Comparison of Option Hedging Strategies Using Deep Q Network and Deep Deterministic Policy Gradient with the Time-Discretized Black-Scholes-Merton Model with Transaction Costs

Yuki Yasuda

June 14, 2019

Abstract

This report investigates option hedging strategies with the Black-Scholes-Merton model with transaction costs, using two reinforcement learning algorithms: Deep Q Network (DQN) and Deep Deterministic Policy Gradient (DDPG). The DQN is simple and applied to a problem with a discrete action space, while the DDPG is complicated and applied to a problem with a continuous action space. An agent hedges 100 call options with fixed strike and expiry, where the underlying stock price obeys the geometric Brownian motion. The case of no transaction costs is first discussed to examine the convergence of policy. In this case, the optimal policy is the delta hedging. The DDPG policy is closer to the delta hedging than the DQN. The case of non-zero transaction costs is next examined. The DDPG and the DQN agent tend to reduce the number of shares of stock in their positions to cut the transaction costs when the option is in the money (ITM). The DDPG agent has a smaller number of shares of stock, in the case of not only ITM but also OTM (out of the money). The DQN agent changes a number of shares of stock not frequently. The results reveal that the hedging strategy of the DQN is statistically different from that of the DDPG.

1 Introduction

European call options have been traded over the world since the Black-Scholes-Merton (BSM) model was derived [1]. Pricing or hedging an option is categorized into stochastic optimal control problems. In these days, reinforcement learning (RL) [2] has been rapidly developed, influenced by the development of deep learning [3]. RL is a method to solve stochastic optimal control problems. It may be natural to apply RL algorithms to hedge or price options.

Halperin [4, 5] discussed how to develop the time-discretized BSM model with a RL techniques named Q-learning [6]. He showed that both the option pricing and hedging can be performed within a framework of the action-value function (Q function). We tried to develop his theory to the time-discretized BSM model with transaction costs, however, the theory became too complicated (not

shown in detail). The main reason for the complication is that both hedging and pricing options are performed within one framework of the Q-learning.

Kolm and Ritter [7] developed a different theory of option hedging with the Q-learning. They separated hedging from pricing and discussed how to hedge an option that is priced with the (continuous-time) BSM model in a realistic market where time is discrete and transaction costs are not zero. Their theory is easily applied to not only European options but also more complicated derivatives. However, in their theory, the number of shares of stock in the hedger position must be treated as an integer, even if its value is quite large [e.g., $O(10^5)$], because the Q-learning cannot be applied to a problem with a continuous action space. It would be more practical to regard the number of shares of stock as continuous, particularly in a larger brokerage securities.

In this report, we applied both the Q-learning and the Deep deterministic policy gradient (DDPG) algorithm to the option hedging and compare both hedging strategies. DDPG is a RL algorithm that can solve a problem with a continuous action space [8, 9]. This report is organized as follows. In Section 2, the Q-learning (deep Q network) and the DDPG are briefly reviewed. An idealized market and a problem of hedging options are constructed in Section 3 on the basis of the theories of Halperin [4, 5] and Kolm and Ritter [7]. Numerical experiments are performed in Section 4, and we compare the hedging strategies. In Section 5, we finally summarize the results obtained in this report.

2 Brief summary on reinforcement learning

We briefly summarize the framework of reinforcement learning (RL) with Markov decision process [2] and then review the two RL algorithms used in this report: Deep Q Network (DQN) [6, 10] and Deep Deterministic Policy Gradient (DDPG) [8, 9].

2.1 Markov decision process and Bellman equations

RL is a method to solve multi-period optimal control problems. An optimal control problem can be formulated using agent and environment. The agent determines an action a_t at every time t, following its policy π and the current state s_t. The environment gives the next state s_{t+1} and the reward r_t based on the agent action a_t and the current state s_t, where the transition probability is denoted by $p(s_{t+1} \mid s_t, a_t)$ and r_t is a function of s_t, a_t, and s_{t+1}. The process is repeated from $t = 0$ to $t = T$. The objective of RL is to obtain the optimal policy π^* that maximizes the cumulative sum of discounted rewards:

$$R_0 = \sum_{t=0}^{T} \gamma^t r_t(s_t, a_t, s_{t+1}), \tag{1}$$

where γ is a discount rate ($0 < \gamma < 1$) and r_T is determined only by s_T.

