Problem Chosen
C

2022
MCM/ICM
Summary Sheet

Team Control Number
2217249

A CNN-LSTM based Semi-fuzzy Strategy for Gold and Bitcoin Trade

Summary

Gradual development in futures market draws more and more attention to this area, and more and more new products are gradually entering the market including bit coin and ethereum, causing it to become more and more complex. Since different products in futures markets possess drastically different price curve, and requires product-specific strategy, the demand for automatic decision-making model is becoming more and more urgent. Our team is asked to develop a decision-making model that relies only on the price data of gold and bit coin for the last five years to provide optimal strategy considering the gold market and the bit coin market simultaneously. To achieve this, we were asked to solve four problems.

For problem 1, We divide this problem into two sub problems: The first sub problem is to develop a prediction model that predicts the short-term price trending of these two products. For problem, we developed a neural network based on the combination of Convolutional Neural Network (CNN) and Long Short-term Memory (LSTM). Due to its structure, our model is capable in extracting both the spatial information and the temporal information of the input, and achieved nearly 60% accuracy. The second sub problem is to develop a strategy model that gives the optimal strategy based on the prediction of short-term price trending in both markets. To achieve this, we firstly assign increasing probability we got by CNN-LSTM into a fuzzy matrix, then define four parameter matrices storing trading rules. We put it in to the history market, and use Simulated Annealing (SA) to learn the parameters.

For problem 2, we will show our model achieves good balance between profit and risk. We achieved this by introduce a hyperparameter to control Max Drawdown. For each threshold, we trained 50 epoches and computed the average Annualized Return. We drawed out their relationship, and found that it's best to control Max Drawdown to be under 18%, we then achieve a Annualized return as high as 132.6%, with just 17.74% Max Drawdown, and the initial 1000 dollar becomes 39498.2 dollars after five years. Also we prove the superiority of the model theoretically. We equate the process of model improvement to the approximation of linear function to measurable function, and prove that under the condition of accurate prediction and sufficient computation force, we can approach the optimal decision indefinitely.

For problem 3, we analyze the accuracy and sensitivity of our model, proving that our model is accurate and stable for different parameters. We view this from two aspects: One is that, when transaction costs changed, we looked at how we could adjust our strategy, and we found that this change could be easily addressed by scaling up or scaling down the transaction size; Another is that, When the market price oscillates within a certain range, the Annualized Return of this strategy is observed, and it is found that when the amplitude is not too large, the returns are basically stable.

For problem 4,we write a Memorandum outlining our work, the results and the advantages of the model. At the same time, we also propose the areas where the model can be improved and where further support is needed.

Keywords: Quantitative Trading, Machine Learning, Fuzzy Math, Simulated Annealing

Contents

1 Introduction

1.1 Problem Background

Quantitative trading, in short, uses advanced mathematical models to replace human subjective judgment and uses Internet technology to screen a variety of "high probability" events that can bring excess returns from massive historical data to formulate strategies. This computer-based strategy can reduce the impact of investor sentiment fluctuations to a certain extent, and avoid making irrational investment decisions when the market is extremely fanatical or pessimistic. In short, write software programs based on historical data to monitor the trading situation in real-time, and automatically execute the buying and selling operations by setting conditions. Due to the increasingly mature quantitative trading market, market traders buy and sell volatile assets frequently, intending to maximize their total return. There is usually a commission for each purchase and sale. Two such assets are gold and bitcoin.

Gold has been used as a general equivalent by humans since its discovery. Gold trading has a long history, and gold prices tend to remain stable and rarely change significantly. In contrast, bitcoin is an emerging concept, which was born in 2010. Before 2017, little attention was paid to the bitcoin market, but after the concept of blockchain was put forward, bitcoin has attracted more and more attention and ushered in a leap in price and transaction volume in 2020. Unlike gold, the price of bitcoin is relatively unstable and often rises and falls sharply.

Since gold and bit coin possess two very different markets, the model proposed in this paper has broad application prospects for it considers the gold market and the bit coin market simultaneously.

Considering the background information and restricted conditions identified in the problem statement, we need to solve mainly the following problems.

- Develop a strategy model that make decisions upon history prices of the two products
- Evaluate the strategy model and test its stability.

1.2 Our Work

The rest of the paper is organized as follows: Section 2 gives some assumptions and justifications of our model. Section 3 list the notations used in the model. In Section 4, we first developed a prediction model based on CNN and LSTM in Section 4.2. Upon this, we derived our strategy in Section 4.3. We then test its performance and analyze the sensitivity in Section 5. We give some further discussion in Section 6 and a brief conclusion in Section 7.

2 Assumptions and Justifications

Assumption 1: Don't consider the impact of other sources of income and expenditure.

Justification 1: To better train and evaluate models, we need a closed trading environment.

Assumption 2: Our trading activities do not affect market prices.

Justification 2: The amount of money we hold relative to the Bitcoin and Gold markets is too small to have an impact on the market.

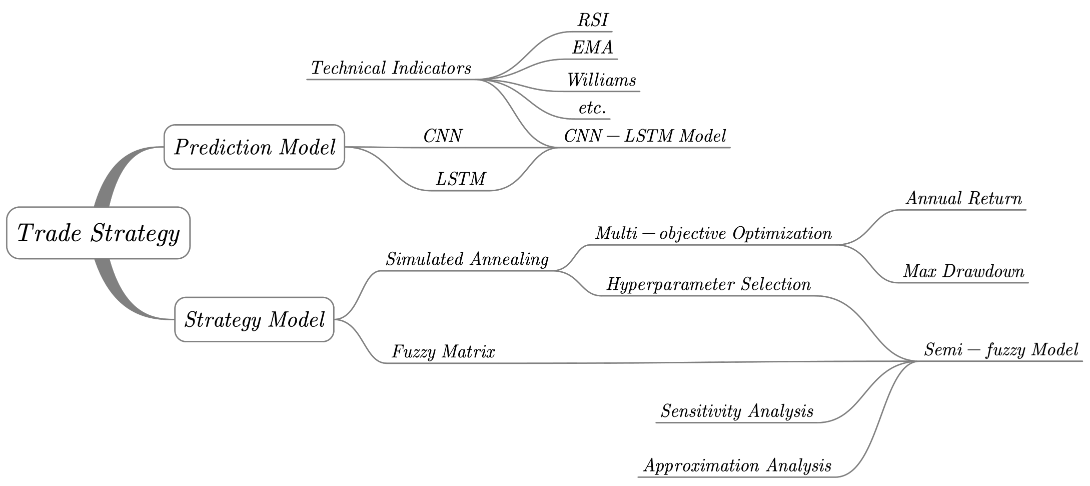

Figure 1: Our work

Assumption 3: Gold prices are unchanged on closed days.

Justification 3: Since there is no transaction, the price can be assumed to remain the same.

Assumption 4: The days gold is closed will also have an impact on subsequent price changes.

Justification 4: Obviously, if trading is closed for a long period of time, then prices will change dramatically.

3 Notations

The primary notations used in this paper are listed in Table 1.

Table 1: Notations of my paper

Symbol	Definition
$[C,B,G]$	Quantity of cash, Bitcoin and Gold one holds
$price_i$	i$^{th} price in some dataframe.$
$P_{B(G)}(-i)$	Price of Bitcoin(Gold) i days before the targeted day
$\Phi^1_{B(G)}$	Probability of increasing Bitcoin(Gold) next day
$\Phi^2_{B(G)}$	Probability of increasing Bitcoin(Gold) the day after next day
S_i	i^{th} fuzzy domain
$C_i(X)$	Degree of membership of X for i^{th} spatial domain for X
$\mathbb{A}(\hat{\mathbb{A}}),\mathbb{B}(\hat{\mathbb{B}})$	Matrix contains strategy for spatial domains for (not) open day
$\pi_{ij}^{\mathbb{A}(\hat{\mathbb{A}},\mathbb{B},\hat{\mathbb{B}})}$	Strategy contained at $\mathbb{A}(\hat{\mathbb{A}},\mathbb{B},\hat{\mathbb{B}})_{ij}$
$C^{\mathbb{A}(\hat{\mathbb{A}},\mathbb{B},\hat{\mathbb{B}})}$	Matrix of joint membership degrees
$X \odot Y$	Multiply matrix X and Y element by element
$\Sigma(X)$	Total sum of all elements in matrix X
$K[i]$	i^{th} day's feature K counted from 2016.9.10, K can be P_G, C, X, etc.
$\pi \circ X$	π acts at X for some defined rules.
Ω	A tuple stores all parameters $(\mathbb{A},\mathbb{B},\hat{\mathbb{A}},\hat{\mathbb{B}},\alpha,\beta,\hat{\alpha},\hat{\beta})$ of model.

