Comp704 Development Journal

1507680 Dean Harland

April 2020

1 Introduction

During my 704 Application of Machine Learning module I will be recording a journal of my experiences, issues and decision making while researching and experimenting with machine learning for game AI. The layout of this journal will firstly discuss the general aspects of my game, collection, data and training for my machine learning, which then moves onto the algorithms and the problems and solutions made for each of them.

2 Game

Initially I was leaning towards implementing a 2D pinball game for this module however this could have been too complex and would make it difficult to create an AI for the game. In this case I decided to go for a 2D endless driving game. The reasoning behind this is my lack of experience in python, as the game is only a small part of the module but also plays a big part in. Acquiring the basic knowledge to produce the game is my first problem. By choosing the type of game I did, I was able to research a few tutorials which were based on making small games in Pygame. As the games these tutorials were making were not too dissimilar to mine, I was able to experiment with them while learning to delve further into the language, using documentations to also learn what certain things do. Before I started I was planning on making the car move itself and have a roaming background to follow it. However now I found that it would be easier to have the car on a set axis near the bottom of the screen and have the other assets/mechanics move towards the player to simulate the feeling of driving forwards to reduce the complexity of the game.

2.1 Game Reflection

Reflecting back on the game aspect of the module, I was happy with the progress I made throughout from knowing barely anything about Python to creating a simple 2d driving game. However, now that it nears the end of the module I feel like I would have been able to produce a better artifact if I had more confidence in myself and set a plan for development. Although I learned a lot by the ways

listed in the Game section, creating a unique piece of work would have been more fulfilling and pushed me more out of my comfort zone. The plan would have consisted of learning the basics of the languages as before, then conducting an iterative approach to developing the game. This iterative approach would then be integrated into the collection stage as to assure that certain problems listed in future work would be somewhat avoided.

3 Collection

Most models will need thousands or even hundreds of thousands of lines of data in order to learn from it and produce a good one, this would mean that someone would have to sit and play the game for 15+hours.

Creating a symbolic AI to play the game for me would relieve the need for a human player. Not only that it can also be left on unattended and play continuously without stopping.

One problem was how to deal with the player moving out of the way of the obstacle while also going for the coins. When the player moves to get a coin it would sometimes just crash straight into the obstacle. To fix this I created an if statement that when the obstacle was below the halfway point (400y) and is near the players X then move a set distance to the side and wait for it to pass before going for the coin. This also led to the game possibly lasting forever, with the player almost never dying and playing perfectly. This would lead to the data being too good and not having much variation. To fix this I created a new variable called Error size, every time an obstacle went off the bottom of the screen there would be a random int check from 1-100, if it is higher than or equal to 75 then decrease the error size. This meant that after a while the player would get closer and closer to the obstacle and then be so close that it loses.

```
# some form of ai thing
if (obstacleY > 400) and (obstacleX <= playerX <= obstacleX + errorSize):
    playerXSpeed = rightSpeed
elif coinX <= playerX:
    playerXSpeed = leftSpeed
if (obstacleY > 400) and obstacleX >= playerX >= obstacleX - errorSize:
    playerXSpeed = leftSpeed
elif coinX >= playerX:
    playerXSpeed = rightSpeed
```

```
x = randint(1, 100)
if x >=75:
    errorSize = errorSize - errorRate
```

After playing the game for a while I realised that the game was rather slow, this meant that I had to wait longer for results and also that less results would be taken in the same amount of time. So a simple fix of doubling all movement variables in the game increased the collection process efficiency considerably.

3.0.1 Collection Reflection

I discuss the reflection of my symbolic AI in the future work section. However another reflection point would be the use of a Http server for collection. This was discussed a few times during lectures and workshops, although I believed I

did not have the skills to make it work and chose the safer option. But by using a server for collection I would have been able to make use of multiple computers to maximise the data collection. This also lifts the restrictions of my personal computer which ran slow while collecting and now takes 20-30 minutes to boot up.

4 Data

The data I was initially collecting was Time, PlayerPos, ObtstacleXPos and CoinXPos. This was collected every time the player collects a coin, it was implemented this way as if the player collects a coin then we know that it is alive and not inside an obstacle. However I soon realised the amount of gaps in data due to only collecting these things when the player collects a coin. To fix this I added in more collection methods such as when the obstacle goes off the bottom of the screen and when a coin also goes off the bottom of the screen.

