Capstone Project

Johnathan Widney
December, 2018

I. Definition

Project Overview

In this project I hope to successfully create a proof of concept that we can use Machine Learning, and more specifically, Reinforcement Learning, to teach computers the patterns of programming.

More of our world is becoming exponentially complex. Technology is getting smarter, more advanced, and doing it all at an accelerated pace. To handle this, I think we can leverage Machine Learning to take on these complex problems. Machines have proven they can accomplish seemingly impossible tasks, from learning how to play video games from scratch, to detecting cancer before it spreads. With those thoughts in mind, I believe I can program an AI to learn how to program. Code is built up of 1s and 0s or represented as hexadecimal values. These are what the machines are great at processing, so why not teach an AI to program using those values? The benefits of this could be extraordinary. The machine would parse and code without having to compile a programming language or translate anything. This would lead to faster processing of data, decreased wait times, and maybe even new undiscovered functions. There are others who have built other AI frameworks that have the ability to code, but none that are taught to program with binary or hex code.

In this project, I will use a dataset of hexadecimal code that I compiled from scratch as input to the algorithm. The reason I chose hex instead of binary is that based on the tools I have available to me, finding the hex code equivalents of python code was much more attainable. Hex and binary are fairly close to each other in terms of functionality in computers, so it is an acceptable substitution. My dataset contains three columns of the following: beginning hex code, executed hex code, and the program label. I gathered the hex code using a hex code editor. I passed through the written python code that I wanted to use, and found it's hex code equivalent. To find the compiled hex counterpart, I took each line of code in python and imported it into a python3 terminal and retrieved the data that was stored in the terminal's __pycache__/ folder within my computer's system. That data was then put through the hex editor to find the compiled python code hex equivalent. Finally, I assigned a label to each of these values. I based the labels off of the type of function the code was performing. The labels

for the hex code pairs are “print”, “add”, “if”, “if/else”, and “subtract”. Five basic computing functions with varied content to add variety to the data. All of these things were manually entered into a python file that the algorithm will use.

Future iterations of this framework could allow an AI to perform complex programming functions and find patterns that we cannot.

Problem Statement

To solve this problem, I am using reinforcement learning techniques with a hand-built dataset of hexadecimal code. The idea is to have the AI observe the differences between the starting and compiled hex code to hopefully develop its own programming methods by finding the patterns with the code.

Metrics

The metrics I am using for this project are accuracy compared to the true program that is associated with the corresponding label. Since I intend for it to create its own code, I will test it against the python code for the particular request.

II. Analysis

Data Exploration

Due to the nature of the dataset, there are no outliers, anomalies, missing values, etc. The dataset was constructed by me, and thus, there are no discussions or analyses regarding this dataset. Nonetheless, there are important features to point out. This set is comprised of three parts, these are the hexadecimal equivalents of a .py file and a .pyc file, followed by a label.

Example of the dataset:

Beginning Hex Code	Executed Hex Code	Program Label
7072696E 74282268 656C6C6F 20776F72 6C642229	330D0D0A 98CFD75B 14000000 E3000000 00000000 00000000 00020000 00400000 00730C00 00006500	print

	64008301 01006401 53002902 7A0B6865 6C6C6F20 776F726C 644E2901 DA057072 696E74A9 00720200 00007202 000000FA	

The data was obtained by writing/executing python code, and entering it into a hex editor. For example:

Python code (.py)	Hex (.py)	Python compiled (.pyc)	Hex (.pyc)	Label
print("hello world")	7072696E 74282268 656C6C6F 20776F72 6C642229	3 fi*Ú[„@sedÉdS)z hello worldN)ŕprint©rr˙	330D0D0A 98CFD75B 14000000 E3000000 00000000 00000000 00020000 00400000 00730C00 00006500 64008301 01006401 53002902 7A0B6865 6C6C6F20 776F726C 644E2901 DA057072 696E74A9 00720200 00007202 000000FA	print

To obtain the compiled python code (.pyc file), I created a file that contained the python code I wanted to compile. I then opened a python3 terminal session, and imported the file. This saved the .pyc file to the __pycache__/ folder located in the same directory. By opening the file contained in the __pycache__/ with the hex editor, I was able to obtain the hex code for that file. I repeated this process for every line of python code in the dataset.

