Problem Chosen	2022 MCM/ICM Summary Sheet	Team Control Number
C		2210390

Portfolio Management Based on DDPG Algorithm of Deep Reinforcement Learning

Summary

We propose a model using the deep reinforcement learning DDPG[1] algorithm to exploit the data characteristics of daily prices of volatile assets: gold and bitcoin over the last five years in order to decide whether traders should buy, sell or hold assets in their portfolios. The results show that using an initial $1,000, from September 11, 2016 to September 10, 2021 according to the trading strategy given by the model, our portfolio [C,G,B] was able to reach a value of $143,322.3 at the market price of that day.

We used the ARIMA model to predict the daily prices of the two volatile assets for the next year and applied our trading strategy, which eventually succeeded in maximizing the total return, verifying that our model produced the best strategy.

We also perform a sensitivity analysis on the transaction costs to demonstrate the robustness of our model. The results show that the transaction cost of both assets, being a fixed percentage of the transaction amount, can significantly affect the choice of trading strategy.

key words : deep reinforcement learning(DRL); deep deterministic policy gradient(DDPG); Autoregressive Integrated Moving Average(ARIMA); portfolio management

MEMO

FROM: Team 2210390 , MCM C
To: The Traders
Date: February 18, 2022

Dear Traders:

We would be honored to help you use past daily prices to determine whether you should buy, hold or sell assets in your investment group. We are writing this letter to report our findings.

We analyzed the daily prices of gold and bitcoin over a five-year period and applied deep reinforcement learning techniques to portfolio management using the Deep Deterministic Policy Gradient (DDPG) algorithm in deep reinforcement learning, which spreads risk by limiting the investment weights of individual assets and using a dropout algorithm (Dropout), i.e., randomly dropping nodes when training the model, is used to solve the overfitting problem. The results show that the value of our deep reinforcement learning-based portfolio has increased significantly by 1400% over a five-year period, demonstrating the effectiveness of our approach. Further experiments show that our portfolio performs better when the data used for training is closer to the test data.

In addition, since commissions are not a fixed percentage of the transaction amount in real market trading, we perform a sensitivity analysis of the transaction costs and recommend that you change your trading strategy according to the fluctuation of the costs in order to maximize the total return.

We believe that our model is useful for trading in the future. You are welcome to contact us at any time for further cooperation.

Sincerely yours
MCM C Team 2210390

Contents

1 Introduction

1.1 Background of the problem

Market traders frequently buy and sell unstable assets (e.g. Bitcoin, gold, etc.), constantly reallocating their capital to different financial products with the aim of reducing transaction costs and increasing investment returns, while curbing risk.

However, financial markets are unpredictable, with new policies, changes in the social and natural environment, scientific and technological developments, and even psychological factors of important investment groups causing changes in the market, making it very difficult to accurately predict the rise or fall of market prices.

To obtain stable profits, investors need to accurately grasp the overall laws of the market and develop a scientific and clear quantitative trading strategy to determine the time, object and quantity of each transaction. Traditional portfolio management methods include ”follow the winner”, ”pattern matching” and ”meta-learning”.

In recent years, deep learning has been successfully applied in various financial fields, including quantitative trading. Deep learning can extract imperceptible features from complex financial markets to assist firms and institutions in developing scientific quantitative trading strategies. And automated trading intelligences in reinforcement learning are able to buy, sell or hold assets as actions and seek optimal trading strategies in interaction with the market environment.

1.2 Restatement of the Problem

Between September 11, 2016 and September 10, 2021, we are asked to trade bitcoin and gold with an initial $1000.

- Design a model that gives a daily trading strategy based on the daily gold and bitcoin trading prices on that day.
- Calculate the final value of the investment on September 10, 2021.
- Prove that our model provides the best trading strategy.

- Perform a sensitivity analysis to determine how sensitive the strategy is to the cost of trading.

1.3 Our Works

- First, we apply deep reinforcement learning techniques to portfolio management, using the deep deterministic strategy gradient DDPG algorithm in deep reinforcement learning to build the model.
- Then, we solve the overfitting problem by limiting the investment weights of individual assets to diversify the risk, and by using a discard algorithm, i.e., randomly discarding nodes when training the model.
- Next, we use the ARIMA model to predict daily prices for the next year, verifying that the trading strategy generated by our model is optimal.
- Finally, we analyze the sensitivity of the trading strategy to cost, and adjust the trading strategy appropriately according to the fluctuation of the commission ratio.

