Optional Assignment: European Option Hedging with Deep Reinforcement Learning Methods

XU Guosong, 20830011, gxuae@connect.ust.hk,
CHEN Zuozhi, 20797609, zchenfu@connect.ust.hk
(https://github.com/LegendreXu/ReinforcementLearning/tree/main/Optional_bonus)

Abstract. Deep Reinforcement Learning to hedge options is regarded as the start of something revolutionary in quantitative finance. Several big-name firms, including J.P. Morgan, Bank of America, and Societe Generale, are working on this project. This report begins with formulating market environment assumptions and the connection between hedging and optimal control. It then further implements two Deep Reinforcement Models to hedge a vanilla European call option and discuss their results.

1.Introduction

The last time we investigated how to hedge a stock under the binomial tree model, the most common financial derivative that needed to be hedged was the European vanilla option. The Black-Scholes-Merton model provides a proper framework to formulate it as a stochastic optimal control problem. Furthermore, the fast development of Deep Learning and GPU offers us an opportunity to apply Neural Networks to RL algorithms to solve complicated problems. Hence, it is natural for us to solve the problem with Deep Reinforcement Learning.

2.Problem Formulation

2.1.Market Environment

In order to keep the log-return as a normal distribution with zero expected value, we assume that the underlying stock price follows Geometric Brownian Motion:

$$S_t = S_0 \exp[\sigma W_t + (\mu - \frac{\sigma^2}{2})t]$$

Where $W = W_t, 0tT$ is a Standard Brownian Motion defined on measure space $(\Omega, \mathscr{F}, \mathscr{F}_{0 \leq t \leq T}, P)$. As for the Environment simulation, we discretized the time interval and use

$$S_{t_i} = S_{t_{i-1}} \exp[\sigma\sqrt{t_i - t_{i-1}}\varepsilon_i + (\mu - \frac{\sigma^2}{2})(t_i - t_{i-1})] \; where \varepsilon_i \sim \mathscr{N}(0,1)$$

we assumed that

- we have 1 unit of short position of ATM European vanilla call option C_0 ,whose underlying stock is S_0 and the initial hedging ratio N_0.
- The risk-free rate is 0 for the convenience of computation.
- For DQN Agent, the action space is $\{0, 0.01, 0.02, 0.03, \ldots, 0.99, 1\}$ with 101 uniformly sliced elements.

2.2.Optimal Control

As an optimal control problem, we define the state as current stock price and time to maturity:

$$state_t = \mathscr{S}_t := R_+^2 = \{(S_t, \tau)\}, \tau = T - t$$

. with the target stock position as control variable or action:

$$action_t = a_t = N_t$$

Consequently, we can get the reward function by the procedure of hedging. Without loss of generality, we assume that the risk-free rate is zero. Hence, the reward function is the conditional expectation:

$$r_t = -|E[(N_{t_{i+1}}(S_{t_{i+1}} - S_{t_i}) - (C_{t_{i+1}} - C_{t_i})|\mathscr{F}_{t_i}]|$$

3.Models

This report needs to compare a NN algorithm of Reinforcement Learning and DDPG, but we had many failures with the Convergence of naive DQN. After a short discussion with TA, we tried other NN models like NAF, PPO, Double DQN... Eventually, we got that the best NN model is Dueling DQN.

3.1. Dueling Deep Q-Network(DDQN)

DQN is a Reinforcement Learning algorithm combined with Q-Learning and ANN, which learns the Q-function $Q(\mathscr{S}_t, a_t, \theta)$ by the Neural Network. While Dueling DQN is a DQN with a Dueling network architecture:

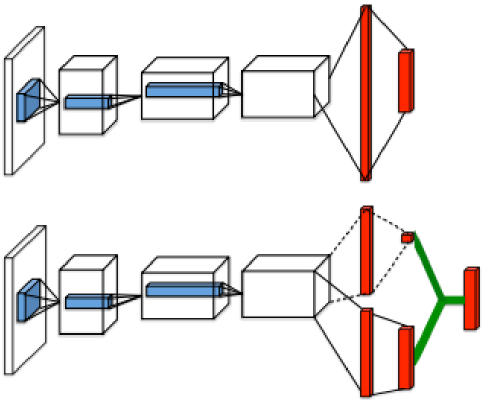

Figure 1. Comparison between architectures of DQN and Dueling-DQN.

Inspired by the idea of Dueling DQN, we noticed that in our problem setting, our action would have no influence on the environment since the environment is only affected by the Brownian Motion of the stock, and we assume no trading cost. Therefore, even a good action in a state will not guarantee to lead us to a good state. However, the brilliant idea of Dueling DQN helps us better fetch those good actions.