The action-value function is introduced to solve the control problem:

$$q_t^\pi(s,a) := \mathbb{E}[R_t \mid s_t = s, a_t = a], \tag{2}$$

where $R_t := \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$. Using the tower rule, we obtain the Bellman equation:

$$q_t^{\pi}(s,a) = \mathbb{E}\left[r_t + \gamma q_{t+1}^{\pi}(s',a') \,\middle|\, s_t = s, a_t = a\right]. \tag{3}$$

The optimal policy π^* gives the maximum action-value function $q_t^*(s,a)$ for any t, s, and a:

$$q_t^*(s,a) \geq q_t^{\pi}(s,a). \tag{4}$$

The function of $q_t^*(s,a)$ satisfies the Bellman optimality equation:

$$q_t^*(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a'} q_{t+1}^*(s',a') \,\middle|\, s_t = s, a_t = a\right]. \tag{5}$$

The optimal policy π^* is obtained in principle by solving the Bellman optimality equation (5). According to the contraction mapping theorem, the unique solution exists for the Bellman optimality equation, which is $q_t^*(s,a)$. The optimal policy is then given by $\pi^* = \arg\max_a q_t^*(s,a)$ for any t and s. However, solving the Bellman optimality equation is quite difficult because the transition probability $p(s_{t+1} \mid s_t, a_t)$ is often unknown and the dimension of state s or action a is quite large (sometimes continuous) in most cases. We have to apply some approximation and sampling methods to obtain the optimal policy.

2.2 Deep Q network (DQN)

Deep Q Network (DQN) [10] is a combination of deep learning [3] and Q-learning [6]. DQN is often applied to a problem in which state is continuous, action is discrete, and the transition probability is unknown. The optimal action-value function is approximated with a deep neural network: $q_t^*(s,a) \approx q_t^*(s,a;\boldsymbol{w})$, where $\boldsymbol{w}$ represents the weight parameters in the deep network. The Bellman optimality equation is solved by sampling data (s_t, a_t, r_t, s_{t+1}) from some buffer. The DQN algorithm is in the following [10].

Algorithm of Deep Q Network (DQN) [10]

Initialize replay buffer *RB*
Initialize action-value function q^* with random weights $\boldsymbol{w}$
for episode $= 1, M$ **do**
  Initialize sequence $s_0 = x_0$
  for $t = 0, T$ **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \arg\max_{a'} q^*(s_t, a')$
    Execute action a_t in the environment and observe r_t and s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in *RB*
    Sample random mini-batch of transitions (s_j, a_j, r_j, s_{j+1}) from *RB*

$$\text{Set } y_j = \begin{cases} r_j & \text{for terminal } s_{t+1} \\ r_j + \gamma \max_{a'} q^*(s_{t+1}, a') & \text{for non-terminal } s_{t+1} \end{cases} \tag{6}$$

    Perform a semi-gradient descent step on $(y_j - q^*(s_j, a_j; \boldsymbol{w}))^2$
  end for
end for

2.3 Deep deterministic policy gradient (DDPG)

DQN is difficult to apply a problem in which both state and action are continuous because the optimal policy is given by $\pi^* = \arg\max_{a'} q^*(s, a')$. Deep deterministic policy gradient (DDPG) algorithm can be applied to such a problem [8, 9]. In DDPG, the optimal policy is a function of state $\mu(s, \boldsymbol{\theta}^\mu)$, which is represented by a deep network with weights $\boldsymbol{\theta}^\mu$. This function is obtained by optimizing the action-value function $q(s, a; \boldsymbol{\theta}^q)\,|_{a=\mu(s)}$, which is described by another deep network with weights $\boldsymbol{\theta}^q$. The function $\mu(s, \boldsymbol{\theta}^\mu)$ is updated according to the deterministic policy gradient theorem [8]. The DDPG algorithm is in the following [9].