2 Assumptions and Notations

2.1 Assumptions

1. We consider only backtest trading, i.e., we only know the current and historical market state at a given moment before making a trading decision, and not the future market condition.
2. The money trading operations we perform as individual traders do not have any impact on the overall market up or down situation.
3. The daily value of gold and bitcoin is constant and the price per trade is the same as the value in the tables LBMA-GOLD.csv and BCHAIN-MKPRU.csv.
4. We can complete any transaction at the moment of market opening, i.e. there is no inability to buy or sell.

5. The cost of a transaction is a fixed percentage of the day's trading volume.
6. We can buy any amount of gold and bitcoin according to the optimal allocation weighting, and the number of purchases can be a non-integer number.

2.2 Notations

Here are all the notations and their meanings in this paper.

Symbol	Meaning
t	Time
a	Action
s	State
y_t	The investment return of each asset of the portfolio expressed as a vector
V_t	represents the closing price of the corresponding asset in the portfolio at time t
p_t	Total value of the portfolio at time t minus transaction costs
r_t	The return of the portfolio at time t
w_t	denotes the weight of each asset of the portfolio at the corresponding moment
π	Strategy
q_t	Network Parameters

3 Model Construction

3.1 Deep learning

Deep learning is a general framework for learning representations using a multilayer nonlinear transformation processing network structure. Thanks to the improvement of computer performance, deep learning has greatly improved the effectiveness of artificial neural network

methods and is an important method in the field of artificial intelligence in recent years.

Traditional artificial neural networks are not suitable for deep networks because they are prone to local optimal solutions or overfitting problems as the number of layers of neural networks increases.This makes learning time-consuming and labor-intensive.

The concept of deep learning was proposed by Hinton et al. at the University of Toronto, Canada, in 2006, based on the unsupervised layer-by-layer pre-training and supervised tuning algorithms of the Deep Belief Network (DBN), which can automatically learn data features and effectively optimize the learning results, solving the problem that the deep neural network structure is prone to local optimal solutions. The problem of local optimal solution of deep neural network structure and the problem of manual design of data features were solved. Later, Hinton proposed the Dropout method to solve the overfitting problem of neural networks. After that, deep learning developed rapidly and was widely used in speech, image, medical and financial fields. The common network structures of deep learning include deep neural network DNN (Deep Neural Networks), deep belief network DBN, convolutional neural network CNN, recurrent neural network RNN, LSTM, etc.

3.2 Reinforcement learning

Reinforcement learning, also known as augmented learning or reactive learning. The main idea of reinforcement learning is that an intelligent body interacts with its environment through trial and error, and uses evaluative feedback signals to optimize its decisions. When an action of an intelligent body leads to a positive reward or reward from the environment, which is the reinforcement signal, the tendency of the intelligent body to produce this action in the future will be strengthened.

Conversely, the tendency to produce this action will be weakened. There are four main elements of reinforcement learning, namely, strategy, reward, action, and environment. The goal of reinforcement learning is to learn a strategy so that the action chosen by the intelligence receives the maximum reward from the environment. The payoff can be calculated as a function, also known as the payoff function. In order to measure the long-term effect of reinforcement learning, a value function (the

In order to measure the long-term effects of reinforcement learning,

the value function is often used instead of the payoff function to measure not only the immediate payoff of an action, but also the cumulative payoff of a series of possible subsequent states from that state onward.

Classical reinforcement learning methods are often unable to solve the problem of high spatial dimensionality of states and actions, and an effective solution is to use function approximation, which represents the value function or strategy as a function. Commonly used approximation functions include linear functions, kernel functions, and neural networks. The most successful function approximation method in recent years is to use deep neural network as a nonlinear approximation function for reinforcement learning, which combines deep learning and reinforcement learning, that is, deep reinforcement learning. Deep reinforcement learning was first proposed by DeepMind team in 2015, and then developed and applied to AlphaGo, which defeated the human Go champion, and AlphaGo Zero, which was even stronger. The deep deterministic policy gradient method, DDPG algorithm, proposed by DeepMind team in 2016, solves the problem of reinforcement learning in continuous action space. In DDPG algorithm, the strategies are parametrized and the parameters of the deep neural network are directly optimized by the strategy gradient method.

