GameBot for Flappy bird with Neuro-evolution

Weiqing Li [1] **Yang Tao** [1] **Haowen Zhu** [1] **Dekun Geng** [1]

Abstract

The field of machine learning is always an interesting and significant area of research in artificial intelligence. In this project, a special type of neural network, neuro-evolution, was applied to the popular mobile game Flappy Bird. Neuro-evolution is a combination method of neuro network and genetic algorithm. We intended to develop a game bot trained by neural network where instead of Back Propagation (BP), we used genetic algorithm to optimize the model. The proposed algorithm was tested in real game environment and the highest score achieved 54169.5 in total 1200 generations, respectively. We also investigated the trade-off between run time and accuracy. With appropriate settings, the neuro-evolution algorithm was proven to be successful with relatively short converge time.

1. Introduction

Flappy bird is a popular two-dimensional side-scrolling game in early 2014(4). Users need to touch the screen or click the mouse to make the bird 'flap', which means go up to avoid hitting pipes. Once the bird hit a pipe, the bird will die and the game ends. If users don't touch the screen, the bird will fall to the ground due to gravity, which also makes users lose the game. The bird can get one point when it goes through a pipe. The further the bird flies without hitting any pipe or touch the ground, the higher score users would get. It is an interesting game to test users response capability. The game interface is shown in Figure.1. In our project, we applied neuro-evolution algorithm to train birds to dodge the pipe, eventually, the bird could play this game to get a relatively high score.

Neuro-evolution(3) is a machine learning algorithm that applies evolutionary algorithms to develop artificial neural networks, taking motivation from the evolution of organic nervous systems in nature. Compared to other neural network learning strategies, neuroevolution is exceedingly general; it allows learning without unequivocal targets, with only sparse feedback, and with arbitrary neural models and network structures. Neuroevolution is an effective approach to tackling reinforcement learning issues. When faced a reinforcement issue, it is usually hard to make a bunch of training data. Also, our goal is to get higher score. Thus, BP is not very suitable in this kind of problems. Neuroevolution most commonly connected in evolutionary mechanical technology and artificial life.

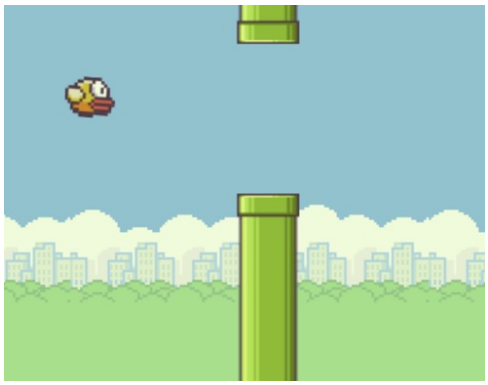

Figure 1. Game interface for Flappy Bird

1.1. Research contributions

In this project, we tried ourselves to develop a gamebot for flappy bird. Instead of the traditional method of reinforcement learning, we tried a relatively new one to see if it could perform well.

2. Related Work

There has been a lot of research on how to use Artificial Intelligence to play Flappy Bird. In 2013, Gatt (2) has already used Q-learning to help users achieve higher score in Flappy Bird. While Chen (1) used OpenCV and deep learning to analyze the image of game screen and get the position of the bird and the pipes, then applied Q-network to the reinforcement issue. Shu, Sun, Yang and Zhu(4) tried different ways including simple Bang Bang Control, SVM and Q-learning. There is also some research on the application of Neuroevolution on vedio game, for example,

[1]Worcester Polytechnic Institute. Correspondence to: Joseph E. Beck <josephbeck@wpi.edu>.

CS534 Artificial Intelligence Course Final Project, Worcester Polytechnic Institute.

Stanley et al.(5) used it to train the NPC in a vedio game.

We will use the project as a trial for neuroevolution as a game bot in a simple game. The source code of Flappy Bird the game is from TimoWilken in Github[1]. We modified some parts to generate a bunch of AIs as a generation and use AI to control the bird instead of people.

3. Proposed Method

3.1. Neural Network

Neural networks are one of the main tools utilized in machine learning. As the neural portion of their title suggests, they are brain-inspired frameworks which are planning to duplicate the way that people learn. Neural networks comprise of input and output layers, as well as (in most cases) a hidden layer comprising of units that transform the input into something that the yield layer can utilize. They are fabulous tools for finding patterns which are distant as well complex or various for a human software engineer to extricate and instruct the machine to recognize.

In our project, we mainly used a feedforward neural network as the framework of the model and applied sigmoid function as activation function. The performance of sigmoid function is shown in Figure. 2[2].

