Project Proposal

Udacity Machine Learning Engineer Nanodegree
Johnathan Widney

Overview:

- The project's **domain background** — the field of research where the project is derived;
- A **problem statement** — a problem being investigated for which a solution will be defined;
- The **datasets and inputs** — data or inputs being used for the problem;
- A **solution statement** — a the solution proposed for the problem given;
- A **benchmark model** — some simple or historical model or result to compare the defined solution to;
- A set of **evaluation metrics** — functional representations for how the solution can be measured;
- An outline of the **project design** — how the solution will be developed and results obtained.

Domain Background:

The domain background for this project is the heart of computer science: Code. This project will encompass programming, binary, machine language, metaprogramming, organic algorithms, and other core concepts. I aim to develop an AI that can program on its own with only receiving a simple command. At first, I will have it program simple functions such as arithmetic, simple network adjustments, and other simple/basic functions and methods of the python language. I think that having a tool such as this would allow more advanced projects to be completed, such that have never been attempted before. I believe the possibility of Quantum Machine Learning may be possible in the next few years after only a few iterations of this project. I believe that AI is the key to achieving great things in our lifetime. That is why I believe this problem should be completed. I also think that reinforcement learning will be an excellent version of machine learning for this problem. There aren't necessarily any patterns or randomness that needs to be solved. It's more of a procedural task, with certain steps that need to take place. Just like learning to walk, the AI could learn to program.

Problem Statement:

This project will attempt to use machine learning to learn machine language. Using a deep reinforcement learning framework, I believe I can teach an AI agent how to program using binary. The problem I wish to solve is that our world is becoming more and more advanced in terms of technology. With quantum computing around the corner, we will need AIs that can process and understand these complex systems. This is the first step. Teaching an AI the core fundamentals of programming, and how to accomplish this efficiently at the machine language level, will drastically improve computer systems; and prepare it for the future.

Datasets and Inputs:

The datasets for this project will be the official Python documentation library. From fundamental functions, methods, and objects, to sample code provided by the documentation. The documentation offers sample code when describing each function. This will be the majority of the code I will use as input to the AI, the remainder would be any code I would need to use to fill in any gaps. This sample code will range from simple arithmetic functions, to slightly more advanced methods. I plan to use a variety of sample code to teach the AI. Each code snippet will be preprocessed by compiling it into binary. The goal of the AI is to take the binary code and determine what task it is associated to. By providing a variety of code, I hope to train the AI to identify programing practices and develop its own way of programing. The AI will have code represented by binary, but not the named task. It will have to learn it and adequately provide its own program to accomplish the task.

Solution Statement:

In the learning phase, it will learn binary programming and learn the associated tasks to the code snippets. It will then go through a phase to validate its new learned skills. In the testing phase, the agent will be given a requested task, and it will program its own code. The code will be tested against example code snippets.

Benchmark Model:

There are a few AIs that are similar to the one I want to create. But to my knowledge, none use Python as their programming language. For my benchmark model I will use a simple random decision tree model and expect an accuracy of 10% or less.

Evaluation metrics:

The AI will be penalized for any errors that are raised in its code. Fatal errors will be penalized heavily. The AI will be rewarded for programming a function that performs the desired task, without any errors. The AI will also have a penalty for the code failing the task, even if no errors are raised. The type of errors that I will classify as penalties will be things like syntax, type, value, etc. Errors that are typical in program development. This will help train the AI to not make errors when developing its own programs. I will set numerical values to each class of error when developing the AI's architecture.

Project Design:

I will bring a list of sample code that is sourced from the official Python docs website, and other public domain websites that have free sample code. Those code snippets will then be put through a compiler to convert them into binary. That will be the dataset. There will be 2 phases to the project: Training + Validation, and Testing. After splitting the dataset into training and testing data, they will be put through the AI model. Because it is a reinforcement learning model, just like training an AI to walk and navigate through a defined space, I will train this AI to understand programming and "navigate" through programming a particular program with a loosely defined architecture. The reason for a loose definition is so the AI won't attempt to replicate the exact program by memorizing it. Rather giving it an idea of what to go for. The type of RL model I intend to use is an NAF model (source: https://arxiv.org/abs/1603.00748). The AI will be given sample tasks and it will provide its own code to accomplish the task. This will be validated against code snippets in the validation dataset that truly accomplish that task. The validations will be used to further improve the model in its ability to code. The AI will learn the patterns of the binary code snippets and devise its own programming structure. In the testing phase, the AI will be given tasks that are requested, and the accuracy of the AI will be tested against the testing dataset of code snippets. The target accuracy for this project is 87% or greater. Points will be awarded to the AI for code that has 0 raised errors, a program that completes the requested task, and done with a lower total file size in the code. The AI will be penalized for any and all errors that may arise in the code, certain errors will have larger penalties. It will also have penalties for writing a program that fails to complete the requested task, and for writing a program with many lines of code and having a high file size. The reasons for those rewards and penalties is that I want the AI to program for the correct task with no errors while also being efficient by writing the program in as little lines of code as possible; that also ties in with large file sizes. I plan to have the AI go through several iterations to achieve 0 errors in the provided code.