1 Introduction

Algorithmic trading in quantitative finance consists of various programmed methods to manage stock exchanging behaviors in a dynamic marketing environment. The design of an algorithmic intelligent agent is expected to perform a series of trading task including forecasting the future tendency, automated trading, generating portfolio strategy, assessing the risk, etc., to optimize the return of investing strategies by analyzing the volatile stock price data. The most common behaviors in stock trading are the bids and asks, which means the trader gains profits through buying in assets or the shares of a company when the price is low and selling them out when the price goes higher. This process is highly dynamic and relevant to time-dependent arbitrage when the trade is conducted among multiple units.

The Deep Reinforcement Learning (DRL) methods have been proved to be successful protocols in dealing with stochastic optimization problems by maximizing the reward function through minimizations the loss function and backpropagating to update the actions in a time sequential manner. Deep Neural Networks (DNN) are applied to approximate the value function (value-based iteration, VI) or the policy functions (policy-based iteration, PI), which are high-order nonlinear functions defined with hyper-parameters in the iterative steps. Popular methods of the former type include Q-leaning, DQN, DRQN, while the latter includes DDPG, PPO, A2C, SAC, etc.

In this task, we try to combine the classic DQN methods with the martingale model originally proposed by F. Black & M. Scholes (1973) as a mixed approach to decide a portfolio strategy which make sure that the expected value of total reward is non-negative. The program is generally based on FinRl-Meta benchmark(https://finrl.readthedocs.io/en/latest/tutorial/Guide.html) and Stable-Baseline 3(https://stable-baselines3.readthedocs.io/en/master/).

Overview of Our Model

Generally, the decision process includes the following steps:

- Train an agent with the classic DQN method, gain the Q-function with initialized stock market data.
- Under the Black-Scholes's assumption, actually we assume that the stock price of an asset are the linear combination of an deterministic process and a Wiener process, and that gives an matrix expression of the Hamilton-Jacobi-Bellman equation, which is the Riccati equation with the loss function a quadratic form.
- We solve the Riccati equation with the Python control system library and that gives a lower bound in the Bellman Optimality Principle with an optimal strategy $\hat{u}(x, t)$and we leave rest of the action to learn to DQN agent.
- Set the decision rule: for every step, by calculating the fair optimal price with the Black-Scholes formula, and observe the predictive value from the DQN agent, if the latter price (predictive price) is higher than the former (fair optimal) price, do a buy action and update the strategy; if the latter price is lower than the former price, do a sell action and update the strategy.
- Call the trade function and update the DQN network.

3 Algorithm Design and Implementation

3.1 Reference Model and the Black-Scholes Formula

The Black-Scholes model start with the basic assumption that:

$$dB_t = r_t B_t dt$$
$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

and we have the differential of the wealth of a shareholder with Ito's lemma:

$$dV(s,t) = \left(\mu S_t \frac{\partial V}{\partial S_t} + \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_t{}^2 \frac{\partial^2 V}{\partial S_t{}^2}\right) dt + \sigma S_t \frac{\partial V}{\partial S_t} dW_t$$

with the value function:

$$dX_t = X_t\left(w_t{}^B r_t + w_t{}^S \mu\right) dt - c_t dt + w_t{}^S \sigma X_t dW_t = \mu\left(t, X_t, u_t\right) dt + \sigma\left(t, X_t, u_t\right) dW_t$$

we see that actually the total revenue of a stock asset can be expressed in linear combination of a predictable process of $\mu\left(t, X_t, u_t\right)$ and a Wiener process. This satisfies the linear model condition and can be expressed in matrix form, which forms a Riccati equation in 3.2. (https://math.stackexchange.com/questions/3776909/optimality-hamilton-jacobi-bellman-hjb-versus-riccati#:~:text=Riccati%20Equations%20can%20be%20derived,and%20the%20cost%20is%20quadratic.&text=with%20the%20terminal%20condition%20V,xTQfx.)

3.2 Hamilton-Jacobi-Bellman (HJB) Equation and Optimal Value Approximation

The objective function in HJB equation is defined as:

$$sup\widehat{J_0} = J_0\left(u_t(x,t)\right) = E\left[\int_o^T F\left(t, X_t{}^{u_t}, u_t\right) dt + \Phi\left(X_T{}^{u_t}\right)\right]$$

and we have the inequality holds for our reference model solved with the Ricatti equation solver:

$$E_{t,x}\left[\int_t^T F\left(s_t, X_t{}^{u_0}, u_s{}^0\right) ds + \Phi\left(X_T{}^{u^0}\right)\right] \geq E_{t,x}\left[\int_t^T F\left(s_t, X_t{}^{\hat{u}}, \widehat{u^s}\right) ds + \Phi\left(X_T{}^{\hat{u}}\right)\right]$$

this guarantees that when we optimize our DQN model with the reference model, the total value of our trades are always greater than that when we take $\hat{u}(x,t)$ as we solved in the Ricatti equation which is a linear model iteration of all sequential steps.

3.3 Deep Q-learning (DQN) with Implementation in FinRL-Meta and Stable-Baseline3

Apply the classic DQN algorithm and we want to optimize the Q-function with the strategy function replaced by our reference model:

$$Q(s,a) = E_{s' \sim \varepsilon}\left[r_t + \gamma_t Q(s',a') \mid s, a, u_t(x,t)\right]$$

the rest of the steps are the same as we do a normal optimization to Q* function in the original paper source. (https://arxiv.org/abs/1312.5602)