4 Model Training

4.1 Technical Analysis Indicator

To provide data for our model, we manually constructed the following 14 indicators which relies only on the price of the product [1]:

- Price: the price of the product
- Simple Moving Average (SMA) with 5 days period: $SMA_{t,5} = \frac{\sum_{i=0}^{4} price_{t-i}}{5}$, which is the average price of last 5 days.
- Exponential Moving Average (EMA) with 5 days period: $EMA_{t,5} = \frac{2}{6}price_t + \frac{4}{6}EMA_{t-1,5}$
- Smoothed Moving Average (SMMA) with 7 days period: $SMMA_{t,7} = \frac{\frac{6}{7}\sum_{i=0}^{6} price_{t-i} + price_t}{7}$
- Triple Exponential Average (TRIX) with 12 days period: The triple exponential average is used to identify oversold and overbought markets. $TRIX = \frac{EMA3_{t,12} - EMA3_{t-1,12}}{EMA3_{t-1,12}}$ where $EMA3$ is the EMA of EMA of EMA.
- TEMA: Tema is another implementation for the triple exponential moving average. $TEMA = (3 * EMA) - (3 * EMA\ of\ EMA) + (EMA\ of\ EMA\ of\ EMA)$
- Relatice Strength Index (RSI) with 12 days period: RSI chart the current and historical strength or weakness of a stock. It takes a window parameter. $RSI_{t,12} = 100 - \frac{100}{1+RS_{t,12}}$ where $RS_{t,12}$ is equal to the average value of the sum of the cumulative increase of the closing price within 12 trading days divided by the sum of the cumulative decrease of the closing price within 12 trading days
- Stochastic RSI: Stochastic RSI gives traders an idea of whether the current RSI value is overbought or oversold.
- change: The percentage change of today's price compared to yesterday's price
- log-ret: the logrithm of change
- delta with 5 days period: The difference between today's price and the price 5 days ago.
- Commodity Channel Index (CCI) with 14 days period: $CCI_{t,14} = \frac{TP_{t,14} - SMA_{t,14}}{0.015 MD_{t,14}}$ where $TP_{t,14} = \frac{max_{i=0}^{13} price_{t-i} + min_{i=0}^{13} price_{t-i} + price_t}{3}$ and $MD_{t,14} = \frac{\sum_{i=0}^{13}(SMA_{t,14} - price_{t-i})}{14}$
- Williams Overbought/Oversold index (WR) with 14 days period: WR is a type of momentum indicator that moves between 0 and -100 and measures overbought and oversold levels. $WR_{t,14} = \frac{max_{i=0}^{13} price_{t-i} - price_t}{max_{i=0}^{13} price_{t-i} - min_{i=0}^{13} price_{t-i}}$
- Moving Average Convergence Divergence (MACD): $MACD_t = (DIF_t - DEA_t) * 2$ where $DEA_t = \frac{2}{10}DIF_t + \frac{8}{10}DEA_{t-1}$ and $DIF_t = EMA_{t,12} - EMA_{t,26}$

Each piece of data fed into our model must contains last 30 days of all 14 above-mentioned indicator to predict the price trending of the next day or the day after next day.

We use the accuracy to measure the performance of our prediction model.

4.2 Prediction Model

To utilize our method, we first need to establish a prediction model that can predict the price trend of the two products, here, we utilized the popular CNN+LSTM architecture.

4.2.1 Convolutional Neural Network

Convolutional Neural Network (CNN) was first proposed in [2]. Compared to traditional Deep Neural Network (DNN), CNN utilize matrix convolution to extract the spatial information of input feature while maintaining a relative small amount of parameters due to shared weight across different channels. CNN is exceptional in image recognition thanks to its ability to extract spatial information of the inputs.

Typically an CNN contains 5 different kinds of layers: input layer, convolution layer, pooling layer, fully connect layer and output layer [3]. Usually, convolution layer and pooling layer appears in pairs. They mainly play the role of feature sampling on the sample. After sampling, it transfers the features to the fully-connect layer for further reconstruction. The latter is connected to the output layer to produce an output result. To help understanding, we provide the architecture of LENET-5 [3], a typical CNN, in Figure 2.

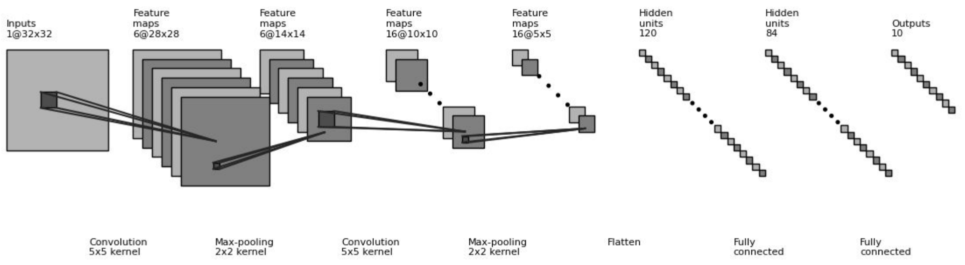

Figure 2: The Architecture of LENET-5

Convolution layer is the main innovative point of CNN. When data is transferred to the convolution layer, a convolution operation will be carried out in this layer. Here we give the formula of discrete convolution:

$$c(n) = (f * g)(n) = \sum_{i=-\infty}^{\infty} f(i)g(n-i) \tag{1}$$

Where f is the input data and g is the convolution kernel and c is the result. Since the input data of CNN is usually images, which can be regarded as an matrix of pixels. The actual formula of Convolution layer is as follows:

$$c_{i,j} = RELU(\sum_{x=0}^{z}\sum_{y=0}^{w} g_{x,y} f_{i+x,j+y} + b) \tag{2}$$

Where $c_{i,j}, f_{i,j}, g_{i,j}$ represents the value at entry (i,j) of the result, the input image and the convolution kernel. z and w is the height and width of the convolutoin kernel. b is the bias.

Basically, the process is to calculate an weighted average of each area covered by the kernel and the essence of convolution is to aggregate the information around each point. Thus, the convolution layer can extract the spatial information of input image. In practice, each convolution kernel is only responsible for extracting one kind of feature. Thus mature models will leverage several convolution kernels in one convolution layer to increase the performance.

Pooling layer is usually used to compress the output of the convolution layer while maintaining most of the information. Two popular ways are max pooling and average pooling.

Fully-connect layer is usually at the end of the CNN. It is responsible for high dimensional feature combination and the actual classification.

4.2.2 Long-Short Time Memory

LSTM structure is a variant of RNN structure first introduced in [4]. As a variant of RNN structure, the biggest difference of LSTM structure is that it draws lessons from the selective input and selective forgetting mechanism of the human brain, and introduces three "gate" structures of forgetting gate, input gate, and output gate, as well as a memory unit to selectively receive the information of the afferent neural network [5]. We give an overview of the LSTM structure in Figure 3

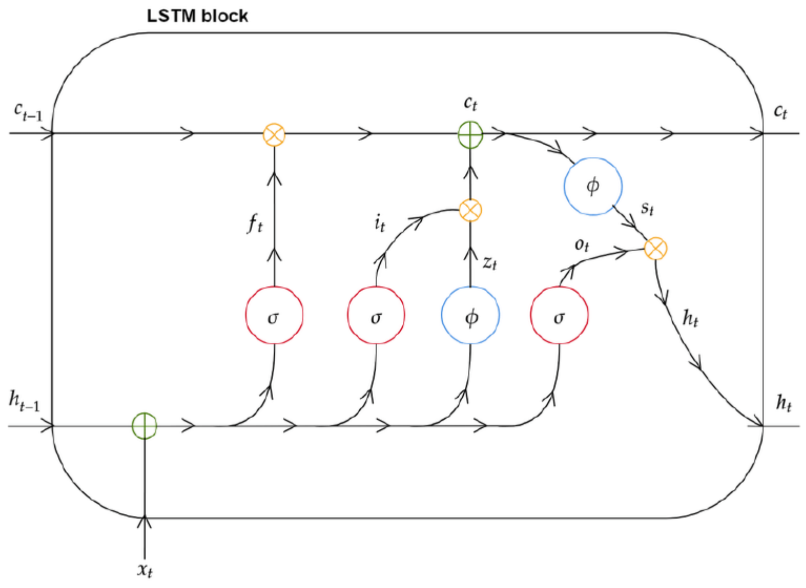

Figure 3: Structure of LSTM

In LSTM structure, the forgetting gate is responsible for receiving the information from the previous time to the current time point [5], that is, it determines how much of the memory unit state of the previous time c_{t-1} can be retained to the current time c_t. The purpose of setting the logic unit is to forget the previous useless information. The forgetting gate calculates the information retention degree f_t according to the output of the previous time stamp h_{t-1} and the input of the current time stamp x_t:

$$f_t = sigmoid(W_f[h_{t-1}, x_t] + b_f) \tag{3}$$

The function of the input gate is to determine which information can be added to the memory unit state of the previous time c_{t-1} and updated to the new memory unit state c_t according to h_{t-1} and x_t [5]. Firstly, the input gate calculates i_t from the output of the previous time h_{t-1} and the input of the current time x_t according to the following formula:

$$i_t = sigmoid(W_i[h_{t-1}, x_t] + b_i) \tag{4}$$

where W and b are weight matrix and bias. Then, it calculates z_t:

$$z_t = tanh(W_g[h_{t-1}, x_t] + b_g) \tag{5}$$

Then, the state of memory unit c_t is updated according to the following formula:

$$c_t = f_t \cdot c_{t-1} + i_t \cdot z_t \tag{6}$$

The output gate is responsible for determining how much of the information of the current timestamp can be outputted [5]. It calculates information output degree o_t from h_{h-1} and x_t accoding to the following formula:

$$o_t = sigmoid(W_o[h_{t-1}, x_t] + b_o) \tag{7}$$

Then, it calculates the output of the current timestamp:

$$h_t = o_t \cdot tanh(c_t) \tag{8}$$

From the above derivation, it can be seen that the structural design of LSTM is very suitable for modeling and analyzing financial time series. At the same time, the emergence of LSTM-based neural networks also solves the problems that cannot be handled by RNN-based neural networks and the problem of gradient disappearance and gradient explosion in the process of model training. By adding multiple "gate" structure logic units to the neurons of each layer, LSTM-based neural networks enable the bias to directly pass through the "gate" in the directional propagation, which improves the phenomenon of gradient disappearance and divergence in the process of error backpropagation, so that the gradient will not disappear completely no matter how far away from the current time. In addition, The gradient of the LSTM-based neural network is relatively stable in the propagation process, and the phenomenon of the same weight matrix multiplication will not occur in the gradient backpropagation like the RNN-based neural networks.