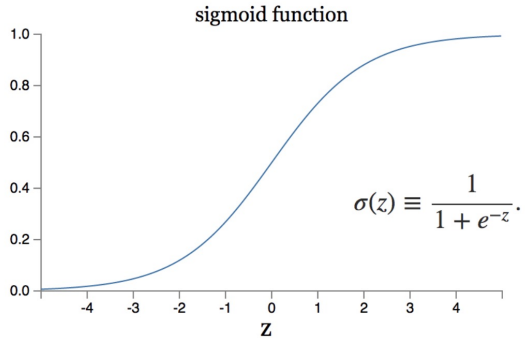

Figure 2. Sigmoid Function

A feedforward neural network [3] is an artificial neural network wherein connections between the units do not form a cycle. As such, it is different from recurrent neural networks. The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

[1] github:https://github.com/TimoWilken/flappy-bird-pygame

[2] https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7

[3] https://en.wikipedia.org/wiki/Feedforward$_n eural_n etwork$

In our project, we use neural network with only one layer of hidden layer. An explanation of the neural network is in Figure.3.

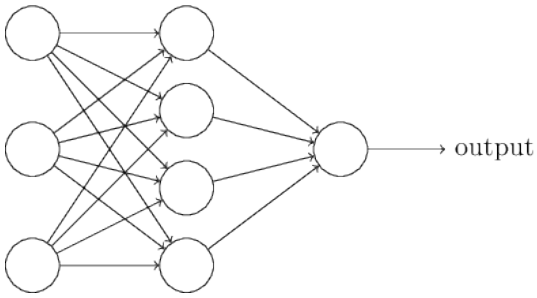

Figure 3. Neural network Model with one hidden layer

3.2. Genetic Algorithm

A genetic algorithm (GAs) is a search heuristic that's motivated by Charles Darwins hypothesis of normal evolution. This algorithm reflects the method of natural selection where the fittest people are chosen for reproduction in arrange to deliver sibling of the next generation.

Following is the foundation of GAs based on the analogy with genetic structure and behaviour of chromosome of the population:

1. Individual in population compete for resources and mate.

2. Those individuals who are successful (fittest) then mate to create more offspring than others.

3. Genes from fittest parent propagate throughout the generation, that is sometimes parents create offspring which is better than either parent.

4. Thus each successive generation is more suited for their environment.

In our project, we used Genetic algorithm instead of the Back Propagation to optimize the parameters of neural network.

3.3. Neuroevolution

Neuroevolution could be seen as a combination of neural network and genetic algorithm. It use the feedforward in neural network to decide the action of AI. As in genetic algorithm, it use the score (or to say total reward) of every AI to generate the next generation.

The details of the application of neuroevolution could be seen in next section.

Algorithm 1 Neuroevolution

Require: Population size P, Neural network size [$input\_number$,$hidden\_number$,$output\_number$]
Ensure: $D = \{Neural\ networks\ in\ last\ generation\}$
1: Initialize: randomly generate P neural network as $Generation$
2: **repeat**
3: **for** nn in $Generation$ **do**
4: put nn in the game
5: get $score$
6: **end for**
7: $Generation$ = **Next Generation**($Generation$)
8: **until** Stop by user

4. Implementation

The main steps in the Filppy Bird is shown in Figure.4. In it, when the bird is alive for one round (a micro second), it will add 0.5 point to the total score. If the bird pass one pipe, there is 1 point as bonus. To set the point as so is because we want to add a reward for the bird's alive to keep it away from flying to the edge and dying young.

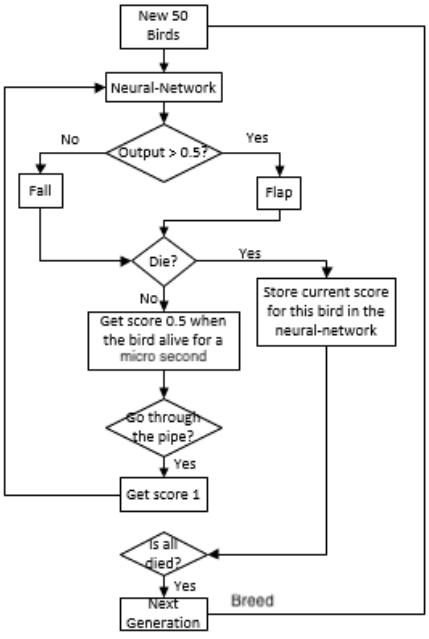

Figure 4. Flow Chart

For the neural network, we set three inputs (bird.y, Pipe.x, Top_height), a hidden layer with nine neurons, and an output. Figure.5 shows how the neural network works. Figure.6 shows the position of input parameters.

