How to Teach a Computer to Play Flappy Bird

Riley E. Stadel

May 15th 2019

[**Project Overview**](#_oecfqm1ktu82) **3**

[**Important Game Elements**](#_1raxpjpfhpnt) **4**

[**Processing the Data**](#_6xfcciansyvq) **4**

[**Teaching the AI**](#_87jv6idfv45s) **5**

[**Learn the Game**](#_se62zagofryz) **5**

[**The Multi\_Layer Perceptron**](#_gf1h0jbys78b) **6**

[**Interpreting the Results**](#_bv55rs18v89o) **7**

[**Real World Applications**](#_efd790m0owwt) **8**

[**Watch The Full Process**](#_n0fo4mu0mdc) **8**

# Project Overview

Flappy Bird is a mobile arcade game create by programmer Dong Nguyen. The game is

scroller where the player has to move a bird through between pipe columns without hitting them. Upon release in May 2013 the game took over the app store due to the how difficult the simple concept was to perfect. For this project I recreated the game using pygame and tried implementing a Machine Learning AI to figure out the optimal strategy to maximize the score. Since there is no end the goal is to have the AI be able to continue infinitely without stopping or dying.



# Important Game Elements

In order to build our AI we need to define what data the program should be evaluating when making a move. We will then store that data in a csv file called “DataInputs”. We will then store the resulting outcome of that move in a csv file called “DataResults”. The game consists of 4 important game elements linked to a result. The PipePositionX element describes where the Pipe is located on the horizontal axis. When the pipe reaches X position 200 the bird has made it through the pipe and we start evaluating for the next pipe. Pipe\_Y describes where the offset between the player and the top pipe. If Pipe\_Y is positive, then the player is below the top pipe, whereas the player is above the top pipe if the value is negative. PlayerMom describes the momentum the player has on the vertical axis. Since there is a gravity element the player will accelerate downwards until the jump input is pressed in which case the momentum will reset to -13. For every iteration that jump is not pressed momentum will increase by 2. The final game element is Input which describes if the player jumped or did not jump on that iteration. The value 1 links to a jump and the value 0 links to no jump. In the “DataResults” file there is 1 element which describes the result of the corresponding input. A 100 links to the bird survived the input and a -100 links to the bird dying due to the corresponding input. The reason behind these values rather than a 1 or 0 is to create a bigger punishment between death and survival.

# Processing the Data

The final step before building the AI is to format the data so the program knows what we want it to return. We will define two elements X and y which will be used to train the Machine to learn what y occurred with past X values. The X element is a 4 dimensional array with values PipePositionX, “Pipe\_Y”, PlayerMom, and “Outcome”. The y element holds the “Input” element because that is the value we want to predict. The AI process will look over the data and use values of X to determine the best y for those values.

# Teaching the AI

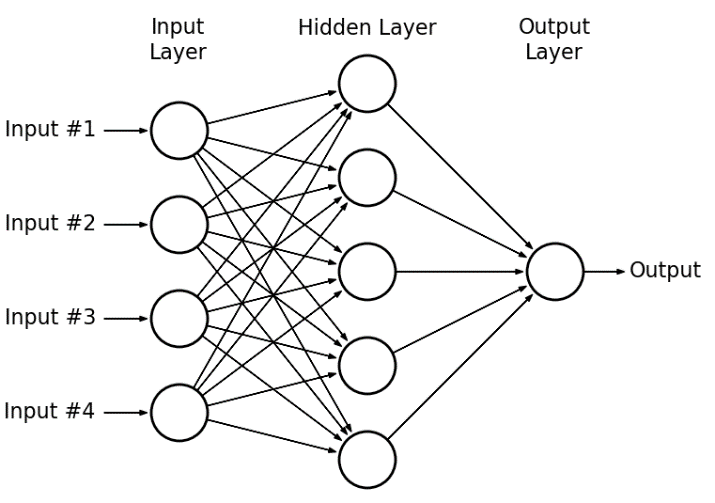
The AI process will take all inputs and group them based on the their corresponding y values. The problem with this is the initial process may not be the best solution. For this reason we used what is called training data and testing data. This process involves splitting up the current data into 90 percent training data, and 10 percent testing data. We will used the training data to classify X values to y values to our chosen Machine Learning Algorithm. The test data is used to test predictions. We will use the X values in our test data to predict a y value, and compare that prediction with actual y value. We can calculate how far off how program was using a function called Mean\_squared\_error. Using this formula we want to loop through this test/training data method for a chosen number of times, usually until it does not improve for a number of iterations, and use the loop with the smallest mean\_squared\_error value.