4.2.3 Overall Model

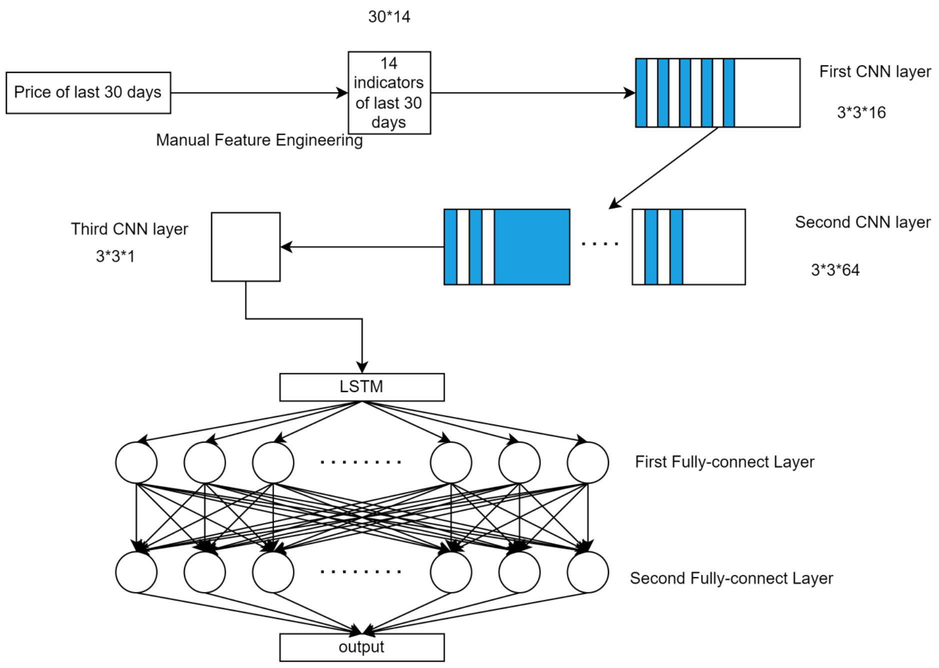
30*14
Price of last 30 days
14 indicators of last 30 days
Manual Feature Engineering
First CNN layer
3*3*16
Second CNN layer
3*3*64
Third CNN layer
3*3*1
LSTM
First Fully-connect Layer
Second Fully-connect Layer
output

Figure 4: Overview of Prediction Model

We establish our prediction model by connecting CNN and LSTM in which CNN is responsible for extracting feature information from inputs and LSTM is responsible for extracting sequential information from inputs [1]. The input of our model containing information from past 30 days, and in each day we mannually constructed 14 indicators only related to price as the feature of each day.These features is arranged as a 30*14 matrix to fit into our model. We give the overview of our model in Figure 4. The first fully-connect layer is responsible for calculating high level feature interaction, and the second fully-connect layer is responsible for the actual classification task. The output of our model will go through a sigmoid calculation to be projected into [0,1] to be regarded as the possibility for the price to go up.

We trained 4 prediction models to predict the price trend of Bitcoin and gold for the next day and the day after next day respectively. Here we provide the performance of our model in Table 3 where Bit 0 and Gold 0 represents the test accuracy for predicting the price trend of Bit coin and gold for the next day and Bit 1 and Gold 1 represents the test accuracy for predicting the price trend of Bit coin and gold of the day after next day.

Table 2: Performance of Prediction Model

Indicator	Test Accuracy
Gold 0	0.5872
Gold 1	0.5957
Bit 0	0.6011
Bit 1	0.5734

4.3 Strategy Model

4.3.1 Model Establishment

1. Strategy Model

For we have already trained prediction model $\Phi^{1,2}_{B(G)}$, we can now compute the predicted probabilities of increasing for both Bitcoin and Gold for both next day and the day after next day.

Here we need to consider the second one is because, when we decide our strategy, taking Gold as an example, we not only consider whether the price will increase, but also whether it will continuously increase, if it will, we will then keep more Gold in our hand until we see a tendency of decreasing. Imagine we don't care about that, we will sell most of our Gold once it may increase, then we may miss more potential benefits. To verify our idea, we will also see the model without continuously increasing probability later.

We now turn to our complete model. Firstly we assign probabilities into the divided fuzzy domains as below:

Domain	[0,20%)	[20%,40%)	[40%,60%)	[60%,80%)	[80%,100%]
Notation	S_1	S_2	S_3	S_4	S_5
Meaning	Should decrease	May decrease	Stable	May increase	Should increase

And we get the related membership degree functions:

$$C_i(x) = \begin{cases} \begin{cases} 1 & x \in [0, 10\%) \\ 1 - \frac{x-0.1}{0.2} & x \in [10\%, 30\%) \\ 0 & \text{otherwise} \end{cases} & i = 1 \\ \begin{cases} 1 - |\frac{x-0.1(2i-1)}{0.2}| & x \in [(20i-30)\%, (20i+10)\%] \\ 0 & \text{otherwise} \end{cases} & i = 2,3,4 \\ \begin{cases} 1 & x \in (90\%, 100\%] \\ 1 + \frac{x-0.9}{0.2} & x \in [70\%, 90\%) \\ 0 & \text{otherwise} \end{cases} & i = 5 \end{cases} \tag{9}$$

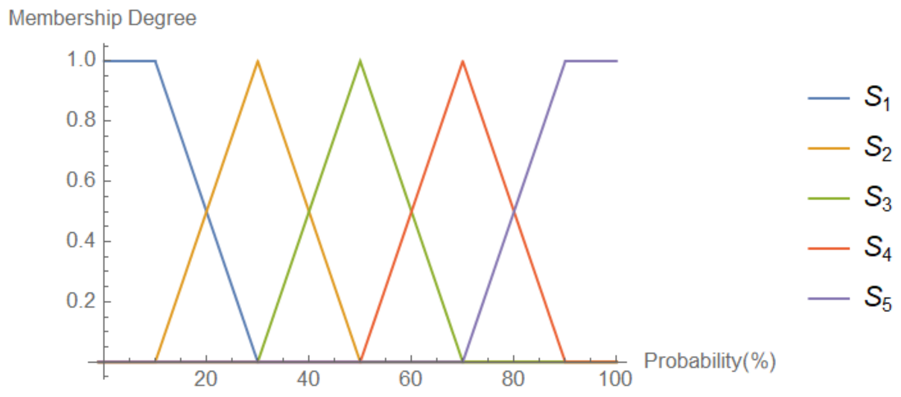

Figure 5: Membership degree functions

The reason for why we utilize fuzzy sets is that, it's hard to tell the true relationship between increasing probability and our strategy, applying fuzzy theory enabling us to use a decision rule in replace to a function, when we can got some unlinear relations [6]. And this model can be easily improved to a better or even the best one, which we will explain later.

We firstly see the strategy applied for the open days. Once we obtain prediction four models $\Phi^{1,2}_{B(G)}$, we then derive feature matrices of each day after October 25th, 2016, notated as X_i. We now focus on just a certain day with feature matrix $X = [X_B^{30}, X_G^{30}]$ consists of history 30 days features, then we map then to increasing probabilities, and compute their membership degrees for each set. which is:

$$C(\mathbb{A}, B(G), X) := \begin{bmatrix} C_1(\Phi^1_{B(G)}(X)) & C_2(\Phi^1_{B(G)}(X)) & \dots & C_5(\Phi^1_{B(G)}(X)) \end{bmatrix} \tag{10}$$

$$C(\mathbb{B}, B(G), X) := \begin{bmatrix} C_1(\Phi^2_{B(G)}(X)) & C_2(\Phi^2_{B(G)}(X)) & \dots & C_5(\Phi^2_{B(G)}(X)) \end{bmatrix} \tag{11}$$

Then we get the joint membership degree matrix as:

$$C^{\mathbb{A}} := C(\mathbb{A}, G, X)^T C(\mathbb{A}, B, X) \tag{12}$$

$$C^{\mathbb{B}} := C(\mathbb{B}, G, X)^T C(\mathbb{B}, B, X) \tag{13}$$

Here, we pay attention to elements of them, for example $(C^{\mathbb{A}})_{ij}$ is just equivalent to the membership degree for the probability pair $(\Phi^1_G(X), \Phi^1_B(X))$ to be belong to the two-dimensional fuzzy domain (S_i, S_j).