Algorithm of Deep Deterministic Policy Gradient (DDPG) [9]

Randomly initialize critic network $q(s,a;\boldsymbol{\theta}^q)$ and actor $\mu(s;\boldsymbol{\theta}^\mu)$
Initialize target network q' and μ' with weights $\boldsymbol{\theta}^{q'} \leftarrow \boldsymbol{\theta}^q$ and $\boldsymbol{\theta}^{\mu'} \leftarrow \boldsymbol{\theta}^\mu$
Initialize replay buffer *RB*
for episode $= 1, M$ **do**
  Initialize a random process ξ_t (e.g., Ornstein-Uhlenbeck process) for action exploration
  Receive initial observation state s_0
  for $t = 0, T$ **do**
    Select action $a_t = \mu(s_t;\boldsymbol{\theta}^\mu) + \xi_t$ according to the current policy
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in *RB*
    Sample a random mini-batch of n transitions (s_j, a_j, r_j, s_{j+1}) from *RB*
    Set $y_j = r_j + \gamma q'(s_{j+1}, \mu'(s_{j+1};\boldsymbol{\theta}^{\mu'});\boldsymbol{\theta}^{q'})$
    Update critic by minimizing the loss: $L = \frac{1}{n}\sum[y_j - q(s_j, a_j;\boldsymbol{\theta}^q)]^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{n}\sum \frac{\partial q(s_j, a;\boldsymbol{\theta}^q)}{\partial a} \,|_{a=\mu(s_j;\boldsymbol{\theta}^\mu)}\, \nabla_{\theta^\mu}\mu(s_j;\boldsymbol{\theta}^\mu) \tag{7}$$

    Update the target networks:

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

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

  end for
end for

3 Time-discretized Black-Scholes-Merton model with transaction costs

3.1 Idealized market

In this section, we formulate the time-discretized Black-Scholes-Merton (BSM) model with transaction costs. We consider the following idealized market:

- Only the two assets exist: stock S_t and riskless bank account B_t.
- The stock price S_t obeys the geometric Brownian motion (10).
- The shares of stock is represented by an integer.

- The risk-free interest rate ρ is constant.
- The market has no holidays.
- The assets are traded at discrete time steps under the constraint of the self-financing condition.
- Transaction costs are described by a function $f := f(S_t, \delta N_t)$, where δN_t is a change of the number of shares of stock N_t.

The geometric Brownian motion is

$$\delta S_t = \mu S_t \, \delta t + \sigma S_t \, \delta W_t, \tag{10}$$

where μ and σ are constant and W_t is a Wiener process. According to the self-financing condition, B_{t+1} is given by

$$B_{t+1} = B_t e^{\rho \delta t} - (N_{t+1} - N_t) S_t - f(S_t, \delta N_t), \tag{11}$$

where δN_t is equal to $N_{t+1} - N_t$. The time evolution and the portfolio rebalance are summarized in the following:

$$\underbrace{(S_t, N_t, B_t e^{\rho \delta t})}_{\text{time } t} \quad \to \quad \underbrace{(S_t, N_{t+1}, B_{t+1})}_{\text{time } t \text{ after rebalancing}} \quad \to \quad \underbrace{(S_{t+1}, N_{t+1}, B_{t+1} e^{\rho \delta t})}_{\text{time } t+1}. \tag{12}$$

The agent sells 100 European call options at $t = 0$ with strike K and expiry T and hedges the 100 options until $t = T$ (i.e., the agent position is fixed). The option price is given by the (continuous-time) BSM model [1].

