Problem Chosen
C

2022
MCM/ICM
Summary Sheet

Team Control Number
2214934

Creating Trading Algorithms for Past Markets

We develop multiple algorithms to maximize profit through buying and selling both bitcoin and gold during a five-year period from 9/11/2016 to 9/10/2021. We take into account the commission prices for both assets, and make decisions about whether to buy, hold, or sell, on any given day. With the data provided, we use some statistical methods to try and predict the future of what will happen to the prices. Once we have an algorithm, we implement it in a coded simulation to see how much money it can make. Finally, we analyze the assumptions we make, how they may have affected our strategies, and we write a memorandum to the trader who posed us this problem.

Contents

List of Figures

List of Algorithms

1 Introduction

Gold and bitcoin are currently two of the most volatile and voluminously traded assets in the financial markets. Since the assets are traded every second, market traders use complex algorithms in order to determine the most profitable trade. Finding the most profitable algorithm requires numerous computations and conditions. There are many options and strategies to go about finding such an algorithm.

Our paper goes into that process, as we attempt to find the most profitable trading strategy using various statistical methods and tactics. Some methods may include complex mathematical algorithms, but we tried to stick with simpler techniques. We will present some of our attempted strategies and their results from them. Some strategies are more successful than others resulting in a larger return. We also analyze our algorithms and our assumptions to determine how we could improve them.

1.1 Problem Overview

Our goal is to find the most profitable trading strategy over a five-year period from 9/11/2016 to 9/10/2021 in the bitcoin and gold market. As an example, looking at just the bitcoin market, we might expect an optimal trader to make the following trades:

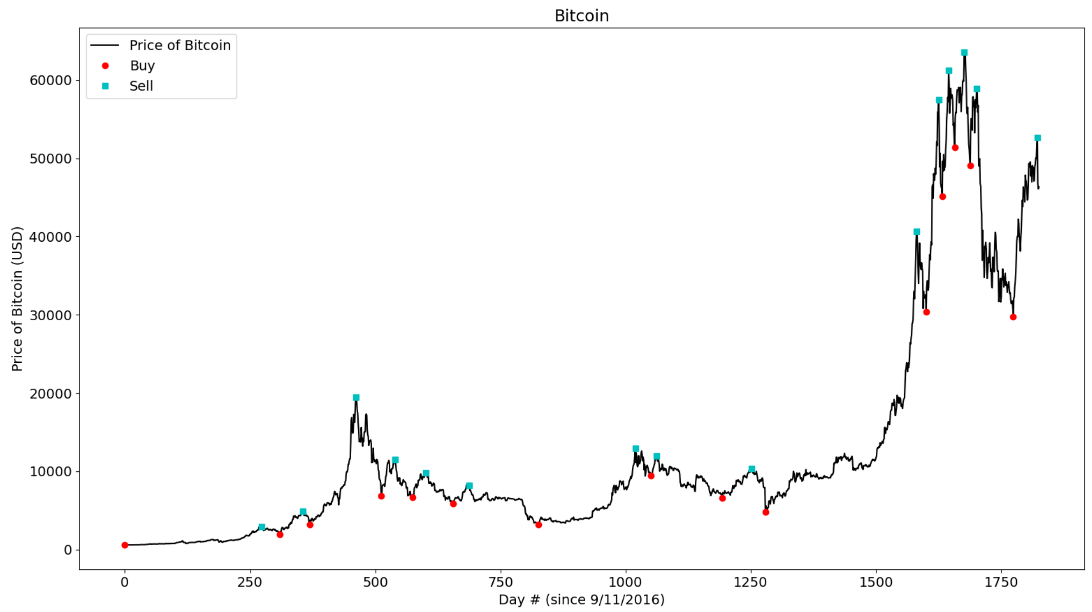

Figure 1: Optimal trading strategy for bitcoin

This idealized scenario would have the trader ending with $61,396,361. Of course, we cannot expect to make such perfect trades, and we must also take into account the gold market.

1.2 Constraints

- We start with \$1000.
- The trading period is from 9/11/2016 to 9/10/2021.
- There is a commission cost for buying and selling gold at 1%.
- There is a commission cost for buying and selling bitcoin at 2%.
- Gold can be traded only on the weekdays, and not on some holidays.
- Bitcoin can be traded every day.