3.3 Deep deterministic policy gradient

The full name of DDPG[2] is Deep Deterministic Policy Gradient, which means Deep Deterministic Policy Gradient. From the name of DDPG, it is composed of D(Deep) + D(Deterministic) + PG(Policy Gradient). Then its features can also be broken down in this way.

- The motivation of DDPG is actually to allow DQN to be extended to a continuous action space.
- DDPG borrows two techniques from DQN: experience replay and fixed Q-network.
- DDPG uses a policy network to output deterministic actions directly.
- DDPG uses the Actor-Critic architecture.

DDPG is an extended version of DQN that can be extended to the control space of continuous actions. Therefore it adds a layer of policy

network on top of DQN for outputting action values.DDPG needs to learn the Q network while learning the policy network.

DDPG has 4 networks, Actor current network, Actor target network, Critic current network, and Critic target network.

- Actor current network: responsible for iterative update of policy network parameters θ, responsible for selecting current action A based on current state S, used to interact with environment to generate S′,R.
- Actor target network: responsible for selecting the optimal next A′ based on the next state S′ sampled in the experience replay pool. The network parameter θ′ is periodically copied from θ.
- Critic Current Network: Responsible for iterative update of the value network parameter w. Responsible for computing the current Q value Q(S,A,w) responsible for computing the current Q value. Target Q-value yi=R+γQ′(S′,A′,w′)
- Critic target network: responsible for computing the Q′(S′,A′,w′) part of the target Q-value. The network parameters w′ are periodically copied from w.

The replication of DDPG from the current network to the target network is different from DQN, which directly replicates the parameters of the current Q network to the target Q network, i.e., w′=w. DDPG does not use this hard update here, but uses a soft update, i.e., the parameters are updated only a little bit at a time, i.e.

$$\begin{aligned} w\prime &\leftarrow \tau w + (1-\tau) w\prime \\ \theta\prime &\leftarrow \tau \theta + (1-\tau)\theta\prime \end{aligned} \tag{1}$$

4 Model Simulation and Analysis

4.1 Model Representation

4.1.1 Asset composition of the portfolio

The portfolio in this paper consists of 2+1 financial assets, including 2 risky assets and one currency (USD). The total weight of the portfolio

is set to 1. If only risky assets are included in the portfolio, investors will have to hold all risky assets when the market is underperforming, which will reduce the return and increase the risk. By including currency as an asset in the portfolio, it is possible to hold only currency when the market is underperforming, which is equivalent to not holding risky assets.

4.1.2 Portfolio asset weights and returns

Let y_t be the investment return of each asset of the portfolio on the interval [t - 1, t] expressed as a vector, which is referred to as the return vector as follows:

$$y_t = \left[1, \frac{V_{g,t,\text{ close}}}{V_{g,t-1,\text{ close}}}, \frac{V_{b,t,\text{ close}}}{V_{b,t-1,\text{ close}}}\right] \tag{2}$$

The first component of the vector represents the yield of the currency, which is 1 because there is no additional gain from just holding the currency, and the value of the currency remains constant from start to finish. The vector of $V_{g,t,\text{ close}}$ is denotes the closing price of gold in the portfolio at time t; $V_{b,t,\text{ close}}$ is denotes the closing price of Bitcoin in the portfolio at time t.Where i = 1, 2, ..., N, thus $\frac{V_{i,t,\text{ close}}}{V_{i,t-1,\text{ close}}}$ denoting the return on investment of i assets over the interval [t - 1, t].

Let p_t be the total value of the portfolio at time t after subtracting transaction costs, p_{t-1} be the total value of the portfolio at time t - 1 after subtracting transaction costs, p_0 be the initial value of the portfolio, and p'_t be the total value of the portfolio at time t before subtracting transaction costs. Let r_t be the return of the portfolio at time t, i.e.