3.2 State, action, and reward

We re-formulate the above problem using the language of Markov decision process. The state s_t is first introduced by combining the ideas of [4, 5] and [7]:

$$s_t := (t, W_t, C_t, \Delta_t, N_t), \tag{13}$$

where t is the discretized time, W_t is a Wiener process, C_t is the option price given by the BSM model, Δ_t is the option delta multiplied by 100 ($\Delta_t := 100 \times \frac{\partial C_t}{\partial S_t}$), and N_t is the number of shares of stock in the portfolio. As the agent hedges the 100 call options, the delta neutral position is constructed by making N_t equal to Δ_t. The stock price S_t is given by W_t:

$$S_t = S_0 \exp\left[\sigma W_t + \left(\mu - \frac{\sigma^2}{2}\right) t\right], \tag{14}$$

which is the solution of the (continuous-time) geometric Brownian motion. The action a_t is a number of shares of stock at $t+1$:

$$a_t(s_t) := N_{t+1}. \tag{15}$$

Following (12), the Markov decision process is defined as

$$\underbrace{(t, W_t, C_t, \Delta_t, N_t)}_{s_t \text{ at time } t} \quad \xrightarrow[a_t]{} \quad \underbrace{(t, W_t, C_t, \Delta_t, N_{t+1})}_{\text{time } t \text{ after taking action } a_t} \quad \xrightarrow[r_t]{} \quad \underbrace{(t+1, W_{t+1}, C_{t+1}, \Delta_{t+1}, N_{t+1})}_{s_{t+1} \text{ at time } t+1}, \tag{16}$$

where r_t is a reward determined by s_t, a_t, and s_{t+1}. The state s_t does not need to contain the bank account B_t because (11) and (14) uniquely determine B_t with the given state s_t.

We finally specify the function of r_t on the basis of [7]. The agent portfolio at time t is described as

$$\Pi_t = N_t S_t + B_t e^{\rho \delta t} - 100\, C_t. \tag{17}$$

The function r_t is determined through the following cumulative sum of discounted rewards:

$$R_0 := \mathbb{E}\left[(N_T S_T - N_0 S_0) + (100 C_0 - N_0 S_0) e^{\rho T} + \sum_{t=0}^{T-1} \{B_{t+1} - B_t e^{\rho \delta t}\} e^{\rho (T-t)} - 100 C_T \right] \tag{18}$$

$$- \frac{\kappa}{2} \mathrm{Var}\left[\Pi_T\right], \tag{19}$$

where the first term is the expected return and the second one is the variance of the portfolio (i.e., the magnitude of risk). The balance between return and risk is controlled by the constant of κ. The term of $(100C_0 - N_0 S_0)e^{\rho T}$ represents that the agent makes a deposit at $t = 0$ immediately after selling the 100 options. We set $N_0 = 0$ without loss of generality. The agent has to determine the initial portfolio immediately after selling the 100 options, i.e., the agent determines the initial number of shares of stock.

The return (18) is decomposed as follows:

$$(N_T S_T - N_0 S_0) + \{(100 C_0 - N_0 S_0) e^{\rho T} - 100 C_0\} + \sum_{t=0}^{T-1} \{B_{t+1} - B_t e^{\rho \delta t}\} e^{\rho (T-t)} - 100 (C_T - C_0) \tag{20}$$

$$= \sum_{t=0}^{T-1} (N_{t+1} S_{t+1} - N_t S_t) + \{B_{t+1} - B_t e^{\rho \delta t}\} e^{\rho (T-t)} - 100 (C_{t+1} - C_t) \tag{21}$$

$$+ \{(100 C_0 - N_0 S_0) e^{\rho T} - 100 C_0\} \tag{22}$$

$$= \sum_{t=0}^{T-1} (N_{t+1} S_{t+1} - N_t S_t) - \{(N_{t+1} - N_t) S_t + f(S_t, \delta N_t)\} e^{\rho (T-t)} - 100 (C_{t+1} - C_t) \tag{23}$$

$$+ \{(100 C_0 - N_0 S_0) e^{\rho T} - 100 C_0\}, \tag{24}$$

where the self-financing condition (11) is used.