1.3 Variables

- C = amount of money you currently have (USD)
- G = amount of gold you currently have (troy ounces)
- B = number of bitcoins you currently have
- p = today's price for one asset (USD)
- Δp = the difference between yesterday's price and today's price (USD)

Then we have our current portfolio as $[C, G, B]$.

1.4 Concessions/Assumptions

- To simplify, we end up using the same algorithm for both gold and bitcoin.
- We do not attempt to trade both assets at the same time.
- When we decide to sell an asset, we sell everything we have of that asset, and when we decide to buy an asset, we buy as much as we can. This leaves us with no money to be able to hold multiple assets at once.

1.5 Creating a Verification System

To know whether our algorithms were any good, we decided to create a system for checking how well they did. This was useful for gut-checking our ideas, and gave us a nice framework for analyzing the models we came up with to see how they could possibly be improved.

We created the system using the programming language Python. It is structured in a way that we can easily switch out the algorithm it uses. After defining an algorithm and testing it, we get a graphical output of where on the timeline we traded and sold, as seen in figure 1. This is useful for analyzing our choices so we can further tweak the algorithm to make it better.

The system works by first parsing the provided data and lining up the dates (since the gold data skips weekends). Then it loops through the parsed data and chooses whether to buy or sell bitcoin, and then whether to buy or sell gold. Every transaction has its associated commission cost accounted for. It keeps track of when these events happen and returns a graph at the end for visualization.

2 Preliminary Model Ideas

There are a number of different strategies for trading. In general, we would want a trading algorithm to buy when the price is low, and sell when the price is high. However, deciding when the price is "high" or "low" is not at all straightforward. Additionally, we have to juggle two different assets at once. To start, we will consider just trading one asset.

2.1 Naive Approach

The naive approach would be to look at the change in the price from yesterday to today, and then make a choice assuming the same price change will happen tomorrow. For example, if you think the price will go up by a lot tomorrow, you might decide to buy today so that you can sell tomorrow and turn a profit. However, if the price is currently going down, you may want to sell what you have before it becomes worth less than you bought it for. This will work well on some days, but the market rarely does the exact same thing more than a couple days in a row. Adding on to this is the fact that you will always be a day behind the trend. Ideally, you would have predicted the price rising today even if it fell yesterday.

Improving on this approach, we can think about keeping track of the price changes over the last, say, seven days. This number of days can change, but we may not want to keep track of too long a time period, since the distant past cannot be expected to heavily affect the price today. It is also important to note that we would like to keep track of price *changes*, as opposed to absolute prices. Then the algorithm might look something like "if the change in price between yesterday and today is 10% greater than the average change in price over the last seven days, then we should buy". Keeping track of changes, especially in terms of percent changes, helps to mitigate the effect of the changing price scale.

2.2 Statistical Approach

Now we look at using methods from statistics to help us decide what to do. In the previous models we were looking at averages, which is a nice idea, but the frame of reference is not quite right. If we take some tools from statistics, we can be more rigorous about our decisions.

We define $\overline{x}$ as the average change over the last seven days and s as the standard deviation of that set. s tells us how close together our data is, so for example, if s is very small, it means the last seven days saw very little variation in the price. If s is very large, then there was a lot of variation in the last seven days. We can also look at how many standard deviations the current day's price change is from the mean with the following formula:

$$n = \frac{\Delta p - \overline{x}}{s} \tag{1}$$

where n is the number of standard deviations away from the mean that Δp is.

With these variables defined we can think about more algorithms. One idea is to only buy or sell if $|n|$ is high (in other words, Δp is much different from the last seven days). Another is to buy if the price has been decreasing recently (s is small and $\overline{x}$ is negative) and to sell if the price has been increasing recently (s is small and $\overline{x}$ is positive).

Another idea was to try and predict the future stock price using some of these statistical measures, along with a little bit of randomness. This strategy would sometimes get lucky and make a great choice, but the opposite might happen as well.

2.3 Other Strategies

We had many other ideas that we did not have time to implement or test. One idea was to take inspiration from the classic "Stopping Problem" (see [1]), which in this context would ask when to stop looking and start buying (or selling). There are fancier applications of this problem which are designed to target stock trading (see geometric brownian motion [2]), but in this context we might spend a little bit of time every month taking samples and then spend the rest of the time waiting for the next best opportunity to take action.