# Learn the Game

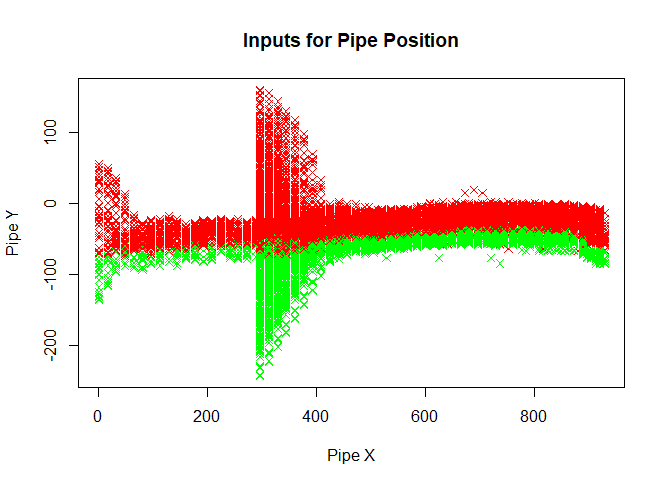
Now that our Algorithm is predicting at the highest level it can with our data, we want to can use it to play the game. The prediction process involves creating a new X value that contains all the current game elements, with the assumption that the bird survived the input. We will use that X to predict a y value which is our input of jump or no jump. If the returned prediction is a 1 we will tell our Bird to jump, otherwise we will tell our bird to not jump. The final step is to record what happened with that prediction so the AI can use that data to not make the same mistake the next time it plays.

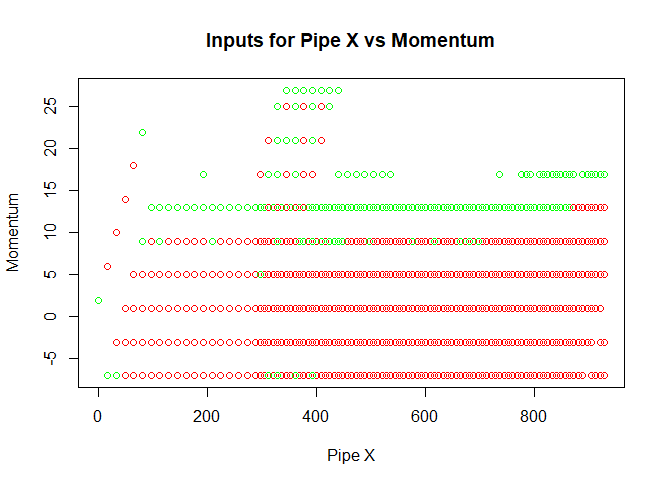
# The Multi\_Layer Perceptron

The multi\_layer perceptron(MLP) is an Artificial Neural Network that uses a technique called backpropagation. Using at least three layers; input layer, hidden layer, and output layer; the MLP differentiates itself from linear perceptron and can evaluate data that is not linearly separable. The algorithm teaches the machine how to predict data by assigning weights to each value. Those weights will be a constant value that will help predict either a 1 or a 0, which correlates to a jump or no jump. The MLP reads the data provided in the input layer along with the expected output value of 1 or 0. Next the MLP calculates values to multiply each data input by such that the weighted values add up to the expected output value. The result will produce an algorithm that can take in a set of never before seen input values and learn the what the best output is.

****

# Interpreting the Results





The graphs below show the inputs for 29895 successful inputs based in order to illustrate how the AI differentiated the data. A green marker represents the AI returning a jump, and the red marker represents the AI returning no jump. The first graph shows the Pipes X position based on the Pipes Y offset with the player. The AI learned that at around Pipe Y of -75 and below it should return a jump, and above that value it should return no jump. The second graph illustrates the relationship with the players momentum. A negative momentum means the player is moving up, while a higher position momentum illustrate the speed it is falling. The AI was able to learn that it should input a jump at around 11 momentum. While watching this play out, this pattern of jumping at 11 momentum every time creates a path where the player stays near a constant Y position.

# Real World Applications

While perfecting the game flappy bird has no greater impact in real world applications, the project can be used to show how a humans brain processes data to learn real world challenges. Every action that we do, we need to process incoming data to return a ‘do’ or ‘do not do’ result. Just like the AI learning that jumping at a Pipe Y offset of -75 results in a positive outcome, we evaluate every day processes the same way. This project also shows why subjects that our brains have already learned take much faster than subject that we have never seen before. Since our brains have already calculated the input weights for mastered subjects we do not need to spend time trying to process how to return an optimal result before performing the act.

# Watch The Full Process

The full process will be posted on the featured video section of [www.RySu.Space](http://www.rysu.space) within the next couple of days. Check it out the see how the AI learned from the beginning.