$$r_t = \frac{p_t}{p_{t-1}} \tag{3}$$

Let R_t be the cumulative total return of the portfolio at time t, i.e.:

$$\mathrm{R_t} = \frac{\mathrm{p_t}}{\mathrm{p_0}} \tag{4}$$

The weight of each asset in the portfolio at [t - 1, t] is denoted by w_t and is referred to as the weight vector, as follows:

$$W_1 = \left[\ W_{1t},\ \ W_{gt},\ \ W_{bt}\ \right] \tag{5}$$

Thus Ease of Knowing:

$$\frac{p'_t}{p_{t-1}} = w_t \cdot y_t \tag{6}$$

4.2 DDPG algorithm structure

The basic elements of reinforcement learning include an intelligent agent, an environment, a state s, a policy π, an action a, and a reward r. The agent perceives the current state s_t from the environment, selects an action a_t according to the policy π, acts on the environment, and obtains a reward r_t, which is converted to the next state s_{t+1}. Since the reward r_t only reflects the reward of the current action and state, but not the impact on the future reward, a value function v is also introduced. The value function v is introduced, which contains the current returns and the estimated discounted (denoted by γ) future returns. Deep reinforcement learning methods use deep neural networks to learn strategies in the continuous action space, which are parametrically represented, and the output is the action.

The DDPG algorithm[3] adopts the reinforcement learning Actor-Critic architecture, which consists of 4 neural networks: 2 structurally identical Actor strategy networks, Actor-online network and Actor-arget network respectively; 2 structurally identical Critic evaluation networks, Critic-online network and Critic -target networks. The Actor-target and Critic-target networks are mainly used to generate the training data set, while the Actor-online and Critic-online networks are mainly used to train and optimize the network parameters. Compared with other neural networks, LSTM networks are more suitable for time series data prediction, so LSTM is used in this paper.

The two Actor strategy networks are composed of a long and short term memory network layer LSTM, a fully connected network layer (FC) and a softmax layer, where the activation function of the fully connected network layer is a linear correction function (ReLU). The network input is the state, i.e., the asset price s_t, and the output is the buy and sell action a_t.

Both Critic evaluation networks are composed of an LSTM network layer, an FC fully connected network layer (input is asset price s_t) and an FC fully connected network layer (input is action a_t), and the last two FCs are summed to another FC fully connected network layer. The network inputs are the asset price s_t and the buy and sell action a_t, while the output is the estimated future discounted return q_{t+1}.

The structure of the DDPG algorithm is shown in Figure 1.

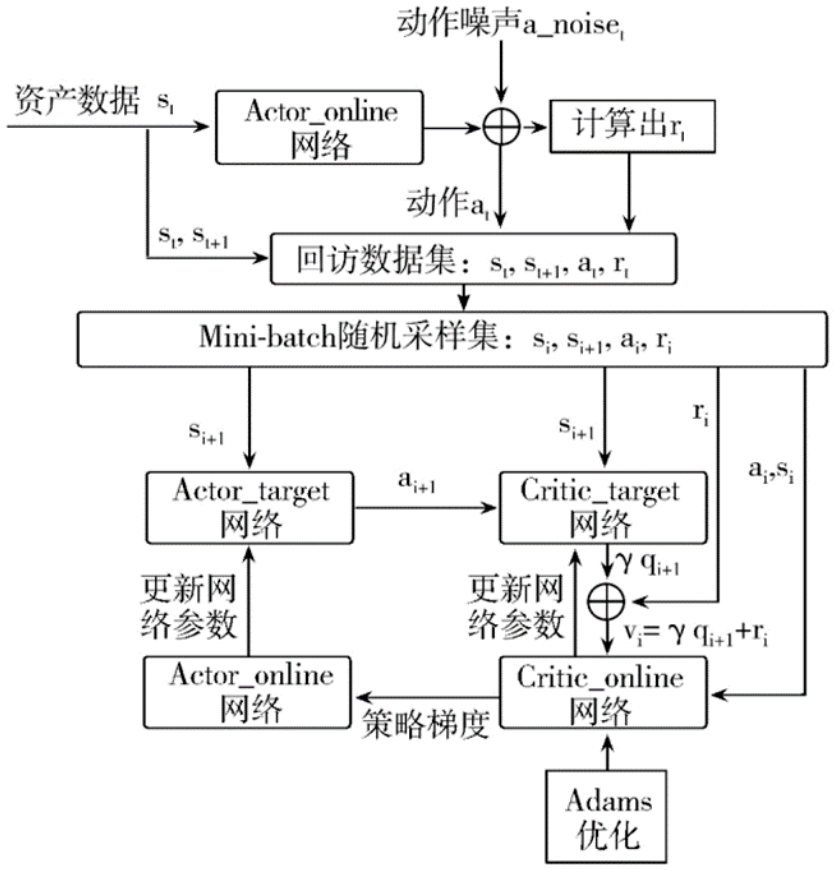

Figure 1: DDPG architecture

The DDPG algorithm is described in the following steps:

1. The Actor-online network generates the playback dataset: the initial action is output according to the asset price s_t , and the action a_t is output with the random noise a-noiset, and the immediate return rt is calculated, i.e., the current return; the transition to the next state, i.e., the asset price s_{t+1} at the next moment.The training data set, also called the playback data set, consists of (s_t , a_t , r_t , s_{t+1}).