In our case, it's a 5×5 matrix, so we can define a same sized block matrix to store the strategy coefficients. that is:

$$\mathbb{A} := \begin{bmatrix} \pi_{11}^{\mathbb{A}} & \cdots & \pi_{15}^{\mathbb{A}} \\ \vdots & \ddots & \vdots \\ \pi_{51}^{\mathbb{A}} & \cdots & \pi_{55}^{\mathbb{A}} \end{bmatrix} \tag{14}$$

$$\mathbb{B} := \begin{bmatrix} \pi_{11}^{\mathbb{B}} & \cdots & \pi_{15}^{\mathbb{B}} \\ \vdots & \ddots & \vdots \\ \pi_{51}^{\mathbb{B}} & \cdots & \pi_{55}^{\mathbb{B}} \end{bmatrix} \tag{15}$$

Each $\pi_{ij}^{\mathbb{A}(\mathbb{B})}$ here stores a strategy when $(\Phi_G^1(X), \Phi_B^1(X))$ falls in the center of fuzzy domain (S_i, S_j). More precisely, take one π as example, it's defined as:

$$\pi =: \begin{bmatrix} G_{in} & G_{out} & P_{in} & P_{out} \end{bmatrix} \; are\ all\ rates \tag{16}$$

$$\begin{cases} G_{in} := Use\ G_{in}\ of\ Cash\ to\ buy\ Gold, \\ G_{out} := Sell\ G_{out}\ of\ Gold, \\ B_{in} := Use\ B_{in}\ of\ Cash\ to\ buy\ Bitcoin, \\ B_{out} := Sell\ B_{out}\ of\ Bitcoin. \end{cases} \tag{17}$$

with constrains as:

$$\begin{cases} G_{in} + B_{in} \leq 1, For\ we\ can't\ use\ cash\ more\ than\ Cash\ we\ have, \\ G_{out} \leq 1,\ B_{out} \leq 1, For\ we\ can't\ sell\ more\ than\ we\ have, \\ G_{out}G_{in} = 0,\ B_{out}B_{in} = 0, There\ is\ no\ reason\ to\ buy\ and\ sell\ at\ the\ same\ time. \end{cases} \tag{18}$$

Assume we have get all the π, then we should utilize the strategies for consideration of the increasing probability and the continuously increasing probability according to the computed joint membership degree matrices:

$$\pi_{\mathbb{A}} := \frac{\Sigma C_{ij}^{\mathbb{A}} \pi_{ij}^{\mathbb{A}}}{\Sigma C_{ij}^{\mathbb{A}}},\ \pi_{\mathbb{B}} := \frac{\Sigma C_{ij}^{\mathbb{B}} \pi_{ij}^{\mathbb{B}}}{\Sigma C_{ij}^{\mathbb{B}}} \tag{19}$$

For it's easy to see $\Sigma C_{ij} = 1$,it's equivalent to:

$$\pi_{\mathbb{A}} = \Sigma(C^{\mathbb{A}} \odot \mathbb{A}),\ \pi_{\mathbb{B}} = \Sigma(C^{\mathbb{B}} \odot \mathbb{B}) \tag{20}$$

Here $\Sigma(\cdot)$ means add by elements (here πs in the matrices). To bind these two consideration together, we introduce weight α and β, which are also need to be trained, then we get the final strategy as:

$$\pi_f = \alpha\pi_{\mathbb{A}} + \beta\pi_{\mathbb{B}},\ \alpha + \beta = 1,\ \alpha \geq 0,\ \beta \geq 0. \tag{21}$$

Now we have a strategy given the history data (once we finally trained our model), then we turn to the trading process. Similarly, we take one day as example, and compute the strategy π_f of it, and $\pi_f = \begin{bmatrix} G_{in} & G_{out} & P_{in} & P_{out} \end{bmatrix}$,and our present assets is $[C, B, G]$, then the process is as (here service charge of trading of Gold and Bitcoin is $\alpha_{gold} = 0.01$ and $\alpha_{bitcoin} = 0.02$):

I. Sell the Golds and Bitcoins to get more Cash.

$$\begin{aligned} &C \leftarrow C + (1 - \alpha_{gold})G_{out}G + (1 - \alpha_{bitcoin})B_{out}B, \\ &G \leftarrow (1 - G_{out})G, \\ &B \leftarrow (1 - B_{out})B \end{aligned} \tag{22}$$

II. Inherit the state from (16), then we buy Golds and Bitcoins. Assume the price of Gold and Bitcoin to be P_G and P_B:

$$\begin{aligned} &G \leftarrow G + \frac{G_{in}C}{(1+\alpha_{gold})P_G}, \\ &B \leftarrow B + \frac{B_{in}C}{(1+\alpha_{bitcoin})P_B} \\ &C \leftarrow (1 - G_{in} - B_{in})C, \end{aligned} \tag{23}$$

Then we conbine these processes together, and get that, for each day's trading, the asset state chandes as:

$$\begin{aligned} &(i)\ C \leftarrow C + 0.99G_{out}G + 0.98B_{out}B, \\ &(ii)\ G \leftarrow (1 - G_{out})G + \frac{G_{in}C}{1.01P_G}, \\ &\qquad B \leftarrow (1 - B_{out})B + \frac{B_{in}C}{1.02P_B}, \\ &(iii)\ C \leftarrow (1 - G_{in} - B_{in})C, \end{aligned} \tag{24}$$

We can view such a process as an interaction of some strategy π and $[C,B,G]$, and we may notate is as $\pi \circ [C,B,G]$

Similarly, for not the open day, we train another pair of strategy matrix $\hat{\mathbb{A}}$ and $\hat{\mathbb{B}}$, also weights $\hat{\alpha}$ and $\hat{\beta}$, with an additional constrain $G_{in} = G_{out} = 0$

We use $\mathbb{I}_{open}$ to indicate whether a day is an open day, which means

$$\mathbb{I}_{open} := \begin{cases} 1 & For\ open\ days, \\ 0 & Otherwise. \end{cases} \tag{25}$$

Now that's make a conclusion for our model.

It has these parameters to learn: $(\mathbb{A}, \mathbb{B}, \hat{\mathbb{A}}, \hat{\mathbb{B}}, \alpha, \beta, \hat{\alpha}, \hat{\beta})$

So we now let day i to be i^{th} day counted from 2016.9.11, it has assets $([C[i], B[i], G[i]])$, price $[P_G[i], P_B[i]]$ and history 30days feature $X[i]$, then the final strategy $\pi_f(\Omega)[i]$ when the given parameters $\Omega = (\mathbb{A}, \mathbb{B}, \hat{\mathbb{A}}, \hat{\mathbb{B}}, \alpha, \beta, \hat{\alpha}, \hat{\beta})$ is:

$$\begin{aligned} \pi_f(\Omega)[i] = \mathbb{I}_{open}[i](\alpha\Sigma(C^{\mathbb{A}}[i] \odot \mathbb{A}) + \beta\Sigma(C^{\mathbb{B}}[i] \odot \mathbb{B})) \\ +(1 - \mathbb{I}_{open}[i])(\hat{\alpha}\Sigma(C^{\hat{\mathbb{A}}}[i] \odot \hat{\mathbb{A}}) + \hat{\beta}\Sigma(C^{\hat{\mathbb{B}}}[i] \odot \hat{\mathbb{B}})) \end{aligned} \tag{26}$$

Where

$$C_{\mathbb{A}(\hat{\mathbb{A}})} = \left[C_1(\Phi_G^1(X[i])) \quad \dots \quad C_5(\Phi_G^1(X[i]))\right]^T \left[C_1(\Phi_B^1(X[i])) \quad \dots \quad C_5(\Phi_B^1(X[i]))\right] \tag{27}$$

$$C_{\mathbb{B}(\hat{\mathbb{B}})} = \left[C_1(\Phi_G^2(X[i])) \quad \dots \quad C_5(\Phi_G^2(X[i]))\right]^T \left[C_1(\Phi_B^2(X[i])) \quad \dots \quad C_5(\Phi_B^2(X[i]))\right] \tag{28}$$

Recall C_i to be the membership degree functions and $\Phi_{B(G)}^{1,2}$ to be the prediction model function. Then given sequential of $X[i]$, we now have a sequential of strategies. We call this kind of strategy "Semi-fuzzy", because the input is turned to a fuzzy domain matrix, while the strategy itself is not fuzzied.

2. Model Target

Our target is to make a balance between benefit and risk. To achieve this, we take both **Annualized Return** and **Max Drawdown** into consideration.

Annualized Return is used to measure a averaged benefit per year. We should know that, when make a strategy, we can seldom think like "I want to get a best return just 5 years later", it's unrealistic, what we want is, "I want to get a better return every year", that's why we use returns averaged by year rather than the final return.

To get this,we need to define some notations first. We use $\Pi(\Omega)[m,n]$ to represent the compounded effect of strategy to the n^{th} day. i.e.

$$\Pi(\Omega)[n] := \pi_f(\Omega)[1] \circ \pi_f(\Omega)[2] \circ \cdots \circ \pi_f(\Omega)[n] \tag{29}$$

Then we have:

$$[C[i], B[i], G[i]] = \Pi(\Omega)[i] \circ [1000, 0, 0] \tag{30}$$

Convert that to total assets $T[i] = C[i] + P_B[i]B[i] + P_G[i]G[i]$, we get our measure to be

$$Y = \frac{1}{5}\left(\frac{T[365]}{T[1]} + \frac{T[730]}{T[366]} + \frac{T[1095]}{T[731]} + \frac{T[1491]}{T[1096]} + \frac{T[1826]}{T[1492]}\right) - 1 \tag{31}$$

Max Drawdown is the maximum observed loss from a peak to a trough of a portfolio, before a new peak is attained. Max Drawdown is an important indicator of downside risk over a specified time period. More precisely:

$$R = \underset{k \leqslant j \leqslant 1826}{Max} \left\{ 1 - \frac{T[j]}{T[k]} \right\} \tag{32}$$

Finally, we use the hyperparameter r to represent for the max R we can tolerate, i.e. we reject strategy with Max Drawdown larger than r when training, the larger r is, the more attention we pay to profit; more attention we pay to safety conversely speaking.