The variance (19) is decomposed as follows:

$$\begin{aligned}
\mathrm{Var}\left[\Pi_T\right] &= \mathrm{Var}\left[\sum_{t=0}^{T-1} \delta\Pi_t\right] && (25)\\
&= \sum_{t=0}^{T-1} \mathrm{Var}\left[\delta\Pi_t\right] && (26)\\
&= \sum_{t=0}^{T-1} \mathrm{Var}\left[N_{t+1}\delta S_t - 100\delta C_t\right] && (27)\\
&\approx \sum_{t=0}^{T-1} \mathrm{Var}\left[N_{t+1}\delta S_t - 100\frac{\partial C_t}{\partial S_t}\delta S_t\right] && (28)\\
&= \sum_{t=0}^{T-1} \left[(N_{t+1} - D_t)\sigma S_t\right]^2 \delta t, && (29)
\end{aligned}$$

where we use the tower rule, the Markov property, the equation (12), the Taylor expansion, and the equation (10).

Collecting (23), (24), and (29), the function r_t is defined by

$$r_t := \begin{cases} \gamma^{-t}\left[\delta r_t - \frac{\kappa}{2}\delta v_t\right] & (0 \le t < T) \\ \gamma^{-T}\left[(100C_0 - N_0S_0)e^{\rho T} - 100C_0\right] & (t = T) \end{cases} \tag{30}$$

where

$$\delta r_t = (N_{t+1}S_{t+1} - N_tS_t) - \{(N_{t+1} - N_t)S_t + f(S_t, \delta N_t)\}e^{\rho(T-t)} - 100(C_{t+1} - C_t), \tag{31}$$

$$\delta v = \left[(N_{t+1} - D_t)\sigma S_t\right]^2 \delta t. \tag{32}$$

Note that the above definition satisfies $R_0 = \sum_{t=0}^{T} \gamma^t r_t$, where R_0 is given by (18) and (19).

4 Numerical experiments

We first specify the configuration of numerical experiments, which is similar to the configuration of [7]. The values of all fixed parameters are summarized in Table 1. A function of transaction costs is given by

$$f(S_t, \delta N_t) = \alpha S_t\left[|\delta N_t| + 0.01 \times \delta N_t^2\right], \tag{33}$$

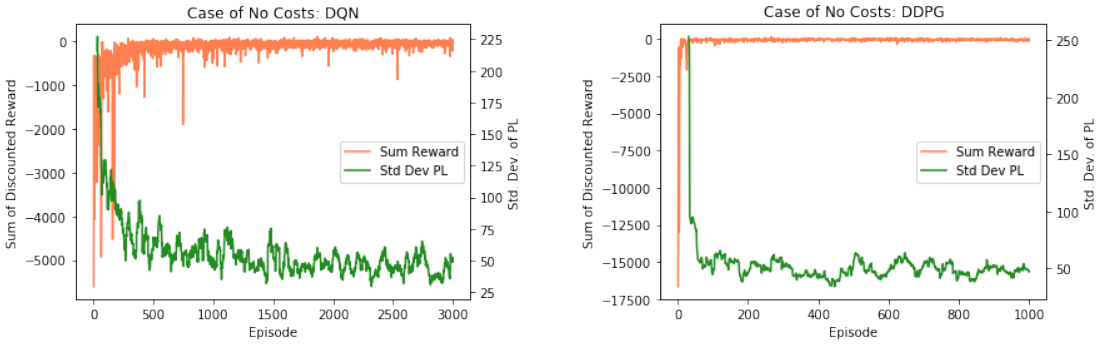
where α is a constant controlling the strength of market friction. The value of α is specified in each subsection. The number of episodes M is 3000 for DQN and 1000 for DDPG. The size of mini-batch is 32 and the capacity of reply buffer is 10000, both of which are common to DQN and DDPG. An action of DDPG is rounded to make it an integer.

