Deep Deterministic Policy Gradient in Stock Hedging Problem under Geometric Brownian Motion

Jin Tian & Zheng Zihao

2022 April

Abstract

In this paper, we used Deep Deterministic Policy Gradient (DDPG) method to solve the optimal dynamic hedging problem for derivatives when there are transaction costs. We used stock price as the continuous states and assume that the asset price follows a geometric Brownian motion. The paper illustrates the approach by showing the difference between using delta hedging and optimal hedging for a short position in a call option when the objective is to minimize a function equal to mean-variance return with transaction costs. Our results show that the methods we use can reduce our costs and take into account more targets.

1 Introduction

Hedging is an important problem for traders. Traders typically try to maintain delta-neutral (or close to delta-neutral) positions, because the delta of an option changes during its life, the trader's position must be rebalanced periodically. In practice, transaction costs and other frictions (which we will collectively refer to as "trading costs") mean that the theoretically optimal strategy must be modified. This paper examines how reinforcement learning approach can be used to take trading costs into account in hedging decisions.

In a previous report, we have implemented the Actor-Critic algorithm to optimize a quadratic utility function with transaction costs to achieve optimal hedging and found that hedging transactions using this algorithm can effectively reduce costs. In this paper, we will continue to optimize this objective by introducing the Deep Deterministic Policy Gradient (Deep DPG) algorithm in hedging problems.We first illustrate the deep hedging approach by assuming that the underlying asset follows Geometric Brownian Motion. In this case, our reinforcement learning approach outperforms delta hedging when there are trading costs.

Deep DPG suggested by Lillicrap et al. (2016) combines the ideas of DPG and Deep Q-learning. Using ANNs as function approximators for both the policy and action-value functions, deep DPG follows the basic algorithm outlined above and addresses the challenges of training ANNs as in Deep Q-learning.

2 Models

In the first place, we introduce geometric Brownian motion of the stock price. This model assumes that stock prices obey the following movements

$$dS = \mu S dt + \sigma S dz \tag{1}$$

where μ and σ are the stock's mean return and volatility (assumed constant), and dz is a Wiener process. The work of Black and Scholes (1973) and Merton (1973) show that the price of a European call option with maturity T and strike price K is

$$S_0 e^{-qT} N\left(d_1\right) - K e^{-rT} N\left(d_2\right) \tag{2}$$

where r is the risk-free rate, q is the dividend yield (both assumed constant), S_0 is the initial value of S and

$$d_1 = \frac{\ln\left(S_0/K\right) + \left(r - q + \sigma^2/2\right)T}{\sigma\sqrt{T}} \tag{3}$$

$$d_2 = d_1 - \sigma\sqrt{T} \tag{4}$$

We choose to solve the reinforcement learning problem using the deep DPG method as the method allows the hedging position to be continuous. The algorithm in geometric Brownian motion depends on (1) interest rate: r (2) volatility: σ.

We use as an example the situation where a trader is hedging a short position in a call option. We assume that the trader rebalances her position at time intervals of Δt and is subject to trading costs. The life of the option is $n\Delta t$. The cost of a trade in the underlying asset in our formulation is proportional to the value of what is being bought or sold, but the analysis can easily be adjusted to accommodate other assumptions. The state at time $i\Delta t$ is defined by three parameters:

1. The holding of the asset during the previous time period; i.e., from time $(i-1)\Delta t$ to time $i\Delta t$
2. The asset price at time $i\Delta t$
3. The time to maturity

The action at time $i\Delta t$ is the amount of the asset to be held for the next period; i.e., from time $i\Delta t$ to time $(i+1)\Delta t$.

Our rewards (negative costs) are given in the form of accounting P&L formulation, which is

$$\delta w_{t+1} = V_{i+1} - V_i + H_i\left(S_{i+1} - S_i\right) \tag{5}$$

for $0 \leqslant i < n$ where S_i is the asset price at the beginning of period i, H_i is the holding between time $i\Delta t$ and $(i+1)\Delta t$, κ is the trading cost as a proportion of the value of what is bought or sold, and V_i is the value of the derivative position at the beginning of period i. (V_i is negative in the case of a short call option position.) In addition, there is an initial reward associated with setting up the hedge equal to $-\kappa\left|S_0 H_0\right|$ and a final reward associated with liquidating the hedge at the end equal to $-\kappa\left|S_0 H_0\right|$).

In the problem we are considering, it is natural to work with costs (negative rewards). This is what we will do from now on. We assume agents have quadratic utility, which implies that their optimal portfolios are given as solutions of the mean–variance optimization problem. In this case, we define the one-period reward function as

$$R_t := \delta w_t - \frac{\kappa}{2}\left(\delta w_t\right)^2 - c\left|S_t\left(H_t - H_{t-1}\right)\right| \tag{6}$$

where κ denoting an agent's risk aversion, δw_t is the individual wealth increments in period t, S_t is the asset price at the beginning of period t, H_t is the holding between time $t-1$ and t and tc is the transaction cost ratio.

Deep Deterministic Policy Gradient (DDPG) is a model-free off-policy algorithm for learning continous actions. It combines ideas from DPG (Deterministic Policy Gradient) and DQN (Deep Q-Network). It uses Experience Replay and slow-learning target networks from DQN, and it is based on DPG, which can operate over continuous action spaces.

Now, we define our algortithm formally:

[H] **Algorithm 1** DDPG algorithm

```
Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weight $\theta^Q$ and $\theta^\mu$.
Initialize target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer $R$
for episode = 1, M do
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state $s_1$
  for t = 1, T do
    Select action $a_t = \mu\left(s_t \mid \theta^\mu\right) + \mathcal{N}_t$ according to the current policy and exploration noise
```

```
        Execute action at and observe reward $r_t$ and observe new state $s_{t+1}$
        Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$
        Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$
        Set $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^Q\right)\right)^2$
        Update the actor policy using the sampled policy gradient:
```

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a \mid \theta^Q\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu\left(s \mid \theta^\mu\right)\Big|_{s_i} \tag{7}$$

```
        Update the target networks:
```

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \tag{8}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'} \tag{9}$$

```
    end for
end for
```

3 Empirical Results

Tables 1 compare the results from using reinforcement learning with delta hedging for short positions in at-the money ($S_0 = K$) call options on a stock lasting one month and three months when $\mu = 5\%$, $r = 0$, $q = 0$. We set $c = 1.5$ in equation (6) so that the hedger's objective is to minimize the mean cost of hedging plus 1.5 times the standard deviation of the cost of hedging. The trading cost parameter, κ, is 1%.

	Delta	Deep Q
Weekly	838.22	670.42
4 Days	669.65	589.21
2 Days	355.69	297.54
Daily	211.17	190.03

Table 1: Cost

As we can see, it converges quickly. So, it seems very effective. The table shows that using RL optimal hedging instead of delta hedging has a small negative impact on the standard deviation of the hedging cost in the cases we consider, but significantly improves the mean cost of hedging.

4 Conclusions

From this paper, we develop our model to solve optimal hedging strategies in the presence of transaction costs using the DDPG approach and explained how reinforcement learning can be used to produce an optimal hedging strategy when a particular stochastic process has been specified. The above work is an extension of the results we made based on what has been done using the Actor-Critic method. Our results show that using RL optimal hedging rather than delta hedging has small negative effect on the standard deviation of the cost of hedging in the situations we consider, but markedly improves the mean cost of hedging.

5 Reference and Appendix

Please see https://github.com/JinTian0717/ddpg