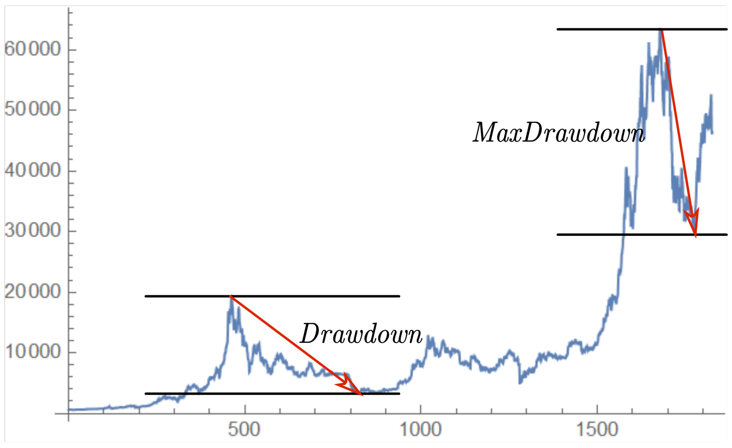

Figure 6: Max Drawdown

3. Model Training Conclusion

Before training, we now conclude our model to make it clear.

$$(\textbf{Input}) \begin{cases} Initial\ assets\ [C[0], B[0], G[0]] = [1000, 0, 0] \\ 30d\ History\ feature\ sequential\ \{X[i]\} \\ Price\ sequential\ \{(P_B[i], P_G[i])\} \\ Prediction\ model\ \Phi^{1,2}_{B(G)} \end{cases}$$

$$(\textbf{Target})\ Maximize\ \frac{T[365]}{T[1]}+\frac{T[730]}{T[366]}+\frac{T[1095]}{T[731]}+\frac{T[1491]}{T[1096]}+\frac{T[1826]}{T[1492]}$$

$$where\begin{cases}\Omega=(\mathbb{A},\mathbb{B},\hat{\mathbb{A}},\hat{\mathbb{B}},\alpha,\beta,\hat{\alpha},\hat{\beta}),\\ T[i]=(\Pi(\Omega)[i]\circ[1000,0,0])[1,P_B[i],P_G[i]]^T,\\ \Pi(\Omega)[n]:=\pi_f(\Omega)[1]\circ\pi_f(\Omega)[2]\circ\cdots\circ\pi_f(\Omega)[n],\\ \pi_f(\Omega)[i]=\mathbb{I}_{open}[i](\alpha\Sigma(C^{\mathbb{A}}[i]\odot\mathbb{A})+\beta\Sigma(C^{\mathbb{B}}[i]\odot\mathbb{B}))\\ \quad+(1-\mathbb{I}_{open}[i])(\hat{\alpha}\Sigma(C^{\hat{\mathbb{A}}}[i]\odot\hat{\mathbb{A}})+\hat{\beta}\Sigma(C^{\hat{\mathbb{B}}}[i]\odot\hat{\mathbb{B}}))'\\ \pi\triangleq[G_{in},G_{out},B_{in},B_{out}]\\ \pi\circ[C,B,G]:=\left\langle\begin{array}{l}(i)\ C\leftarrow C+0.99G_{out}G+0.98B_{out}B,\\ (ii)\ G\leftarrow(1-G_{out})G+\frac{G_{in}C}{1.01P_G},\\ \quad\ \ B\leftarrow(1-B_{out})B+\frac{B_{in}C}{1.02P_B},\\ (iii)\ C\leftarrow(1-G_{in}-B_{in})C.\end{array}\right.\\ C_{\mathbb{A}(\hat{\mathbb{A}})}=\left[C_1(\Phi^1_G(X[i]))\ \ldots\ C_5(\Phi^1_G(X[i]))\right]^T\left[C_1(\Phi^1_B(X[i]))\ \ldots\ C_5(\Phi^1_B(X[i]))\right],\\ C_{\mathbb{B}(\hat{\mathbb{B}})}=\left[C_1(\Phi^2_G(X[i]))\ \ldots\ C_5(\Phi^2_G(X[i]))\right]^T\left[C_1(\Phi^2_B(X[i]))\ \ldots\ C_5(\Phi^2_B(X[i]))\right]\\ C_i\ is\ stated\ by\ equation\,(6).\end{cases}$$

$(\textbf{Parameters})$ 404 $parames\ in(\mathbb{A},\mathbb{B},\hat{\mathbb{A}},\hat{\mathbb{B}},\alpha,\beta,\hat{\alpha},\hat{\beta})$ *and* 1 *hyperparameter* r
Where $(\mathbb{A},\mathbb{B},\hat{\mathbb{A}},\hat{\mathbb{B}})$ *are parameter matrices,each element stores four parameter*

$$(\textbf{Constrain})\begin{cases}\underset{k\leqslant j\leqslant 1826}{Max}\left\{1-\frac{T[j]}{T[k]}\right\}\leq r\\ \alpha+\beta=\hat{\alpha}+\hat{\beta}=1,\ \alpha,\beta,\hat{\alpha},\hat{\beta}\geq 0\\ For\ each\ element\ in\ \mathbb{A}(\mathbb{B})\ and\ \hat{\mathbb{A}}(\hat{\mathbb{B}}):\\ \begin{cases}G_{in}+B_{in}\leq 1,\\ cG_{out}\leq 1,\ B_{out}\leq 1,\\ G_{out}G_{in}=0,\ B_{out}B_{in}=0.\end{cases}\\ For\ each\ element\ in\ \hat{\mathbb{A}}(\hat{\mathbb{B}}),additionally:\\ G_{in}=G_{out}=0.\end{cases}$$

4.3.2 Simulated Annealing Training

1. Simulated Annealing

Here we use simulated annealing (SA) algorithm [7] to perform random search optimization for all parameters in the strategy. The parameter budget is $\mathbb{P}=(\mathbb{A},\mathbb{B},\hat{\mathbb{A}},\hat{\mathbb{B}},\alpha,\beta,\hat{\alpha},\hat{\beta})$.

Simulated annealing algorithm is a general probabilistic algorithm which is used to find the optimal solution of a proposition in a large search space. Simulated annealing algorithm is derived from the principle of solid annealing, solid heating to high fully, then let it slowly cooling, finally reach the ground state in normal temperature, can reduce to a minimum. [8]

The way to generate new solution here is add a noise $\nu\sim U(0,1)$ belongs to uniform distribution to the last parameter. The scale of noise is propositional to temperature T. Take the parameter $\mathbb{A}$ as an example.

$$\mathbb{A}'\leftarrow\mathbb{A}+kT\nu \tag{33}$$

We set the temperature update schedule as always discounted by a factor γ.

$$T' \leftarrow \gamma T \tag{34}$$

Here we show the pseudo code of the simulated annealing algorithm. Given the former evaluation function f, temperature update schedule and the way to generate new solution, we use it to search for the solution that we needed.

Algorithm 1 Simulated Annealing Algorithm ($\mathbb{P}_{\nvdash}, T_0$, discount, limit)

```
$\mathbb{P} \leftarrow \mathbb{P}_{\nvdash}, T \leftarrow T_0$
$E \leftarrow f(\mathbb{P}), E'' \leftarrow E, \mathbb{P}'' \leftarrow \mathbb{P}$
while not $T = 0$ do
    $\mathbb{P}' \leftarrow \text{generate}(\mathbb{P})$
    $E' \leftarrow f(\mathbb{P}')$
    if $E' > E$ or $e^{(E'-E)/kT} > rand(0,1)$ then
        $\mathbb{P} \leftarrow \mathbb{P}', E \leftarrow E']$
        if $E' > E''$ then
            $\mathbb{P}'' \leftarrow \mathbb{P}, E'' \leftarrow E$
        end if
    end if
    $T \leftarrow schedule(discount, limit, T)$
end while
return $\mathbb{P}''$
```

We use grid search to set the hyper-parameters related to SA. They are:

$$T_0 = 1, discount = 0.95, limit = 0.001, k = 0.8. \tag{35}$$

Then, the evaluation is continuously decreased until converge to final value. The process is shown in Fig. 14.

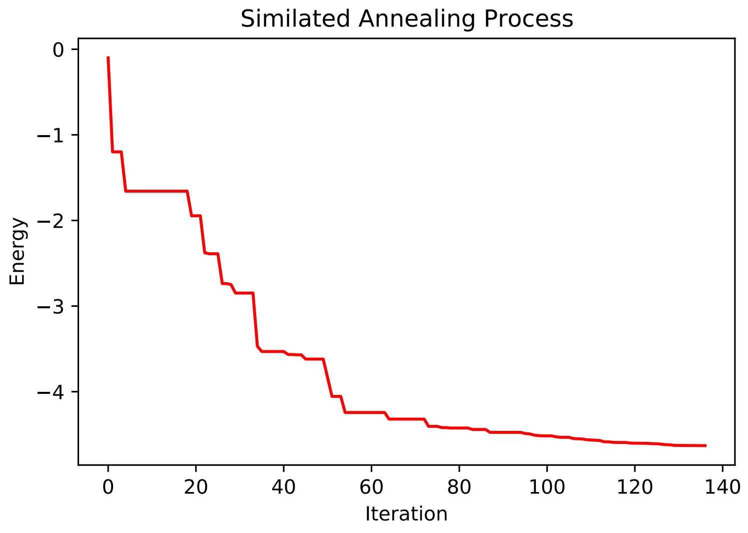

Figure 7: Simulated Annealing Process

2. Hyper-parameter Searching

Here we set different thresholds for the hyper-parameter r to control the Max Draw Down of the entire investment process. After large number of experiments, the relationship between Mean Annualized Returns or Total assets with thresholds is shown in the figure below.