Name	Value
T	30 [days]
Time steps	30 (i.e., 1 step per day)
μ	5 % (yearly rate)
σ	20 % (yearly volatility)
ρ	5 % ($=\mu$, yearly rate)
S_0	100
K	100
N_0	0
κ	0.1
γ	0.99

Table 1: Values of all fixed parameters.

4.1 Cases of no transaction costs

In this subsection, we investigate the cases of no transaction costs [i.e., $\alpha = 0$ in (33)]. In this case, the optimal policy is the delta hedging [7], that is, the optimized agent makes the delta neutral position at every time. We confirm whether the DQN and the DDPG policy are sufficiently convergent to the delta hedging policy.



(a) DQN (b) DDPG

Figure 1: Learning curves of (a) DQN and (b) DDPG. The standard deviation of PL at each episode is calculated from the last 30 episodes.

Figures 1a and 1b are the learning curves of DQN and DDPG, respectively. Each figure shows the histories of the cumulative sum of discounted rewards (orange) and the standard deviation of profit and loss (PL) at the expiry (green), where the standard deviation at each episode is calculated from the last 30 episodes. The sum of rewards should be convergent to zero because the

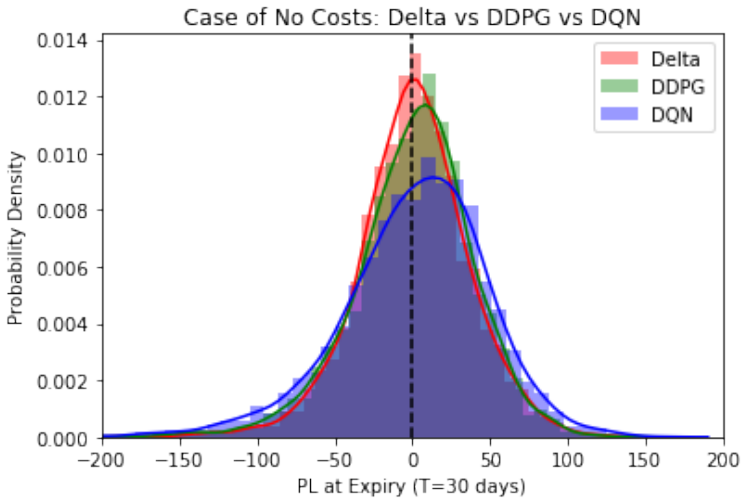


Figure 2: Histograms of PL at the expiry ($t = T$) by the Delta (red), the DDPG (green), and the DQN (blue) agent. The PL is defined by (34). The PL values are calculated from newly sampled 3000 paths. Each curve represent the kernel density estimation.

delta hedging is optimal. The standard deviation of PL should be smaller as the learning appropriately proceeds. Figures 1a and 1b suggest that both DQN and DDPG learnings are sufficiently convergent.

Although the optimal policy is the delta hedging [7], the hedge error is not exactly zero due to the discreteness of time [11]. The hedge error of an agent can be measured by the standard deviation of PL at the expiry ($t = T$):

$$\mathrm{PL}_T := (N_T S_T - N_0 S_0) + (100C_0 - N_0 S_0)e^{\rho T} + \sum_{t=0}^{T-1} \{B_{t+1} - B_t e^{\rho \delta t}\} e^{\rho(T-t)} - 100C_T. \tag{34}$$

Note that the expectation of PL at the expiry is the first term of the cumulative sum of discounted rewards (18). The delta hedging agent (shortly, Delta agent) has the minimum standard deviation of PL because its policy is optimal. Figure 2 shows the histograms of PL by the Delta (red), the DDPG (green), and the DQN (blue) agent, which are calculated from newly sampled 3000 paths. The standard deviation of PL by the DDPG agent is smaller than that by the DQN, which means that the DDPG solution is closer to the global optimizer.