2. Mini-batch sampling is performed on the playback data set, i.e., the same number of random and consecutive samples are sampled each time as the training data for the network. The set of data in a single Mini-batch is represented by (s_i , a_i , r_i , s_{i+1}).

3. Estimate the value function vi using the Mini-batch sampled data.

a_{i+1} is calculated by the Actor-target network, q_{i+1} is calculated by the Critic-target network, and finally the value function $v_i = r_i + q_{i+1}$ is calculated.

4. The Critic-online network is trained with the Mini-batch sampled data and the calculated value function v_i , and optimized by Adams (AdaptiveMoment Estimation) method, while the Critic-online network optimizes the Actor-online network by the policy gradient.

5. Finally, the parameters of the Actor-target network and the Critic-target network are updated by the Critic-online network and the Actor-online network, respectively.

4.3 Problems encountered and solutions

The computer operating system is Windows 11, the running environment is Tensorflow 1.4, and the programming language Python 3.7 is used. In the actual training of the network, we mainly found the problems of overfitting and unreasonable distribution of asset weights, and we analyze the causes and solutions below.

1. In the process of algorithm implementation, the neural network is prone to good results in training and poor performance in testing, which means that the network is too dependent on the training data set, and therefore cannot effectively handle new data when they appear, that is, the phenomenon of overfitting. In order to solve this problem, the following measures are taken: add Dropout option to all LSTM networks; appropriately reduce the number of samples of the data set (Max Step); change the window length of the input asset data (Window Length), "Window Length" is set in the trading process The "Window Length" is the number of days of historical data set in the trading process, which is used to calculate the current optimal weighting.

2. Since reinforcement learning is based on a Markovian decision process, i.e., future returns are considered to be independent of past states, but only related to current and future states. Since the risk (i.e., variance) of a portfolio needs to be calculated from historical data, minimizing the risk of a portfolio as an optimization objective will make the future return related to the past state. Therefore, the

approach used in this paper takes the maximization of the value function v (i.e., current return plus future discounted return) as the only optimization objective, and thus cannot simultaneously minimize risk. However, the original weights can be further processed in order to properly diversify the asset weights in the portfolio to optimize the return and reduce the risk without affecting the performance of the portfolio.

5 Strengths and Weaknesses

Based on the modeling process, we make some comments on our model as listed below.

5.1 Strengths

- a final total return on investment of 14 times the initial capital.
- that the reinforcement learning algorithm enables the trading strategy to adapt well to new environments.
- deep neural networks predict the rise and fall of financial asset prices in the short term with a high accuracy rate.

5.2 Weaknesses

- less data, prone to overfitting problems, resulting in too much concentration of asset weights.
- failed to consider the impact of its own transactions on market prices.
- only two assets have been experimented, for other markets need to be further explored.

6 Sensitivity Analysis

The ARIMA model can be expressed by the following equation

$$\left(1-\sum_{i=1}^{p}\varphi_i L^i\right)(1-L)^d X_t = \delta + \left(1+\sum_{i=1}^{q}\theta_i L^i\right)\varepsilon_t \tag{7}$$

First, we use the model and historical daily prices to forecast volatile assets with a gradient of transaction cost ratios against six groups set at 0.5

The results show that trading strategies will tend to abstain from buying when the cost is too high until the increase in return is greater than the increase in cost.

Figure 2: Best Difference Sequence Chart

We then compared the results with policy gradient. The model with the policy gra□dient algorithm obtained 12,435 RMB at the end of the test, with an annual return of 17.53

Analyzing the results, we can see that our model has the highest annual return.