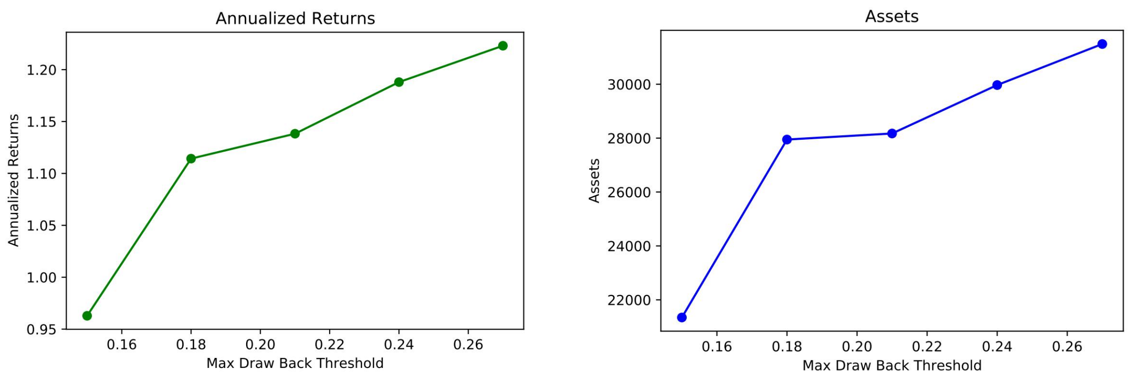

Figure 8: The Relationship between Annualized Returns & Total Assets with Threshold r

The observation shows that, in general, the Max draw Down of investment strategy is inversely proportional to mean Annualized returns or Total assets. However, when Max Draw down is greater than 0.18, the return of the investment strategy is not much improved, but the risk degree rises sharply. Therefore, we choose threshold r as 0.18 to train our model. After many experiments to get a group of best model parameters, the final investment strategy. Its several evaluation indicators are as follows:

Table 3: Evaluation Results of Final Strategy

‘ Evaluation Indicator	Value
Max Draw Down	17.74%
Annualized Returns	132.6%
Total Assets (9/10/21)	$39498.2

5 Our Solutions

5.1 CNN-LSTM Semi-Fuzzy Strategy

For Problem 1, we have get our final strategy in Section 4.3.2, and we put the trained parameters in Appendix A. Given these, now let’s go through our strategy system, following these steps:

Step 0 Once we step into a new trading field, we should allow 45 days for observation and no trading during this period. Keep record the price every day and compute the technical indicators we introduced in section 4.1 and form a data set $\mathbb{D}$

Step 1 For each normal trading day, searching for data set $\mathbb{D}$ to get all the history 30 days features matrix X.

Step 2 Put X into our CNN-LSTM prediction model $\Phi^{1,2}_{B(G)}$. Then we get a bunch of probilities $\mathbb{P}$ indicating increasing or continously increasing. (Whether it will increase tmorrow and the day after tomorrow)

Step 3 Compute the membership degree matrix of $\mathbb{P}$, the fuzzy domain and membership function was defined by equation (12). Put it and whether it's an open day into our semi-fuzzy strategy system trained in Section 4.3.2 and stored in Appendix, then we will get a strategy from the formular $\pi = \alpha\Sigma(C^{\mathbb{A}} \odot \mathbb{A}) + \beta\Sigma(C^{\mathbb{B}} \odot \mathbb{B})$ if it's an open day. $((\hat{\mathbb{A}}, \hat{\mathbb{B}}, \hat{\alpha}, \hat{\beta})$for not open day.)

Step 4 This kind of π takes the form as $[G_{out}, G_{in}, B_{out}, B_{in}]$ in our model, they are or rates between 0 and in, which advise you to do these step by step: (1) sell G_{out} of your Gold and B_{out} of your Bitcoin, (2) Use G_{in} of your cash to buy Gold and use B_{in} of your cash to buy Bitcoin at the same time. In our strategy, there will not be action for which both buy or sell Gold (so do Bitcoin).

We use a picture to show the process more clearly:

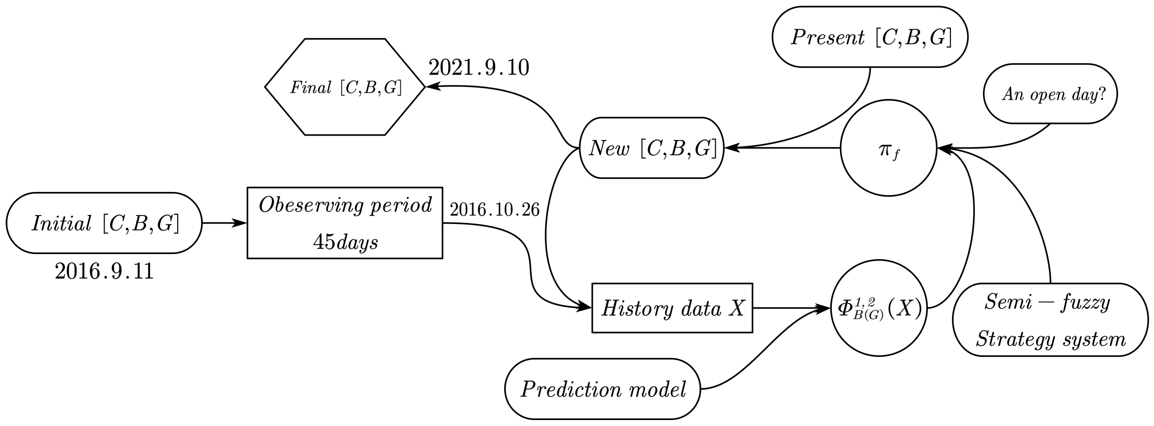

Figure 9: CNN-LSTM Semi-fuzzy Strategy

5.2 Strategy Evaluation

For the results of Section 4.3.2, we have known that our strategy have a high profit, which guarantees 132.6% Annualized Return, and also achieves a balance in safty for only 17.74% Max Drawdown. It seems impossible, but let's take a look at Figure 10,11:

We find that, the changing of our assets always have the same trend of Bitcoin. It's as if we invested in bitcoin ahead of time knowing it was going to soar. In fact, historically, this is the best solution. All in Bitcoin, of course, generates huge profits. With more data, we could learn strategies that are better suited to the situation, but for now, the results are reasonable.

On the other hand, this also shows that our model does learn the optimal strategy for the

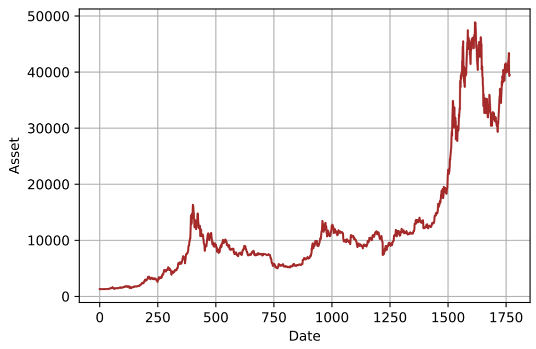

Figure 10: Assets during trading

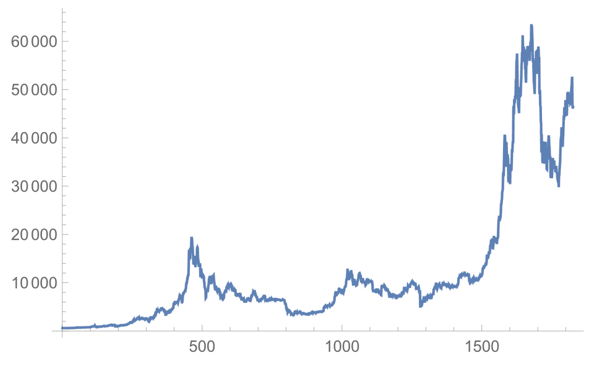

Figure 11: 1: Bitcoin Price

past five years, so we can imagine that if we have more data after 2020, we can naturally form a better solution that's where our model powerful.

Actually, it's an improvable strategy and has much potential. That is because, in this paper, we devided the probility into 5 fuzzy domain. imagine what will happen if we let 5 to be 10? 20? 10000? As long as the prediction model is good enough, we can make the decision infinitely close to the theoretical optimal solution.

Theoretically, we can view the optimized relationship between the predicted probilties $[\Phi_{B,G}^{1,2}]$ and the strategy $[G_{out}, G_{in}, B_{out}, B_{in}]$ as a measurable function $f:\ [0,1]^4 \mapsto [0,1]^4$ (actually we assume it to be measurable. For such a function, we don't think it will have too bad natures).So essentially, this problem is equivalent to approximating a measurable function with a linear function, since the operator we used are all linear. And it will absolutely work, for we can use simple functions to approximate measurable one, and step function is in some way a linear one except for all the edge which has a zero measure. i.e:

$$for\,\forall\varepsilon, \exists simple\ function\ \varphi, ||\varphi - f|| < \frac{\varepsilon}{2}, \varphi = \Sigma\chi_{E_i}, \chi\ characteristic.$$

$$then\ \exists mesh\ n\times n\times n\times n\ of\ [0,1]^4, \frac{1}{n^4} < Min\{m(E_i)\},$$

$$then\ \exists\phi = \Sigma\chi_{F_i}, F_i\ to\ be\ meshed\ regions, then\ \forall F_i, \exists F_i \subseteq E_j, thus\ ||\phi - f|| < \frac{\varepsilon}{2}$$

$$let\ \Omega\ to\ be\ the\ edges\ of\ such\ mesh, linear\ function\ g = \phi\ for\ x \in [0,1]^4 \backslash B_r(\Omega)$$

$$g\ can\ be\ any\ form\ continuously\ linear\ in\ B_r(\Omega), but\ all\ bounded\ by\ 1.$$

$$so\ for\ m(\Omega) = 0, take\ r \to 0, have\ ||g - \phi|| \leqslant m(B_r(\Omega)) \to 0,$$

$$thus\ \exists\ ||g - \phi|| < \frac{\varepsilon}{2}, so\ ||g - f|| < \varepsilon, that\ is\ to\ say\ \exists linear\ g \to\ measurable\ f$$

So now we have proved this imformerly. This small property ensures that, As long as we have enough computation power, we can approach the optimal decision arbitrarily.