The DQN agent has the hidden layers of 16x16 (i.e., two hidden layers, each of which has 16 units). In the DDPG agent, the hidden layers of the actor and the critic are 12x12x12 and 24x24x24, respectively. These facts suggest that the standard deviation of PL by the DQN is larger because its deep network does not have sufficiently large degree of freedom. We perform similar numerical experiments using the DQNs with no hidden layer or the hidden layers of 16x16x16 (three hidden

layers) and 16 (one hidden layers). The standard deviation of PL is smallest in the case of the 16x16 hidden layers (i.e., two hidden layers). This suggests that the above result of convergence (Fig. 2) is not sensitive to the DQN architecture.

The DQN agent selects an action from the 101 possible ones, which is the number of shares of stock from 0 to 100. An action of the DQN is determined by $\arg\max_{a'} q^*(s,a')$. These facts indicate that the action space is too large for the agent to explore sufficiently and it is difficult to estimate q^* with high accuracy. In contrast, the DDPG agent determines its action by $a = \mu(s)$. When we use an appropriate function approximation, a DDPG agent does not have to explore the action space in detail. The effective degree of freedom is often smaller than the dimension of action or state space. Such effective degree of freedom is represented by the function parameters, which are tuned by any sample data.

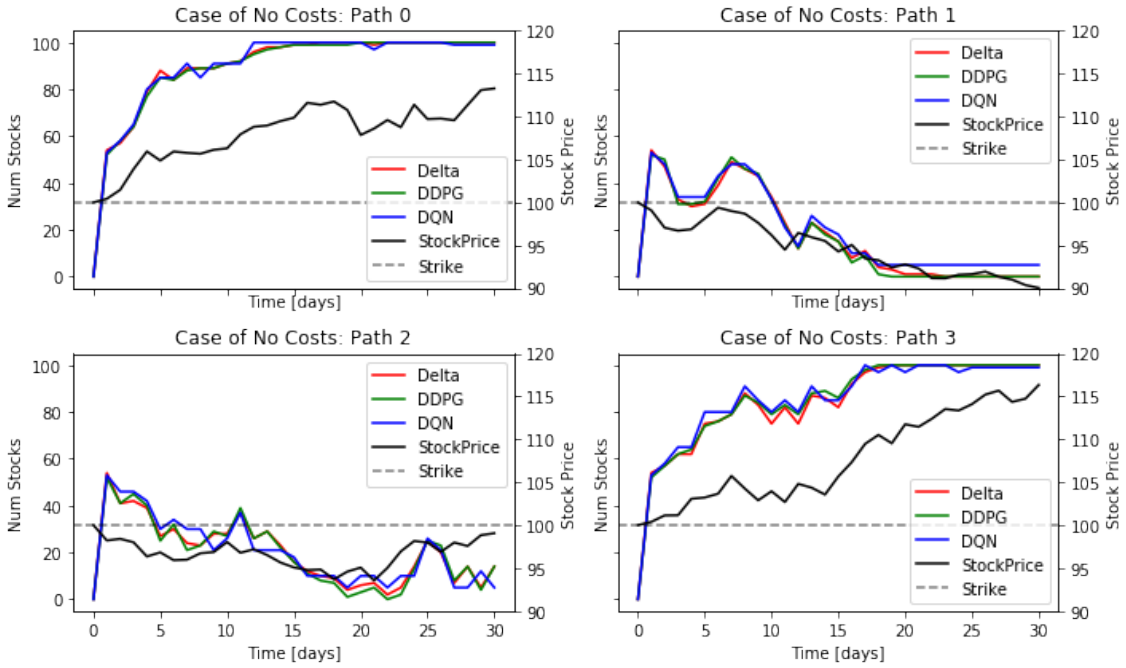


Figure 3: Sample paths. Each figure shows the time series of the number of shares of stock by the Delta (red), the DDPG (green), and the DQN (blue) agent. The time series of stock price is also shown (black). The horizontal dashed lines represent the strike K ($= 100$). In Path 0 and 3, the option is in the money at the expiry ($t = T$), while it is out of the money in Path 1 and 2.

