CMPSC 383
Multi-Agent and Robotic Systems
Spring 2017

Name: Colton J. McCurdy

Pledge:

1 Overview of Progress

Since the previous update one week ago, I have made significant progress on the system for using simple moving averages to determine when to buy and sell bitcoins. Not only have I made progress there, but I now have a better understanding of what this project aims to contribute. Prior to this, I was just focused on creating a tool that used simple moving averages to buy and sell with little consideration of buying and selling algorithms (e.g., how much you should buy and sell on any given day). While three approaches were originally considered: a greedy algorithm, a random approach and a more sophisticated, equation for determining the optimal amount to order at a given time. While the third approach seems to be the most interesting and from the name indicates that it would perform the best, an equation such as this does not seem to exist for bitcoin ordering as it does for purchasing stocks.

In the stock market, it is often recommended to purchase a variety — between 15 and 20 different — stocks instead of investing all of your money in on stock. However, since there is only one bitcoin, the equations for distributing your money to multiple entities does not exist. While you could look at bitcoin as a single currency in a market of many other currencies, this is out of the scope of this project. This would be an interesting problem to consider in future work.

Looking back at what I discussed in my previous update, a lot of what I had in mind for this project still stands. For example, I talk about releasing the tool to GitHub and using certain algorithms for determining when to buy and sell. I have built on what I called a framework in the last update. I would say that all but the analysis of the technique is finished. With that being said, implementation of the tool is being wrapped up. There are minor additions that will make the analysis stage much easier, such as writing the data from each step to a CSV file to then be read in and evaluated in the R programming language.

2 Tools

As mentioned earlier, while I have had experience creating a bot that would analyze the price index of bitcoins, it was not created in a way that I would have liked it to be. Also, I was not able to contribute as much as I would have liked due to my lack of experience. This project provides me with an opportunity to create this tool using the programming language of my choice, namely the Go programming language.

Along with leveraging the power of Go for the implementation of the bot, I will leverage the capabilities of the R programming language for analyzing the data collected from each configuration

of the implemented algorithms. I plan to compare the worth — the total value of the assets — associated with each configuration of a technique over time.

2.1 Google's Go Programming Language

I have recently been particularly interested in the Go programming language — often just referred to as Go — due to its increasing popularity in industry. Go's syntax is often compared to a combination of Python and C, it often requires minimal code versus a more verbose language such as Java. Not only have I found this enjoyable, Go is often recognized for its ability to comlete tasks concurrently. Go was designed with concurrency at the top of the list of objectives of language features. In this project I have leveraged the power of Go's hybrid, lightweight threads, called goroutines.

2.2 The R Programming Language

Especially in the last year, I have done a lot of work in the R programming language, often just referred to as Rlang or R. R is widely used in both industry and research to effectively and efficiently evaluate trends in data. I have personally been able to experience the power of R when needing to make production-grade plots very quickly given a large amount of data. There is one key notion to keep in mind when working in R to make your workflow easier and more efficient, it is always keeping your data in a tidy dataframe.

This refers to the structure of your data. In a tidy dataframe, each row is an observation with all of the information about that observation available. For example, in this work, both simple moving averages evaluated in a given observation will be available. Observations will be days of the year, and because I am comparing multiple combinations of the simple moving averages, there will be data available for each combination, for each day.

2.3 CoinDesk

One of the issues that I faced when trying to obtain stock information was finding an API to connect to to obtain a given stock price index. While searching for an API to do this for the stock market, I came across CoinDesk which did exactly what I wanted except was focused on the bitcoin price index. After visiting CoinDesk's API website and seeing the clear documentation and useful examples, I was sure that this was what I wanted to work with. Not only does CoinDesk have routes for obtaining the current bitcoin price index, they have routes to obtain historical data for any given date range, which is exactly what I would need to perform the moving average calculations. For these reasons, I changed the focus of my project from the stock market to the bitcoin market.

The CoinDesk API also acts as another agent, which is focused on obtaining the bitcoin market price. It is with this agent that the buying and selling agent communicates to obtain the information for each day necessary to determine whether on that day it is better to buy or sell bitcoins.

3 Algorithms

In this bitcoin bot, I have implemented the most popular algorithm for determining when it is best to buy and sell bitcoins, namely simple moving averages. These various simple moving averages are combined and tell a person or a bot when to buy and sell based on when they cross. Simple moving averages are often considered the baseline technique for determining when to buy and sell. Because of the minimal complexity of these algorithms, I have been able to complete a bot that can effectively buy and sell. In future work, I could see myself adding additional algorithms to this tool to then compare to the baseline techniques.

3.1 Simple Moving Averages (SMA)

The simple moving average is often the baseline technique when determining when to buy and sell stocks. This technique has an infinite number of configurations. The most common configuration considers the price index from 50 days ago to that of 200 days ago. What this does is help smooth out votality The values of 50 and 200 are chosen because the curve being evaluated is smooth. Shorter date ranges while more close to the actual price index are much more volatile.

To calculate the simple moving average, you can use the following equation, where n is the number of days being evaluated. When considering two configurations of the moving average, when they cross is when the bot would buy and sell.

$$SMA_n = \frac{i_1 + i_2 + \cdots + i_n}{n} \tag{1}$$