5.3 Sensitivity Analysis

5.3.1 Influence of Transaction Cost

To evaluate the impact of transaction cost on our model and results. For each given transaction cost α_{gold} and $\alpha_{bitcoin}$, our strategy for the model is scaled up or down by a factor of k. This is to adjust the overall strategy of the trade. We need to find the optimal k so that the whole strategy can get the maximum returns (annualized returns). Take $\alpha_{gold} = 1.1, \alpha_{bitcoin} = 1.2$ as an example. The relationship between k and annualized returns is shown as follows.

As can be seen from the Fig. 12, when $k \simeq 0.98$, the annualized returns is the highest. There is also an obvious peak. Similarly, we tried other several groups of transaction costs to explore the corresponding value of k and made the following figure.

As can be seen from the left part of Fig. 13, the relationship curve generally decreases with the increase of transaction cost. That is, with the increase of transaction costs, the amount of transactions is relatively reduced. At the same time, the change degree is not more than 5% Thus we can conclude that the strategy is stable.

We then analyzed the final Annualized returns under the searched strategy. From right part of Fig. 13, we can find that the change in the final results was no more than 5%. Thus, it can be

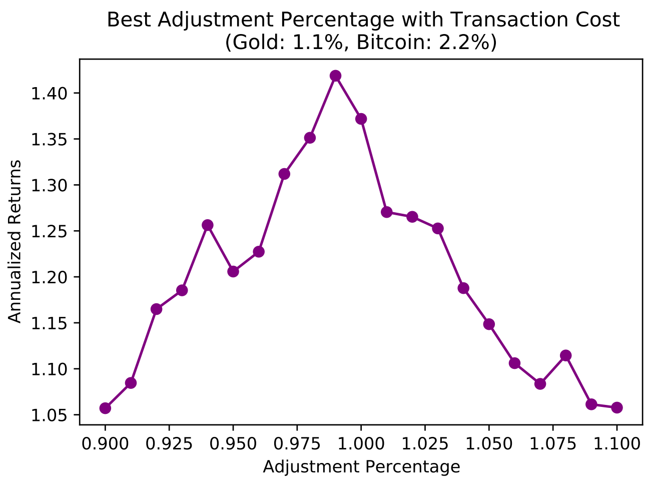

Figure 12: Best Adjustment Percentage with Transaction Cost (Gold: 1.1%, Bitcoin: 2.2%)

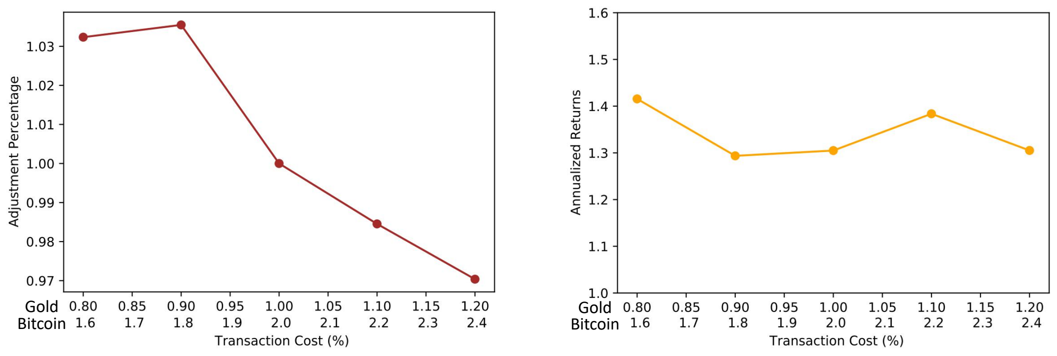

Figure 13: Sensitivity of Strategy and Annualized Returns to Transaction Cost

proved that our method is not sensitive to the change of transaction costs.

5.3.2 Sensitivity of Result to Daily Prices

To assess the sensitivity of our model to slight changes in the market prices of gold and bitcoin, we add different degrees of perturbation to the prices and then use our model to generate new daily trading strategies based on the new market prices. Then we calculate annualized returns and assess their fluctuation. Noise is generated by a Gaussian distribution with a mean of 0, and its variance is adjusted to indicate the difference in the degree of perturbation. The results are shown in the figure below.

From the result, it can be seen that the annualized returns obtained by the strategies generated by our model are less than 5%, even when the level of interference reaches 1%. Therefore, our model has strong anti-perturbation ability to market price fluctuation.

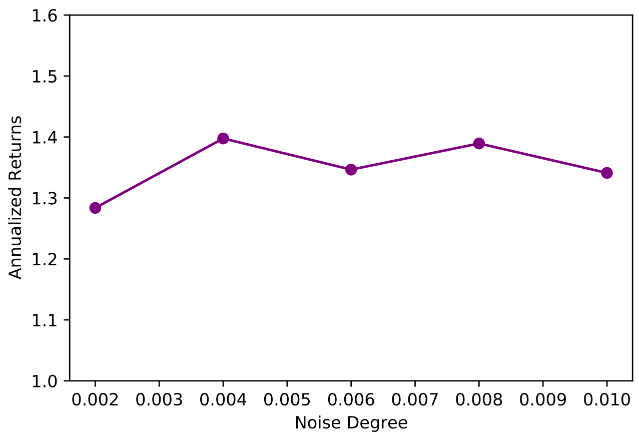

Figure 14: Sensitivity of Annualized Returns to Different Fluctuation

6 Further Discussions

6.1 Strengths

- The prediction model can both extract the spatial information (CNN) and the temporal information (LSTM) of the input data and derive a rather accurate prediction.
- The model has a high rate of return and a balance between return and risk
- The model is insensitive to market fluctuation and has high stability

6.2 Weaknesses

- The prediction model is much too complicate for such little data, that it nearly immediately converge to a local minimum and vibrate around it. Our model need to be pruned to fit such little data.
- The calculation process is complicated and it is difficult for individual users to put into practical application
- Training tends to fall into local optimality

6.3 Further Remarks

- The machine learning part can use more advanced models
- If we have more up-to-date data, we can train a better model which are more instructive
- If we have enough computing power, we can get better strategic systems

Memorandum

To: Any posible trader
From: Team 2217249
Date: Febuary 21st, 2022
Subject: A powerful strategy model focused in Gold and Bitcoin trading

In this memo, we want to introduce you a powerful strategy model focused in gold and bitcoin trading . In fact, according to our result, relying on our model may brought you profit several dozens times more than the investment, and guarantee the safety at the same time. Here we present you the mechanism of our model and the test results.

As we all know, bitcoin is gaining momentum and has become a hot area for quantitative trading. But bitcoin is riskier than traditional gold, making it difficult for human analysts to gauge the size of the transactions. Therefore, we introduce machine learning theory and a new strategy system to guide you to trade more objectively.

First, we conclude that investing in Bitcoin is a good choice in the long run, and our model points this out. In addition to buying a portion of gold in the process to offset risk and expand the fund, our model shows that bitcoin transactions are more important and frequent. For you, just plug the historical data for the last 30 days into the model and immediately get a guidance recommendation. Our results show a 132% Annualize Return on our recommendation and a Max Drawdown rate less than 18%, which is definitely a great result.

In the process of sensitivity analysis, we find that no matter how the transaction cost fluctuates, we can avoid risks and maintain a good return rate as long as we expand or decrease the transaction size year-on-year. This is another powerful point of this model

Next, I'll explain our model in detail and suggest some expectations for future investments.

Firstly, our prediction part will give the prediction of current price trend according to the features generated above. Our prediction model combines the advantage of two popular architecture: CNN and LSTM. Our model is able to extract both spatial and temporal information from input data and gives an accurate prediction. According to our test, our model manages to achieve an astonishing accuracy of 60%, not to mention it is only trained on a dataset that only contains less than 2000 lines of data. Also, our model is high variable and can be trained to predict any specific product, as long as the price data is provided.

Secondly, our strategy model uses the original semi-fuzzy model. By substituting the predicted values obtained in the first part into the model, the strategy can be obtained according to a set of regular matrices. The advantages of this structure are twofold: first, it gives very good results, as we have shown. Secondly, we demonstrate theoretically that the model of this structure can be arbitrary approximated with sufficient computational power.

Based on the above considerations, we give you the following suggestions :1. It is important to pay more attention to the Bitcoin market in any case. 2. Utilize our model in your daily trade, this will bring you more profit. 3. If possible, our model needs more data to learn more up-to-date strategies, and more financial support to purchase computers with high computing power, so that our model can be further optimized.

I will be glad to discuss this recommendation with you during any time you are convenient and follow through on any decisions you make.

References

[1] C. Li, “Stock price forecast and quantitative stock selection research based on cnn-lstm,” 2021.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[3] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, “Handwritten digit recognition with a back-propagation network,” *Advances in neural information processing systems*, vol. 2, 1989.

[4] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm and other neural network architectures,” *Neural networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[6] H. Y. L. xia;Shuming chen;, “Fuzzy trading decision based on apriori algorithm and neural network,” *System Science and Mathematics*, vol. 41, pp. 2868–2891, 2021.

[7] P. J. Van Laarhoven and E. H. Aarts, “Simulated annealing,” in *Simulated annealing: Theory and applications*, pp. 7–15, Springer, 1987.

[8] R. A. Rutenbar, “Simulated annealing algorithms: An overview,” *IEEE Circuits and Devices magazine*, vol. 5, no. 1, pp. 19–26, 1989.