As for the implementation of Dueling DQN, we follow the most common architecture. Besides, borrowing from the idea of DDPG, we use the soft target update to improve the performance. The Q-function $Q^{\pi}(s,a)$ and advantage function $A^{\pi}(s,a)$ are defined as:

$$\begin{cases} Q^{\pi}(\mathscr{S},a) = E\left[R_t \mid \mathscr{S}_t = \mathscr{S}, a_t = a, \pi\right] \\ V^{\pi}(\mathscr{S}) = E_{a\sim\pi(s)}\left[Q^{\pi}(\mathscr{S},a)\right] \\ A^{\pi}(\mathscr{S},a) = Q^{\pi}(\mathscr{S},a) - V^{\pi}(\mathscr{S}) \end{cases}$$

And the forward mapping is defined as:

$$Q(\mathscr{S},a;\theta,\alpha,\beta) = V(\mathscr{S};\theta,\beta) + \left(A(\mathscr{S},a;\theta,\alpha) - \sum_{a'} A\left(\mathscr{S},a';\theta,\alpha\right)/|A|\right)$$

3.2. Deep Deterministic Policy Gradient(DDPG)

Due to the nature of DQN, $\pi^* = \arg\max_{a'} q^*(\mathscr{S},a')$, it is hard to be applied to continuous problem. However, DDPG can solve the continuous hedging problem better since it optimizes the policy instead of the action. DDPG has two networks: the Actor network $\mu\left(s \mid \theta^{\mu}\right)$ and the Critic network $Q\left(\mathscr{S},a \mid \theta^{Q}\right)$. With a OU-process $\mathscr{N}_{\sqcup}$, we choose the action:

$$a_t = \mu\left(\mathscr{S}_t \mid \theta^{\mu}\right) + \mathscr{N}_{\sqcup}$$

Then we randomly sample a batch and calculate the loss function L and update the Critic network:

$$\begin{cases} y_i = r_i + \gamma Q'\left(\mathscr{S}_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right) \\ L = \frac{1}{N}\sum_i \left(y_i - Q\left(\mathscr{S}_i, a_i \mid \theta^{Q}\right)\right)^2 \end{cases}$$

Afterwards, we update the Actor network by the sample policy gradient:

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

And now we can update the two target network.

4.Numerical Result

We trained Dueling DQN and DDPG model $N = 15000$ episodes respectively and get the final results of hedging agents.

4.1.Convergence

Since we define the negative absolute value of expected return as our training reward, it is supposed to converge to 0 as the training episode goes to ∞. In order to get a smooth graph, we choose the average reward of latest 50 episodes to plot.

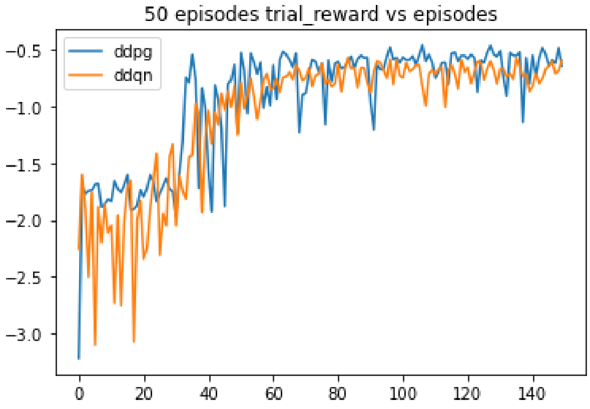

Figure 2. Reward curve of Dueling-DQN and DDPG.

The x-axis represents 100 eposides.

4.2.Risk Comparison

Under friction-less GBM assumption and Black-Scholes-Merton Framework, the theoretical optimal hedging strategy should be Delta hedging. Hence we compared the result of agents and the theoretical Delta value:

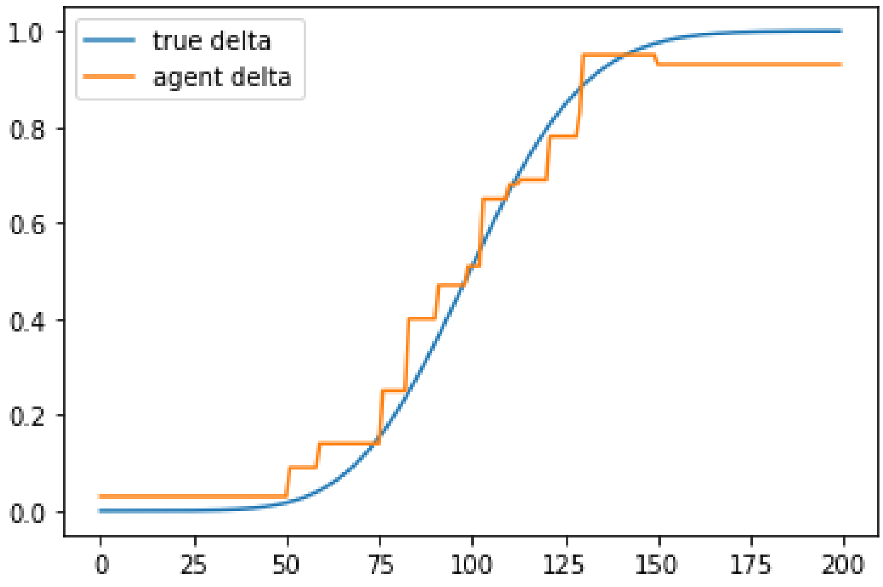

Figure 3. Comparison of DDQN and ground truth.

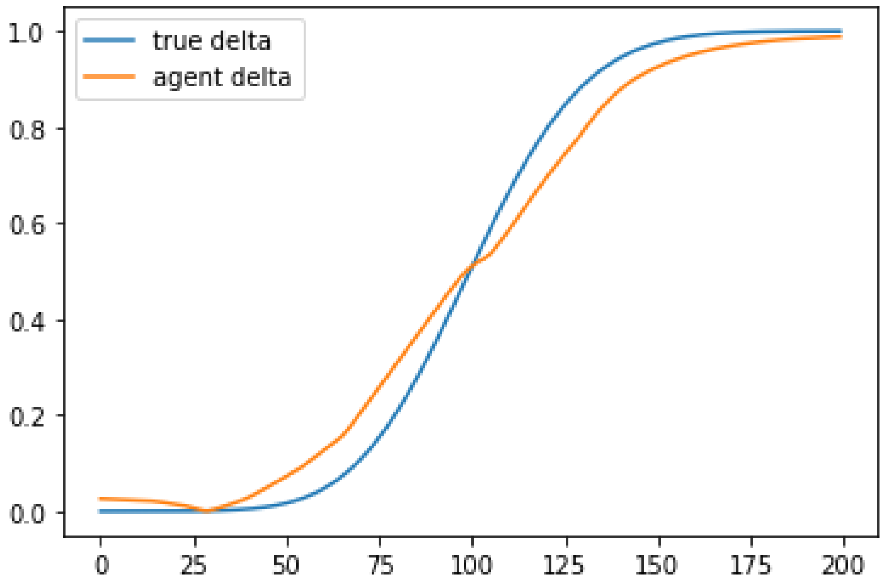

Figure 4. Comparison of DDPG and ground truth.

5.Conclusion

We figured out that the convergent speed of DDQN is slower than the one of DDPG(the blue curve is slightly higher than the orange curve in Figure 1). Moreover, as Figure 2 and Figure 3 illustrate, the hedging result of DDPG is closer to the ground truth as well. Hence, we conclude that DDPG would have better performance in solving the continuous-valued problems like hedging an option in the GBM world. However, we noticed that we assumed zero risk-free rate, friction-less market, and perfect divisibility of underlying stock. There could be plenty of discrepancies between theory and reality. For instance, non-zero risk-free rate, cost of carry, commission fee, taxes, market impact, non-Gaussian log-return, settlement conventions... We are willing to spare more effort on related research in our future life.

Reference

1 DeepMind, Z. W. G., Wang, Z., DeepMind, G., DeepMind, T. S. G., Schaul, T., DeepMind, M. H. G., Hessel, M., DeepMind, H. V. H. G., Hasselt, H. V., DeepMind, M. L. G., Lanctot, M., DeepMind, N. D. F. G., Freitas, N. D., amp; Metrics, O. M. V. A. (2016, June 1). Dueling network architectures for Deep Reinforcement Learning: Proceedings of the 33rd International Conference on International Conference on Machine Learning - volume 48. Guide Proceedings. Retrieved May 6, 2022, from https://dl.acm.org/doi/10.5555/3045390.3045601

2 Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2022. Continuous control with deep reinforcement learning. [online] arXiv.org. Available at: ¡https://arxiv.org/abs/1509.02971¿ [Accessed 6 May 2022].

Acknowledgments

This group assignment is completed by XU Guosong and CHEN Zuozhi with equal contribution. We had many times of video and face-to-face discussions to design this algorithm. Moreover, we tuned the parameters together for a real long time And this report is wrote by us together as well, and XU is responsible for the LaTeXcode. Also, we would like to express our sincere appreciation to Professor Chak WONG and TAs because of their patient help. Especially, Mr.Fang gave us very important talk on WeChat.
And the DDPG code was inspired by
github.com/philtabor/Youtube-Code-Repository/blob/master/ReinforcementLearning/PolicyGradient/DDPG/pytorch/lunar-lander/ddpg_torch.py