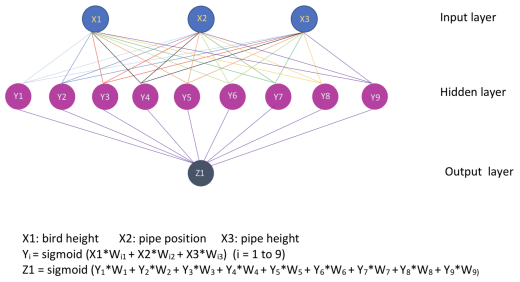

Figure 5. Neural Network

We choose this three parameter as inputs is because these three are the mainly contributors to the action of the bird. The bird's x position is fixed, so that when flap, it only changes its y position. The y position is an important factor for the bird to choose if there is a necessity to flap. As for the pipe, we have to consider its x position, as the pipe will move nearer and nearer to the bird and the position of the hole. However, the hole's length for the bird to go through is a fixed number, so that we could just consider the Top_height.

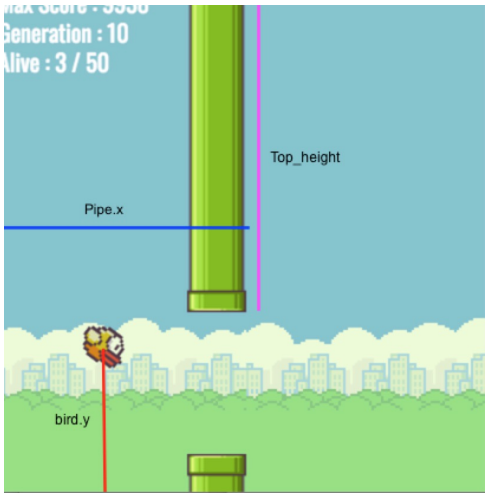

Figure 6. Position of input parameters

When comes to the genetic algorithm, the key point is how to breed the next generation. In our project, we have three part for generate the next generation: elitism, random behavior and breed.

In breed, we choose the parents as rank first with rank second, rank first with rank third, rank second with rank third... In this method, with a population of 50, to solve the function $\frac{(1+n)n}{2} = 50 \cdot 0.7$, we could find out we only use the first 8 neural network. Although it seems too "exploit", in our test using a roulette choosing, after 1000 generations the bird still can't fly over 10 pipes. With this way, after

500 generations, the bird could fly really well.

The method for two parents' breed, we randomly choose one weight, which means only one number, of the child's weights from parents' corresponding weight. After chosen the number, there is a chance for this number to mutate. Instead of just mutating to a random number, it will mutate in a set range so that the weight will not become too weird.

In elitism, we keep 20% of the top neural networks in the last generation, it is because we don't want to loose these good neural networks. However our method to generate next generation is too exploit, so we add a part to randomly generate 10% of the next generation.

For the algorithm we use in our project, we mainly follow Algorithm.1. However we have done some adaption in the genetic algorithm part. The details could bee seen in Algorithm2.

Algorithm 2 Neuroevolution in Flappy bird

Require: Population size P, Neural network size [$input\_number$,$hidden\_number$,$output\_number$], Mutation rate m, Mutation range mr, Elitism rate e, Random behavior rate r
Ensure: $AIs = \{Neural\ networks\ in\ last\ generation\}$
{Genetic}
function Breed($indi1$,$indi2$)
  for weight w in $new.neweights$ **do**
    w = **random.choice**(corresponding w in $indi1$, corresponding w in $indi2$)
    {Mutation}
    if random() $<$ m **then**
      w += **random**()$\cdot mr$
    end if
  end for
  Return new
end function
function Next Generation(AIs)
  initiate $next$ as an empty list
  {Elitism}
  for $indi$ in top e in AIs **do**
    put $indi$ in $next$
  end for
  {Random}
  generate $r \cdot P$ individuals randomly
  put them in $next$
  {Breed}
  repeat
    for AI in AIs **do**
      for AI_better ranked before AI **do**
        $child$ = **Breed**(AI,AI_better)
        put $child$ in $next$
      end for
    end for
  until $next$'s size reach P
  Return $next$
end function
{Game}
function Start
  Build and draw pipes and background
  Build P birds for P AIs(neural network) to control
  $score$ = 0
end function
function Game
  Start
  Initiate P AIs with random weights
  repeat
    for AI in AIs **do**
      Get $input$ as $[bird.y, pipe.x, pipe.height]$
      res = **feedforward**($input$)
      if res ¿ 0.5 **then**
        Bird flap
      end if
      if bird dead **then**
        record $score$ as this AI's score
        if birds all dead **then**
          Start
          AIs = **Next Generation**(AIs)
        end if
      end if
      if Through a pipe **then**
        $score$ += 1
      end if$score$ += 0.5
    end for
  until Stop by user
  Return AIs
end function