I wanted to save the data in a new worksheet every time the player died from colliding with an obstacle. So I created a naming convention for each new sheet to be Run(playthrough); play through being the number of restarts. However when it came to using and loading the excel sheet I was unable to do so because of an error from pandas. This error was due to it being saved as an XLXS instead of a CSV, so I would have to redo the data so that it is in a single sheet format.

A very simple problem that occured which had me going back and fourth was that I pickled my model using one method and then I unpickled it using a different one. This caused a lot of confusing because I was sure that nothing was wrong. By taking a step back from the computer for a day after researching the problem, I was easily able to identify the problem the next day. [1] [2]

4.1 Data Reflection

Overall I was somewhat happy with my data. I had enough data to produce a semi functioning model and although I improved on it throughout there were gaps in the data. The things I would do differently would be to gather more data/data types than I originally thought. I gathered enough for it to work but I could have just collected most of the variables I had in my game and then trimmed the ones that were unnecessary during training. This would have allowed me to test different input variables for my regression which could have led to better results.

5 Training

With the continuous state of my player and game, using a regression model would fit best. As my player is locked on the Y axis it will be best to use regression to find out the linear relationship between the input(Obstacle and

Coin) and the output(player). It should then predict where the player should be along the X axis.

6 Linear Regression and multi regression

With the basic nature of linear and multi regression I have put them in the same section. This is because they both acted in similar ways.

Initially starting with 4000 lines of data for my linear regression model the player would move towards the coins with ease however as expected had no awareness of the obstacle. I did not expect to get much more from linear regression so moving onto multi regressions I collected 8000 lines of data in total. The movement of the player was similar to that of the linear regression, however it would sometimes move slightly in the direction of the obstacle and miss the coin. My first thought was the lack of data in my model, so after collecting 26000 lines of data I tried it again. Although the movement of the player was smoother it still had the same problem of missing coins because it would move slightly in the direction and the main problem of not avoiding the obstacles in the slightest.

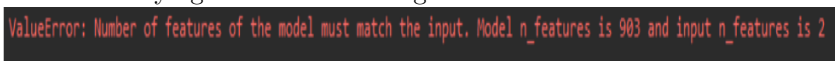
Maybe the lack of data for the Y positions of coin and obstacle was causing problems as it was still only working out its position compared to only the X. by simply adding in the Y for obstacle and coins into the database append would fix this. However this would mean re recording all the data, I would take this moment before recording to add some more data collection. This involved adding 2 checks in for when the obstacle moved down past a certain part of the screen so that it can save the locations when it is halfway down.

After returning with the new data of around 16000 lines and implementing it into my multi regression model, the same problem occurred. I did not expect much from this new data as further research showed just how basic linear and multi regression is. [3] [4]

7 Gradient Boosting Regression

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. (Wikipedia definition) So, the intuition behind gradient boosting algorithms is to repetitively leverage the patterns in residuals and strengthen a model with weak predictions and make it better. [5][6] [7]

With the gradient boosting regression algorithm I used the past data to train the model and test how well it performed as a base line. The model trained fine however when trying to use it I would get an error.



The problem was in where I was setting the X variables as dummies, which is giving categorical variables instead of continuous variables. So every obstacle

X would have 0.1 is a variable. Which in this case is creating of 903 features which expects inputs for each

After I was able to use this model in my testing game, I began experimenting with the algorithm to improve the AI of the player. Although n_estimators helped with the smoothness of the prediction it again led me to believe I needed more data for a more accurate prediction. Leaving the collection of data overnight I was able to collect around 60000 lines of data, with this and my algorithm the smoothness of the player was down to moving left and right in the single digits.

The model's AI was very good at collecting coins, with above a 80 % success rate. However obstacles played a big part in how poorly it performed. Compared to linear regression the AI is attempting to dodge the obstacles just not very well. Before going any further with the algorithm I want to see if adding a variable in for when the obstacle is below a certain point. This is to assist the model in determining when the obstacleY is close and it needs to be careful during this time. While implementing this I realised I had some problems in my collection, such as the append for when the obstacle moves down the screen past a certain point. I had them reversed so it was doing it twice in the half of the screen, changing it around to record at the bottom half the screen, I also added in 2 more append 100Y apart for more data. Another problem was some of the appends were called in the wrong order, such as if my coins were collected, the coin would respawn and then append getting the new coordinates. After implementing the new collection method, I recorded 14000 lines of data which did have slight improvements and the player would sometimes move around obstacles, showing that I am moving in the right direction. I then increased the data to 23000, which makes the player try to avoid obstacles albeit not very well.