We also considered using multiple time periods, like 30 days as well as 7 days, and then comparing how the averages of them differed. This information could be useful in determining whether the current day's price is significant or not.

3 A Closer Look at the Algorithms

The main part of the algorithms is in deciding whether to buy or sell an asset on any given day, but there were a couple other things to take into account:

- When buying an asset, keep track of the amount you bought it for. Then, when you sell that asset, do not sell it for less than you bought it for (with the commission price included).
- Our algorithm tells you when to buy or sell, but sometimes you will not be able to. So, we include some checks to make sure you do not buy when you can't, and you do not sell when you can't.

The first algorithm we implemented was using statistics from the last seven days in order to decide whether the price was going up or down the next day. We get n the same way as in equation 1, using the standard deviation and the mean. If the number of standard deviations away from the mean is more

Algorithm 1 Using n to decide whether to buy or sell (or hold)

```
buy ← False
sell ← False
if $n > 2$ then
    buy ← True
else if $n < -2$ then
    sell ← True
end if
```

than two, that means that the difference in price between yesterday and today was large and positive. Given, this, we might expect the price to continue to rise, so we should buy now before it gets too high in hopes of selling when it starts to fall again. Similarly, if n is less than negative two, it means the price just fell significantly and so we should sell before it gets too low.

This algorithm has room for modification, as we can tweak the number of standard deviations away before we buy or sell. We landed on the number 2 somewhat arbitrarily. If the number is too low then we would be buying and selling all the time, but if it is too high, we would rarely trade.

The next algorithm we implemented uses the standard deviation s and the mean xb, along with the price of the asset p on the current day, to make a prediction about tomorrow's price. The way

Algorithm 2 Using s, xb, and p to predict tomorrow's price

```
Require: $r \in (0, 1)$ is random
  buy ← False
  sell ← False
  predicted_change ← $xb + (s * r)$
  pct_change ← predicted_change $/p$
  if pct_change $> 0.05$ then
      buy ← True
  else if pct_change $< -0.05$ then
      sell ← True
  end if
```

we predict the next day's price is by increasing the average change over the last seven days by some random multiple (between 0 and 1) of the standard deviation. Then if we predict the change to go up by more than 5%, buy. If we predict it to go down by more than 5%, sell. Again, these parameters are somewhat arbitrary, and we could possibly improve the algorithm by being more rigorous about our choices. We note that this algorithm follows a strategy that is similar to how a Markov process works ([3]).

4 Results

Now we look at the graphs of algorithms 1 and 2 in figures 2 and 3. In figure 2, the ending portfolio was [9268.7, 0, 0]. From the graph we can see that a lot of trades were made, both in the bitcoin and gold markets, but many of them were not very good trades. There were a few cases of selling at a lower price than an asset was bought for, which may have been the result of a bug in the code since we were specifically trying to avoid such a scenario. We also notice a slight pattern of buying very quickly after selling, which suggests that the algorithm is trying to buy more often than it should. Given these results, we may want to tweak some of the parameters or use other data to inform the decisions in this algorithm.

In figure 3, the ending portfolio was [0.0, 0, 0.6028], which equates to approximately $27,951 on the final day of trading. This algorithm only traded a total of nine times, and never even touched the gold market. We suspect this was in part due to the fact that the algorithm bought bitcoin quickly after selling it, and so there was not much of a chance for it to consider how the gold prices were changing. It ends up holding bitcoin for a long time, which results in a decent ending portfolio, but it is clear that the algorithm was not doing what we really intended for it to do. This result also suggests that our parameters are off, and that maybe we should be more restrictive about when to buy.

Ultimately, we think these algorithms are on the right track, but they are nowhere near optimal. With more time, we would like to make some changes to their parameters.