5. Results

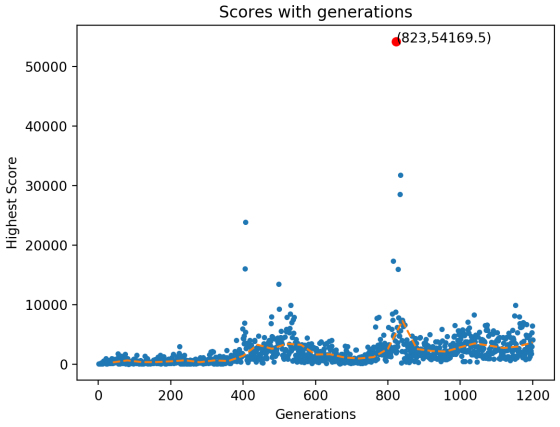

Figure 7. Scatter plot for highest score with generation

The scatter plot for number of generations versus highest score is shown in Figure.7. The orange line is the average score for ever 40 generations. The max score is in red. For a better view, we plot the figure with log-scaled score in Figure.8.We can infer that with the increase of the number of generations, the flappy bird would survive for a longer time, which indicates that our neuro-evolution algorithm will effectively to help machine learn how to get higher score with more training time.

If we log the score. We could clearly see from Figure.8, after 400 generations, the bird seems began to know how to fly. And more generations doesn't mean the bird will fly better. Actually after 500 generation, the bird seems already know how to fly. However, the more time you trained, the max score in history will definitely go up. In our result, the highest score is 54169.5, reached in 823 generations, which means the bird went through nearly 652 pipes. Another interesting is sometimes the AIs will have an extremely high score like 54169.5, which seems like some kind of luck. They may not remain this excellent behavior in next generation.

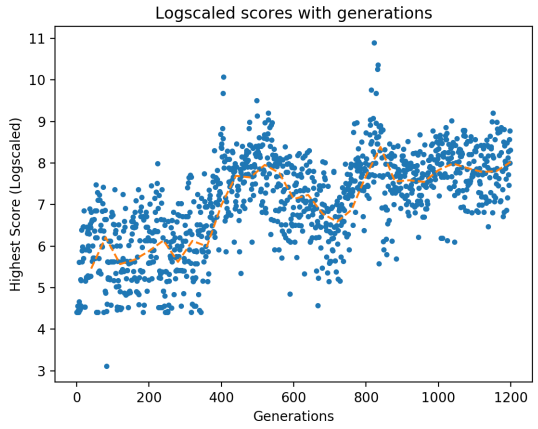

Figure 8. Scatter plot for highest score in log with generation

We also record a demo with the Flappy Bird AI in 285 generation. Demo Video for Neuro-Evolution Flappy bird

If interested in our code, could check Github address. When download and run the code, it could continue with the 1000th generation which has already trained before.

6. Discussion

Although the bird seems to fly well after 500 generations. But it takes a long time to train it to 500 generations. We spent nearly 6 hours for the 500 generation and spent 27 hours for 1000 generation. When the bird flies well, it takes a long time to end one generation.

In this project, we initiate the weight of the neural network with random number. There is a possibility that with a better way to initiate the weight, the algorithm could converge faster.

Compared to Shu et al.(4)'s work, the outcome of neuroevolution and Q-learning is similar. When reached 400 generation, the ai could play nearly 4000 points in our algorithm, which is about 4 in log-scaled score in their work.

Another problem is we played these AIs inside the game, which means we could get the input directly from the game. If these data are not available, we may think about using OpenCV to analyze the shot of game interface and get these data.

7. Conclusions and Future Work

Briefly summarize the key findings your paper and point out interesting directions for future inquiry.

In this program, we set pipes static when the bird goes through, in order to increase the difficulty, we can set pipes move at a constant velocity when the bird moves.

Besides, we only use one generation to hybrid new birds, in the future, we could use two generations, since for each generation, birds live in different surroundings (pipes are randomly set), birds who performs excellent last generation may be excluded in this generation because of a bad surrounding. To conquer this blind point, we could firstly choose 30% elitism from last two generations, exclude same birds and use the remaining to hybrid based on former rules with a descending order of their scores.

References

[1] Kevin Chen. Deep reinforcement learning for flappy bird.

[2] Brian Gatt. Reinforcement learning-flappy bird. *Goldsmiths University of London*, 2014, 2013.

[3] J. Lehman and R. Miikkulainen. Neuroevolution. *Scholarpedia*, 8(6):30977, 2013. revision #133684.

[4] Yi Shu, Ludong Sun, Miao Yan, and Zhijie Zhu. Obstacles avoidance with machine learning control methods in flappy birds setting. *Univ. of Stanford, CS229 Machine Learning Final Projects Stanford University*, 2014.

[5] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE transactions on evolutionary computation*, 9(6):653–668, 2005.