Changing the loss function had a notable difference in how the AI played.

Lad

Lad was semi accurate when collecting coins but it still made silly mistakes when obstacles were involved. Overall this was the worst of the lot even after adjusting the algorithm.

Huber

Huber performed better than lad in most cases, but it still did not have the accuracy of least squares. Certain aspects of huber were good in the sense that it knew where it wanted to be but wasn't efficient enough at it.

Least squares

Least squares performs a little better with collecting coins and is more accurate than huber, it also has an increased overall performance. There was a few instances of it dodging obstacles like a human would, where it would smoothly swerve around and come back on itself.

7.1 Gradient boosting regression reflection

This algorithm was the bulk of my learning and final piece. Personally I was very pleased with the result even though the model wasn't great. Looking back, the only thing I would do differently is to spend more time researching gradient boosting regression and compiling a better note book beforehand instead of trying to jump straight into it. This is because I was constantly stopping and doing research so that I could understand what was happening for each of the iterations. Where as if I had done more initially it would have been at hand.

8 Voting Regression

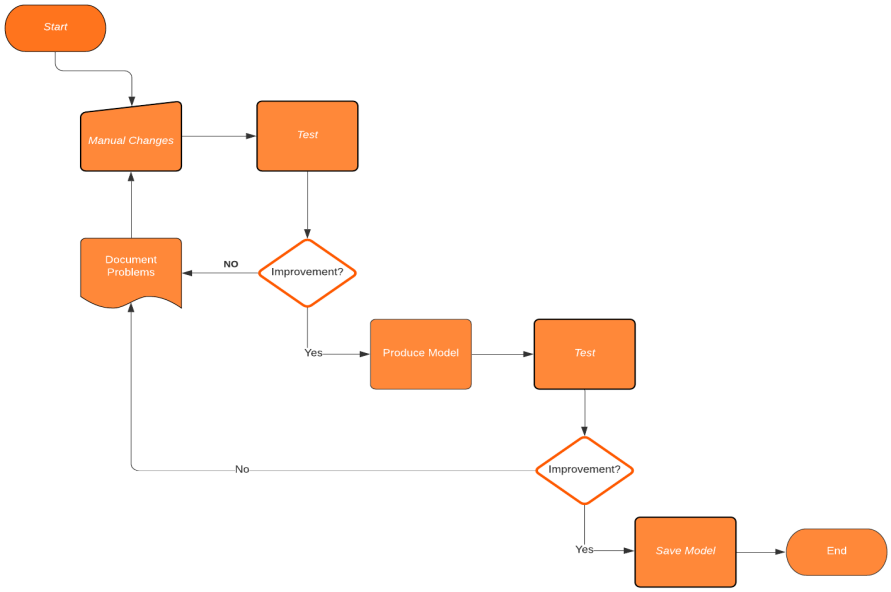
Voting regression was easier to implement and experiment with after using the other algorithms so there were very few noteworthy problems that occurred with the algorithm itself.

8.1 Voting Regression Reflection

The main point I'd like to reflect upon is the use of 2 gradient boosting regression models in the voting regression algorithm. Although this was done because least squares and huber produced somewhat different results and made the model perform better. By using one of the many different regression models like the Gaussian process regression, it could have produced a varied and maybe better model. In the future I would have liked to have 4 different algorithms the change between just to see how well it could have done.

9 UML

Below is my UML diagram for the iterative approach I will be making throughout this module during the training and testing of my data. It works so that after each improvement iteration, I will be able to review the work and document the problems that occur while also moving towards the finalised model. This type of method can be used for each algorithm to simplify the work flow.

