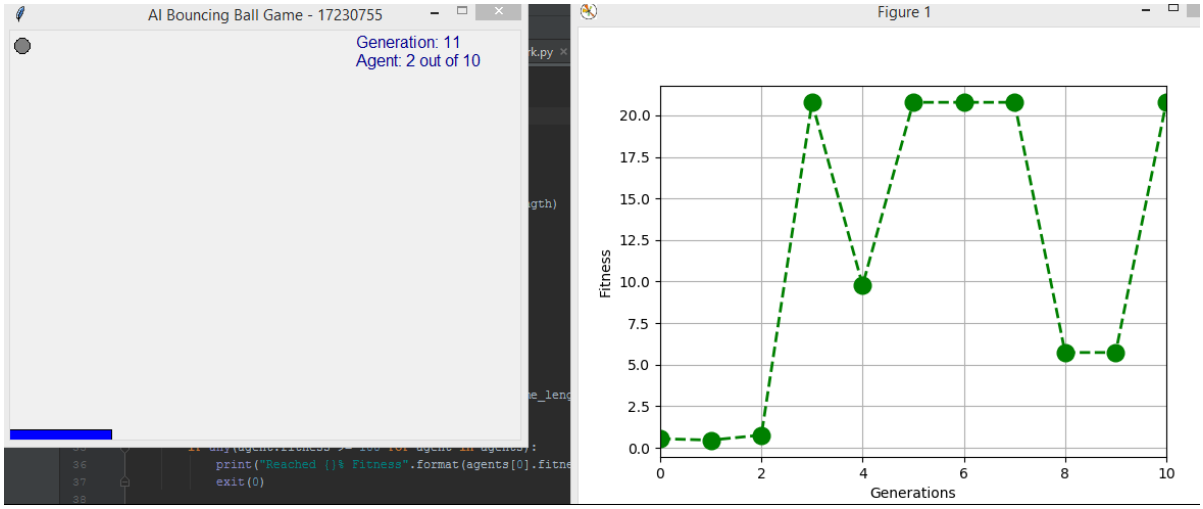
- Feature 1: x_position_ball – x_position_paddle
- Feature 2: y_position_ball – y_position_paddle

Output node is given to the paddle controller. If output is <= 0.5 turn left else turn right. The weights of the NN is represented as chromosome from genetic algorithm.

Note: Random initialization of weights is done + or – around base or reference value. This is because to save simulation time. This value is found using experimentation.

Dynamic_plot.py: This used to get dynamic plot of fitness vs no of generations.

Simulation Results and Discussion:




```
Generation No: 10
For Generation: 10, Agent: 1, Score: 8
For Generation: 10, Agent: 2, Score: 4
For Generation: 10, Agent: 3, Score: 2
For Generation: 10, Agent: 4, Score: 1
For Generation: 10, Agent: 5, Score: 0
For Generation: 10, Agent: 6, Score: 6
For Generation: 10, Agent: 7, Score: 21
For Generation: 10, Agent: 8, Score: 3
For Generation: 10, Agent: 9, Score: 2
For Generation: 10, Agent: 10, Score: 21
String: [5.107529200457767, 5.6396775921088, -0.2884975951471609, 0.12704336053764664, -6.076249414074317, -4.421758935381646] Fitness: 20.781666666666666
String: [4.248606354722618, 3.563134735386237, -0.18230897654757694, 0.12704336053764664, -6.076249414074317, -4.421758935381646] Fitness: 20.781666666666666
String: [4.1955262146339845, 5.710174810927212, 1.3545290750194638, 0.12704336053764664, -5.096304472388526, -4.421758935381646] Fitness: 7.7683333333333335
String: [4.1955262146339845, 5.46050176830163, 1.3545290750194638, 0.12704336053764664, -6.076249414074317, -4.421758935381646] Fitness: 5.733333333333333
String: [4.1955262146339845, 4.359324998714501, 1.3545290750194638, 0.12704336053764664, -6.076249414074317, -3.9692812742187993] Fitness: 3.756666666666667
String: [4.1955262146339845, 3.769623616262105, 1.3545290750194638, 0.12704336053764664, -6.076249414074317, -2.494443309059252] Fitness: 2.775
String: [4.1955262146339845, 2.6433486785257267, 2.1859570329344873, 0.12704336053764664, -6.076249414074317, -4.421758935381646] Fitness: 1.325
String: [2.5283549369671263, 3.769623616262105, 1.3545290750194638, 0.12704336053764664, -6.076249414074317, -3.106764764317067] Fitness: 1.1949999999999998
String: [4.1955262146339845, 3.769623616262105, 1.3545290750194638, 0.12704336053764664, -4.61705640822311, -4.421758935381646] Fitness: 0.7666666666666666
String: [4.1955262146339845, 3.769623616262105, 1.3545290750194638, 0.12704336053764664, -6.076249414074317, -4.421758935381646] Fitness: -0.27
```

Conclusion:

From the above simulation we can see that for fitness is reached around 21 for 10th generation. Moreover, similar fitness value also reached at generation 5, 6 and 7. Hence, there may be another solution which can give better solution. This needs to be experimented. From above simulations we can see that GA stuck at local maxima. We need to include more diversity in order to get out from the local maxima. More generations is also likely to help to kick off from local maxima.

Running Instruction:

- Folder structure – Genetic_Algorithm -> Demo, python_vitrual_environment
- Demo -> Dynamic_plot.py, Neural_network.py, Genatic_Algorithm.py, Game.py
- Create python virtual environment in Genetic_Algorithm folder
- Required Packages –
    - certifi==2018.4.16
    - curses-util==0.0.6
    - cycler==0.10.0
    - fuzzywuzzy==0.16.0
    - kiwisolver==1.0.1
    - matplotlib==2.2.2
    - numpy==1.14.5
    - Pillow==5.1.0
    - pygame==1.9.3
    - pyparsing==2.2.0
    - python-dateutil==2.7.3
    - pytz==2018.4
    - scipy==1.1.0
    - six==1.11.0
    - wincertstore==0.2
- Run **Genetic_Algorithm.py** file –

Reference:

[1]. DJ Oamen – Bouncing ball game code python.
https://www.youtube.com/watch?v=9tVCYIcwNjw

[2]. Genetic Algorithm – wiki https://en.wikipedia.org/wiki/Genetic_algorithm