7 Conclusion

Since there is no application of deep reinforcement learning technology in portfolio management, this paper improves the DDPG algorithm of deep reinforcement learning for the first time to enable its application in portfolio management by limiting the investment weights to diversify the risk and using the discard algorithm to solve the overfitting problem. In this paper, two unstable assets are used as risky assets in the portfolio for experiments. The experimental results show that the performance of

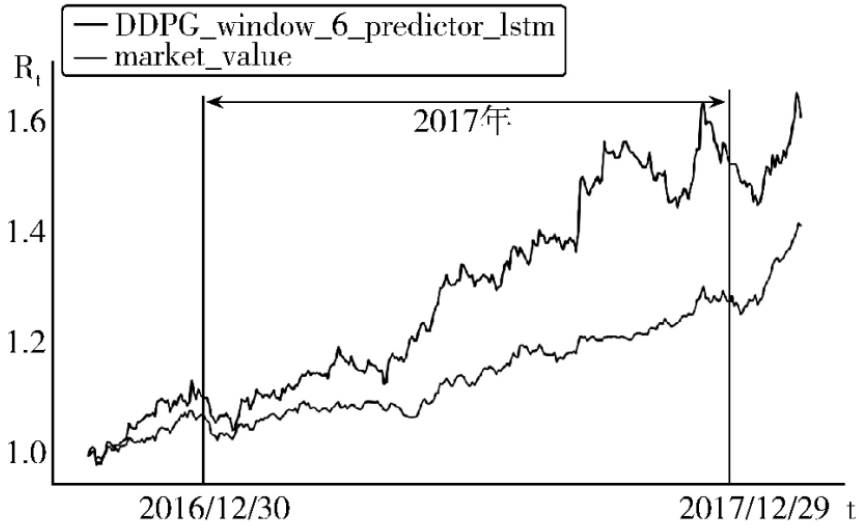

Figure 3: Comparison of our portfolio with the control group

the portfolio constructed by this method is much better than that of other control groups, which demonstrates the effectiveness of this method.

However, some assumptions are made for the sake of simplification, such as the assumption that the trading capital is small enough not to affect the behavior of other investors in the market and the price of financial assets, but when the users of this method are institutional investors or individual investors with large capital, they need to consider the impact of their own trading on the market price, so it is necessary to conduct more in-depth research.

In addition, the algorithm can be applied to portfolios to maximize rewards while maximizing entropy, ensuring stability and explorability. The model proposed in this paper takes raw market data and historical prices as input and directly outputs the weight vector of financial assets in the portfolio. The method proposed in this paper, does not rely on any financial theory and is therefore highly scalable. After analysis, we have achieved satisfactory results in terms of returns, annual returns.

The limitation of the work in this paper is that it is not possible to use historical data to fully simulate real online transactions. Future improvements can be made in terms of how to selectively and purposefully pick quality investments and add financial market information.

References

[1] 万里鹏, 兰旭光, 张翰博, and 郑南宁, "深度强化学习理论及其应用综述," 模式识别与人工智能, no. 1, p. 15, 2019.

[2] G. Huang, X. Zhou, and Q. Song, "Deep reinforcement learning for portfolio management," *Papers*, 2020.

[3] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," *Papers*, 2017.

Appendices

Here is Code we used in our model, which python is the main development language.