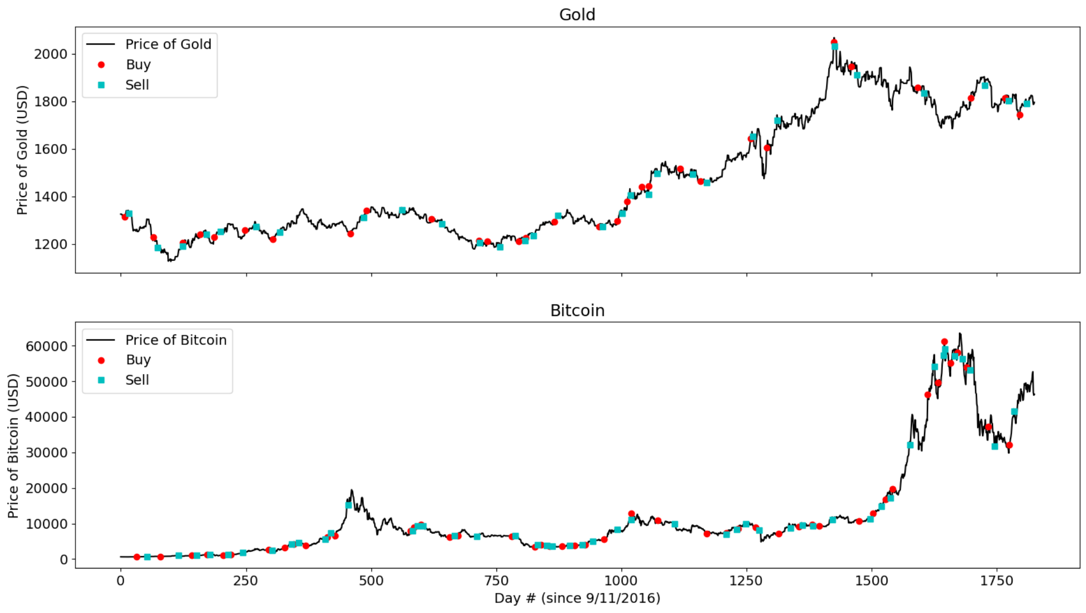

Figure 2: First strategy, based on algorithm 1

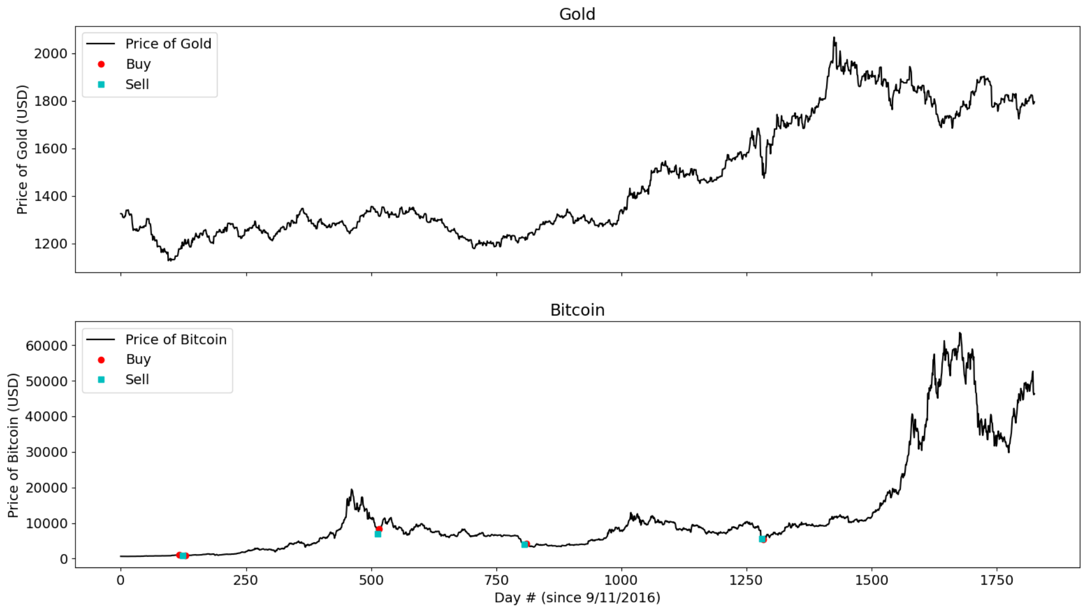

Figure 3: Second strategy, based on algorithm 2

5 Analysis of Assumptions and Limitations

5.1 Assumptions

Our first assumption was that we must use the same algorithm for both gold and bitcoin. This idea is very limiting since during some periods, bitcoin was prone to very high volatility, but other periods had gold changing a lot. Additionally, gold and bitcoin are unlikely to have the same patterns or signals that an algorithm would latch on to, so using the same algorithm for both may result in a poor overall performance.