Exploratory Visualization

In this particular dataset, I only have two features: .py hex code and .pyc hex code. The reason for having these is I wanted to have two examples of the hex code, both before and after processing, that it could analyse and identify patterns between the two.

Algorithms and Techniques

Initially, I intended to implement the NAF algorithm (Continuous Deep Q Learning with Accelerated) for Reinforcement Learning when analysing the data and making improvements. However, upon further research, this algorithm is quite new. While it does have potential for future problems, currently there is little support or knowledge on how to implement this algorithm for a problem such as mine. Therefore, I will be using a modified DQN and Monte Carlo reward system for my project. These have greater support, and should achieve the desired goal.

Benchmark

Benchmark is a random forest decision tree classifier. The expected accuracy of this model is less than 10%. Whichever of these metrics is highest, either >10% or if the decision tree arrives at a higher percentage, will be the target for the advanced machine learning model. Both of these benchmarks are hypothetical at the moment, as I have still not been able to find a working solution.

III. Methodology

Data Preprocessing

There have been many setbacks and difficulties that I have faced in regards to preprocessing this data. The only preprocessing that I have realized that needed to be accomplished, is that despite the fact that computers process hexadecimal values better than any other type of value, it is extremely difficult to get the hex code interpreted by the code as a hexadecimal value. Most often, the code will consider the hex code as a regular decimal value. This leads to errors that prohibit the model from performing. I have tried forcing the data set to be viewed as a string type value, and found it unsuccessful. Currently, I am still in the process of finding a solution. The next implementation I will attempt is encoding the hex code before loading it and then decoding it when processing.

Implementation

This is how it went. The neural network architecture for the model is sound, there were no errors when executing the model. The only issue was that no insights could be gathered from the data. Meaning, the dataset could not be understood by the data parser.

Refinement

There are quite a few improvements that could be made with the model, namely, getting the dataset to work. Beyond that, I'm not entirely sure if the model would need further improvements. Once I have quantifiable results from the model, I can proceed with implementing alternative results.

IV. Results

Model Evaluation and Validation

In the current state, the model is not at the operating capacity that I would prefer. Most of the time spent in improving this model was in the dataset and formatting that to an operating capacity.

Justification

Even though the model is not working, I still think this is a good first step at trying to accomplish a task that has not been attempted at this level before. Through further experimentation, I believe I can get the model to perform excellently, not only at this task, but many others like it.

V. Conclusion

Free-Form Visualization

This model has what I think are the very first steps at accomplishing the tasks. After further iterations, I believe this will eventually achieve the goal. As the dataset has become difficult to manage, I do not currently have results to plot and show overall performance. In time however, I seek to complete this model and show the results on GitHub.

Reflection

Despite not getting the model to achieve the goal I wished it to accomplish, I believe I have learned a great deal from this project, and the course as a whole. I will take these skills I have learned with me to further develop my skills and further my career in artificial intelligence and machine learning. I thank everyone on the Udacity staff and those who have helped me alone this journey. I look forward to creating many more models and solving more problems that machine learning is suited to solve. Thank you!

Improvement

There are quite a few improvements that could be made to this model. There are many other ways that I can manipulate the dataset and attempt to have it be interpreted by the model. Namely, I still could implement a lambda function, or perhaps configure the model to interpret it as a Bag of Words model and then treat the rest as though it were a NLP model. Beyond that, I am sure there are many other methods, functions, and applications that I could use for this model. When I finish this project on my own time, I will be sure to document my findings and discoveries.