Appendices A: An automated trading robot training environment built on deep reinforcement learning

```
from gym import spaces
import pandas as pd
import numpy as np
import math
MAX_GOLD_NUM=1
MIN_GOLD_PRICE=1125.7
MAX_GOLD_PRICE=2061.75
MAX_BIT_NUM=2
MIN_BIT_PRICE=594.08
MAX_BIT_PRICE=63554.44
MAX_DAY=1825
INITIAL_WORTH=1000

def read_data(filepath):
    data = pd.read_csv(filepath)
    data['Date'] = pd.to_datetime(data['Date'])
    return data


class my_env(object):
    action_space = spaces.Box(low=np.array([-1,-1]),
       high=np.array([1,1]), dtype=np.float32)
    # 动作空间：是否交易黄金，交易多少，是否交易比特
       币，交易多少
    observation_space = spaces.Box(low=np.array([0,
       0, MIN_GOLD_PRICE, 0, MIN_BIT_PRICE]), high=np
```

```
    .array([1000, MAX_GOLD_NUM, MAX_GOLD_PRICE,
    MAX_BIT_NUM, MAX_BIT_PRICE]), dtype=np.float32
    )
# 环境状态空间：当前的美金，黄金的持有数和当日价
    格，比特币的持有数和当日价格
def __init__(self):
    self.gold_data = read_data('LBMA-GOLD.csv')
    self.bit_data = read_data('BCHAIN-MKPRU.csv')
    self.cash = 1000
    self.gold = 0
    self.bit = 0
    self.day_bit = 0
    self.day_gold = 0
    self.gold_prize = 0
    self.bit_prize = 0
    self.last_worth=1000

def seed(self, seed):
    np.random.seed(seed)

def reset(self):
    self.gold = 0
    self.bit = 0
    self.gold_prize = self.gold_data['USD (PM)'
      ][0]
    self.bit_prize = self.bit_data['Value'][0]
    self.cash = 1000
    self.day_bit = 0
    self.day_gold = 0
    self.worth = 1000
    return np.array([self.cash,self.gold,self.
      gold_prize,self.bit,self.bit_prize])

def take_action(self,action):
    #当天黄金和比特币的价格
    self.bit_prize = self.bit_data['Value'][self.
      day_bit]
    self.gold_prize = self.gold_data['USD (PM)'][
```

```
            self.day_gold]
        #交易动作和交易量
        if self.gold_data['Date'][self.day_gold]!=
            self.bit_data['Date'][self.day_bit]:#周末
            不进行黄金交易
             action_gold = 0
        else:
             action_gold = action[0]
        action_bit = action[1]

        #计算资产
        if action_gold>0: #买入黄金
             self.cash -= float(math.tanh(action_gold)
                *self.cash*(1+0.01) )#手头美金=原有美
                金-买黄金的钱-佣金
             self.gold += float(math.tanh(action_gold)
                *self.cash/self.gold_prize) #手头黄金=
                原有黄金+新买黄金
        if action_gold< 0: #卖出黄金
             self.cash -= float(math.tanh(action_gold)
                *self.gold*self.gold_prize*(1+0.01))#
                手头美金=原有美金+卖黄金的钱-佣金
             self.gold += float(math.tanh(action_gold)
                *self.gold )#手头黄金=原有黄金-卖出黄
                金
        if action_bit > 0:
             self.cash -= float(math.tanh(action_bit)*
                self.cash*(1+0.02))
             self.bit += float(math.tanh(action_bit)*
                self.cash/self.bit_prize)
        if action_bit < 0:
             self.cash -= float(math.tanh(action_bit)*
                self.bit*self.bit_prize*(1+0.02))
             self.bit += float(math.tanh(action_bit)*
                self.bit)

    def step(self, action):
        # 当天黄金和比特币的价格
```

```
self.bit_prize = self.bit_data['Value'][self.
   day_bit]
self.gold_prize = self.gold_data['USD (PM)'][
   self.day_gold]
#在环境内执行动作
self.take_action(action)

done = False #done判断是否终止

#在终止日期结束
if self.day_bit>MAX_DAY:
    done = True

# 计算资产值。
self.worth = self.cash + self.gold * self.
   gold_prize + self.bit * self.bit_prize
#计算收益和收益比(加强奖励和惩罚)
profit=self.worth-self.last_worth
if profit>= 0:
    reward = profit*10
else:
    reward = -1000
if self.worth <= 0:#当前总价值<0, 停止)
    done = True
obs = np.array([self.cash,self.gold,self.
   gold_prize,self.bit,self.bit_prize])
#更新last worth
self.last_worth=self.worth
#更新日期
if self.gold_data['Date'][self.day_gold]!=
   self.bit_data['Date'][self.day_bit]: #黄金
   工作日开
    self.day_bit +=1
else:
    self.day_gold += 1
    self.day_bit += 1
return obs,reward,done,{}
```

```
    def render(self):
        self.bit_prize = self.bit_data['Value'][self.
           day_bit]
        self.gold_prize = self.gold_data['USD (PM)'][
           self.day_gold]
        # 打印环境信息
        profit = self.worth - INITIAL_WORTH
        print('-'*30)
        print("\ngold price:",self.gold_prize)
        print("\nbit price:", self.bit_prize)
        print("\ncash:",self.cash)
        print("\ngold",self.gold)
        print("\nbitcoin",self.bit)
        print("\nprofit:",profit)
```