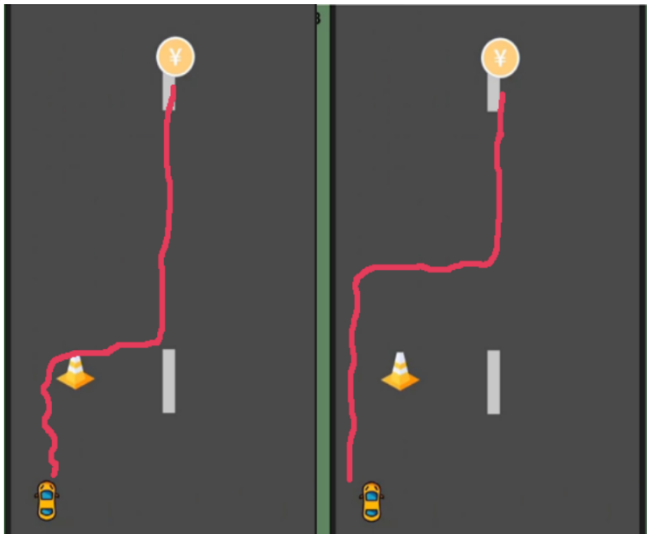


10 Thoughts for future

Does my symbolic AI make the positioning too close to the obstacle because of the symbolic error rate I set up for recording of data?

A problem that became clearer near to the end of development was how close the AI player went to the obstacle. Because of the .25 error of the model, if the AI were to try to narrowly dodge the obstacle it would a lot of the time crash into it. This is because of 2 main points I can potentially see causing the problem. However it would mean that I would need to re design the symbolic AI and then re record all my data again which isn't feasible given time constraints. The first problem which could be fixed with minimizing the models error range. During gameplay the AI player moves to a predicted X coordinate, however because of the error range, the player moves left and right on the spot which inevitably leads to it moving towards the obstacle as it passes it. This leads to the second point of redoing the symbolic AI portion. The way it is currently set up means the AI will not cross below an obstacle if said obstacle is below a certain part of the screen. It will stay a few centimeters off its X while it passes and then move towards the coin. The distance it stays away from then has a chance to be reduced, leading to a smaller and smaller range. Data wise it will cause the AI to learn to stay as close as possible to the obstacle before passing, but because of the above problem this can mean it will crash. A new version which consists of a larger symbolic error range will teach the AI to have more distance from the obstacle. This will help with the models error range and any unfortunate predictions that will lead to the AI crashing. By also increasing the time after the obstacle passes before the AI player can start to move again will also mean

that it knows to keep a safe distance till it completely passes. Another change I would like to make is how the symbolic error range decreases as to avoid data that is too close to the obstacle. Instead I would have 3 different ranges to start with, which are further away than original. Then when an obstacle hits the bottom of the screen, the AI will roll a d100 and if it's above 90, it would start to dodge to the next range down. When the third range is rolled higher than 90 it will set the range to the obstacles as to crash on purpose to restart the collection process. I believe that this would have been a big improvement in the overall performance of my models. To help visualise why this would be important see below.



For my references I did not want to go and find academic references just for the marking rubric, as I did not use them for my solutions. The ones listed below are a few of the ones I used back and fourth until I was able to start producing my models. However I understand the importance of academic research and although we had paper club, finding papers specifically about my research would be a great tool to help me learn and improve upon my work.

References

[1] J. Brownlee, "Save and load machine learning models in python with scikit-learn," https://machinelearningmastery.com/save-load-machine-learning-models-python-scikit-learn/, June 2016.

[2] T. Vanderheyden, "Pickle in python: Object serialization," https://www.datacamp.com/community/tutorials/pickle-python-tutorial, April 2015.

[3] A. Menon, "Linear regression using least squares," https://towardsdatascience.com/linear-regression-using-least-squares-a4c3456e8570, Sep 2018.

[4] R. Sah, "Simple machine learning model in python in 5 lines of code," https://towardsdatascience.com/simple-machine-learning-model-in-python-in-5-lines-of-code-fe03d72e78c6, Mar 2016.

[5] P. Grover, "Gradient boosting from scratch," https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d, Dec 2017.

[6] D. K, "Implementing gradient boosting in python," https://blog.paperspace.com/implementing-gradient-boosting-regression-python/, Dec 2019.

[7] scikit-learn developers, "sklearn.ensemble.gradientboostingregressor," https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html, n/a 2019.