Appendix A: Parameters of Strategy Model

α=0.486,

β=0.514,

$\mathbb{A}$=[[[0.013, 0.033, 0.034, 0.], [0.009, 0.072, 0.148, 0.032], [0.038, 0. , 0.006, 0.009], [0.007, 0.026, 0.003, 0.025], [0. , 0. , 0.022, 0.012]],[[0. , 0.014, 0.052, 0.004], [0.036, 0.01 , 0.008, 0.025], [0.053, 0.034, 0.05 , 0.078], [0.003, 0.017, 0.016, 0.006], [0.055, 0.004, 0.004, 0.015]],[[0.036, 0.067, 0.039, 0.034], [0.021, 0.039, 0. , 0.037], [0.032, 0.046, 0.016, 0.035], [0.074, 0.007, 0.053, 0.012], [0.061, 0.019, 0.089, 0.]],[[0. , 0.009, 0. , 0.006], [0.115, 0.061, 0.077, 0.], [0.052, 0.009, 0.022, 0.071], [0. , 0.035, 0.029, 0.015], [0.039, 0.014, 0.051, 0.023]],[[0.025, 0.079, 0.075, 0.], [0.046, 0.008, 0.03 , 0.048], [0.009, 0.002, 0.059, 0.031], [0.015, 0.135, 0. , 0.021], [0.006, 0.06 , 0.031, 0.]]],

$\mathbb{B}$=[[[0.036, 0.088, 0.088, 0.005], [0.011, 0.097, 0.103, 0.004], [0.006, 0.041, 0.019, 0.05], [0.053, 0.029, 0.031, 0.005], [0.068, 0.012, 0.017, 0.]],[[0.077, 0.024, 0.076, 0.056], [0.066, 0.002, 0.045, 0.], [0.007, 0.01 , 0.006, 0.031], [0.034, 0.008, 0.006, 0.043], [0.009, 0.035, 0.05 , 0.014]],[[0.028, 0.107, 0.013, 0.029], [0.013, 0. , 0.001, 0.029], [0.002, 0. , 0.037, 0.009], [0.011, 0.002, 0. , 0.039], [0. , 0.005, 0.015, 0.026]],[[0.006, 0. , 0.008, 0.016], [0.01 , 0.059, 0.021, 0.062], [0.012, 0.044, 0.026, 0.035], [0.009, 0.022, 0.03 , 0.01], [0.039, 0.007, 0.044, 0.003]],[[0.01 , 0.017, 0.056, 0.043], [0.058, 0. , 0.022, 0.049], [0.023, 0.042, 0.111, 0.027], [0. , 0.003, 0.019, 0.], [0.006, 0.099, 0.044, 0.]]],

$\hat{\alpha}$=0.441,

$\hat{\beta}$=0.559,

$\hat{\mathbb{A}}$=[[[0. , 0. , 0.12092181, 0.01315896], [0. , 0. , 0.01491097, 0.0069585], [0. , 0. , 0.05082998, 0.01826301], [0. , 0. , 0.06848397, 0.01155161], [0. , 0. , 0.09076427, 0.]], [[0. , 0. , 0.01518378, 0.02970495], [0. , 0. , 0.07508329, 0.04375617], [0. , 0. , 0.01500047, 0.01689974], [0. , 0. , 0.04332263, 0.05513588], [0. , 0. , 0.0569539 , 0.00602539]], [[0. , 0. , 0.09503058, 0.], [0. , 0. , 0. , 0.02438407], [0. , 0. , 0. , 0.02848117], [0. , 0. , 0.04548466, 0.08024514], [0. , 0. , 0.05503268, 0.0180284]], [[0. , 0. , 0.03155772, 0.], [0. , 0. , 0.03085961, 0.02366323], [0. , 0. , 0.02638542, 0.05803832], [0. , 0. , 0.0091637 , 0.0090518], [0. , 0. , 0.05853706, 0.00302817]], [[0. , 0. , 0.01314618, 0.00822959], [0. , 0. , 0. , 0.01096658], [0. , 0. , 0.00485794, 0.], [0. , 0. , 0.04595638, 0.02519672], [0. , 0. , 0.06836741, 0.]]],

$\hat{\mathbb{B}}$=[[[0. , 0. , 0.04656756, 0.01982281], [0. , 0. , 0. , 0.01288995], [0. , 0. , 0.00714917, 0.0152795], [0. , 0. , 0.05038157, 0.00897572], [0. , 0. , 0.09922753, 0.00386074]], [[0. , 0. , 0.01171401, 0.01453661], [0. , 0. , 0.02703172, 0.01158302], [0. , 0. , 0.01103658, 0.0241226], [0. , 0. , 0.00937348, 0.00370453], [0. , 0. , 0.02714607, 0.07443981]], [[0. , 0. , 0.04434002, 0.03988098], [0. , 0. , 0.03204451, 0.0390944], [0. , 0. , 0.00617256, 0.08362676], [0. , 0. , 0.04638327, 0.], [0. , 0. , 0.03212002, 0.00133314]], [[0. , 0. , 0.02985292, 0.], [0. , 0. , 0.02877802, 0.03097864], [0. , 0. , 0.00280296, 0.0313284], [0. , 0. , 0.02025571, 0.00991207], [0. , 0. , 0.02872257, 0.02016266]], [[0. , 0. , 0.08464347, 0.01052469], [0. , 0. , 0.02823744, 0.07768397], [0. , 0. , 0.0050779 , 0.01403218], [0. , 0. , 0.0107202 , 0.], [0. , 0. , 0.02868789, 0.]]].

Appendix B: Part of Our Codes

```
# Data Preprocession
import pandas as pd
import numpy as np
pd.set_option('display.max_rows', None)
gold = pd.read_csv('LBMA-GOLD.csv')
bitcoin = pd.read_csv('BCHAIN-MKPRU.csv')
data = bitcoin.copy().merge(right=gold, on='Date', how='outer')
multiplier = (0.5 - data['Gold'].isna()) * 2
data.fillna(method='ffill', inplace=True)
data.fillna(method='bfill', inplace=True)
data['Gold'] *= multiplier
```

```
# Prediction Model
import torch
from torch.nn import Conv2d, LSTM, Sequential, ReLU, Linear, Flatten, Squeeze

class MyNet(torch.nn.Module):
    def __init__(self):
        super().__init__()
        self.net = Sequential(
            Conv2d(1, 16, (3, 3)), Conv2d(16, 64, (3, 3)), Conv2d(64, 1, (3, 3)),
            Squeeze(), LSTM(8, 64, 2, batch_first=True), Flatten(),
            ReLU(), Linear(24 * 64, 64), ReLU(), Linear(64, 1),
        )
    def forward(self, x):
        x = self.net(x)
        return torch.sigmoid(x)
```

```
# Computer Member Degree Matrix
def get_membership_from_prob(prob):
    membership = np.zeros(5)
    if prob < 0.1:
        membership[0] = 1
    elif prob > 0.9:
        membership[4] = 1
    else:
        for i in range(5):
            dist = np.abs(prob - (0.2 * i + 0.1))
            membership[i] = 0 if dist > 2 else round(1 - dist / 0.2, 4)
    return membership

# Compute Evaluation Indicators
def get_asset_from_strategy(strategy_list):
    portfolio = np.array([1000, 0, 0])
    alpha_G, alpha_B = 0.01, 0.02
    annual_asset = [[portfolio, 1000]]
    all_asset = list()
    for i in range(strategy_list.shape[0]):
        Value_G, Value_B = np.abs(data.iloc[i]['Gold']), data.iloc[i]['Bitcoin']
        G_in, G_out, B_in, B_out = strategy_list[i]
        C, G, B = portfolio
        C = C + G * G_out * Value_G * (1 - alpha_G) + B * B_out * Value_B * (1 -
            alpha_B)
        G = G - G * G_out + C * G_in / Value_G
        B = B - B * B_out + C * B_in / Value_B
        C = C - C * G_in - C * B_in
        portfolio = np.array([C, G, B])
        asset = C + G * Value_G + B * Value_B
        if data.iloc[i]['Date'][:4] == '9/10':
            annual_asset.append([portfolio, asset])
        all_asset.append(asset)
    all_asset = np.array(all_asset)
    back_min = np.array([np.min(all_asset[i : ]) for i in
        range(all_asset.shape[0])])
    max_draw_back = np.abs(np.mean((back_min - all_asset) / all_asset))
    return annual_asset, portfolio, asset, max_draw_back
```

```
# Simulated Annealing
def simulated_annealing(A, B, alpha, S1, S2, beta, T, discount, limit):
    E = evaluate(A, B, alpha, S1, S2, beta)
    A_b, B_b, alpha_b, S1_b, S2_b, beta_b = A, B, alpha, S1, S2, beta
    E_best = 0
    while T:
        A_n, B_n, alpha_n, S1_n, S2_n, beta_n = generate_solution(A, B, alpha,
            S1, S2, beta, T)
        E_new = evaluate(A_n, B_n, alpha_n, S1_n, S2_n, beta_n)
        if E_n > E or np.exp((E_n - E) / T) > np.random.rand():
            A, B, alpha, S1, S2, beta = A_n, B_n, alpha_n, S1_n, S2_n, beta_n
            if E_new > E_b:
                A_b, B_b, alpha_b, S1_b, S2_b, beta_b, E_b = A, B, alpha, S1, S2,
                    beta, E_n
        T = schedule(discount, limit, T)
    return A_b, B_b, alpha_b, S1_b, S2_b, beta_b

# Optimization Process
A, B, alpha, S1, S2, beta = correct_constraint(
    np.zeros((5, 5, 4)), np.zeros((5, 5, 4)), 0.5, np.zeros((5, 2)),
        np.zeros((5, 2)), 0.5)
strategy_list = get_strategy_list(C1_list, C2_list, B1_list, B2_list, alpha, A,
    B, S1, S2, beta)
annual_asset, portfolio, asset, max_draw_back =
    get_asset_from_strategy(strategy_list)
T, discount, limit = 1, 0.95, 0.001
A_b, B_b, alpha_b, S1_b, S2_b, beta_b = simulated_annealing(A, B, alpha, S1,
    S2, beta, T, discount, limit)
```