Another potential flaw in our algorithm was not trading both assets at the same time. Our algorithm traded either bitcoin or gold at a given time, but not both. There could be periods of time where one asset is more profitable than the other, and the algorithm would only hold an asset until it decides to sell. So if gold was profitable and the portfolio was holding bitcoin, it would hold bitcoin until deciding to sell, missing an opportunity to buy gold.

Our algorithm also only traded the entire amount of that asset in that given time. This could lead to a loss of gains, as there were potential trades that were not made because our portfolio was made entirely of one asset. This also feeds in to the previous flaw.

5.2 Limitations

The main limitations were the commission price and the starting amount of \$1,000. Since we did not end up differentiating between gold and bitcoin in our algorithms, the differing commission prices would only come up when deciding whether it is worth it to sell at the current price. Changing the bitcoin commission price up from 2% may have a slight effect in making bitcoin trading less frequent, but would not otherwise do much to change the trading scheme.

The same can be said about the starting amount of \$1,000. Increasing this amount to, say, \$2,000 would likely make the ending portfolio worth more, but otherwise it is unclear how changing this starting amount would affect the trading.

6 Memorandum

Hello,

We came up with a couple of strategies for trading that turn a profit, but they come with some caveats. The most optimal result we achieved returns about $25,000 to $50,000 after the five year period is over, depending on how lucky we are. This was our so-called algorithm 2, where we attempt to predict the price of the asset on the next trading day using statistics based on the changes in price over the last seven days. It would only buy an asset if the predicted price was greater than 5% of the current price. While the algorithm allows for the trading of both bitcoin and gold, all of our simulations only traded bitcoin, completely ignoring the gold market. It also makes very few trades, which ends up being profitable since the price of bitcoin rises rapidly, but clearly this result is not expected to transfer to other assets.

Our other algorithm makes a more meager return of $9,000, which is still a substantial gain, but nowhere near optimal. This algorithm looks at the difference between today's price and the last seven days' prices, and if it is significantly different (which has some room for interpretation), we take action and either buy or sell. The biggest flaw here is that our predictions are lagged behind by at least one day, and since the market changes so quickly we end up making a lot of bad decisions.

Both of these algorithms also suffer from having to make choices regarding their parameters. For the previous algorithm we chose 2 as the magic number of significance, and for algorithm 2 we chose 5%. However, these numbers may not be optimal and we did not have too much reasoning behind our choices aside from basic intuition.

In addition to the basic logic of the strategies, we also include a "safeguard" to make sure that at the very least, we are not losing money. This works by ensuring we never sell for less than we bought for. However, this does lead to fewer trades as it could take a longer period to find a trade that fulfills this condition.

There are plenty of adjustments that could be made to the algorithms to improve them. We may want to look into how different strategies for the two assets could affect the result. Coming up with a way to effectively interplay between the markets would most likely be very complicated, but if done right, could result in a very successful algorithm. Additionally, adding more conditions to check, while also increasing complexity, may reduce the risk of making poor choices. We may also consider including the commission price as a proper variable in our decision making process, instead of leaving it only to dictate how expensive a given purchase or sale is.

Ultimately, while we have found many fascinating results, we cannot say we have reached a conclusion about an optimal model for this scenario. We would like to look into other potential models as alternatives to the ones we have posed. For example, we might take inspiration from the existing ideas behind the problems of optimal stopping, or other financial prediction models. We have learned a lot from this venture into the world of financial trading, and we hope you consider our findings interesting.

Thank you.

References

[1] “When To Stop.” https://www.ruth-ng.co.uk/the-optimal-stopping-problem/, Mar 17 2018. [Online; accessed 2022-02-21].

[2] “Geometric Brownian Motion.” https://stats.libretexts.org/@go/page/10406, Aug 10 2020. [Online; accessed 2022-02-21].

[3] “Introduction to Markov Processes.” https://stats.libretexts.org/@go/page/10288, Aug 10 2020. [Online; accessed 2022-02-21].