Figure 3 shows four sample paths. Each figure shows the time series of the number of shares of stock by the Delta (red), the DDPG (green), and the DQN (blue) agent, together with the time series of stock price (black). The DDPG paths are closer to the Delta paths. This fact can also be confirmed in Fig. 4, which shows the histograms of the difference between the number of shares

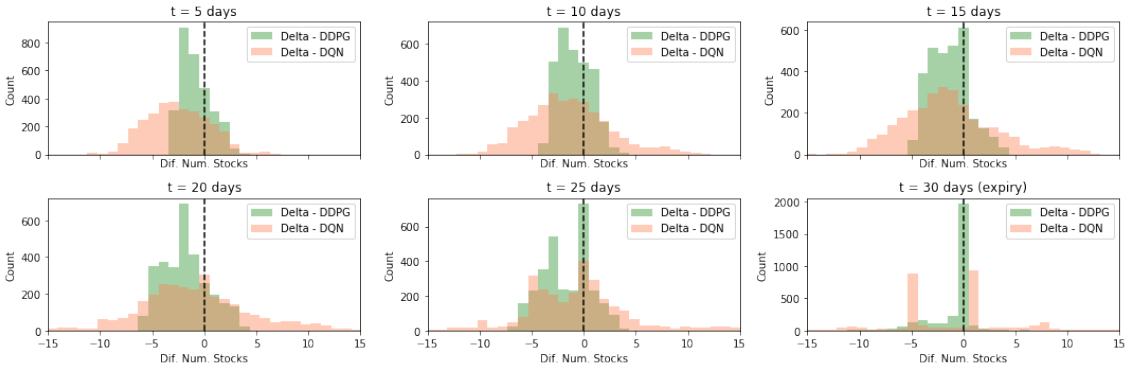


Figure 4: Histograms of the difference between the numbers of shares of stock with no transaction costs. The figures are made from newly sampled 3000 paths.

of stock at various times. The DDPG agent makes the position closer to the delta neutral position at each time.

4.2 Cases of non-zero transaction costs

We investigate the cases of non-zero transaction costs: the value of α in (33) is 1×10^{-3}, 4×10^{-3}, or 8×10^{-3}. In these cases, the delta hedging is not optimal and we expect that the DDPG or DQN policy is close to be optimal.

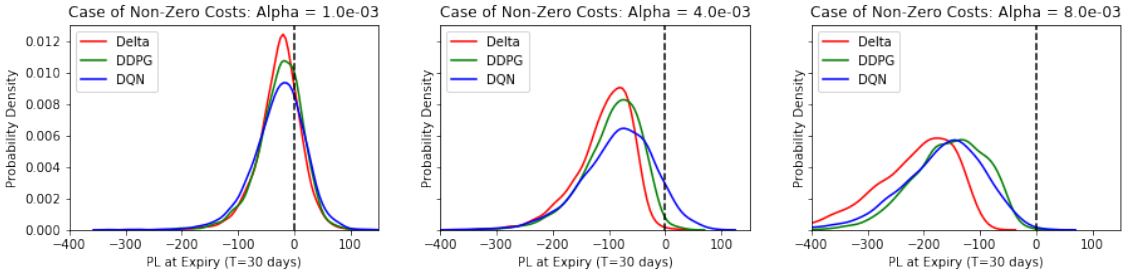


Figure 5: Kernel density estimations of the histograms of PL at the expiry ($t = T$). The estimations are made from newly sampled 3000 paths.

Figure 5 shows the kernel density estimations (KDEs) of the histograms of PL at the expiry ($t = T$) by the Delta (red), the DDPG (green), and the DQN (blue) agent. The statistics of PL are displayed in Tables 2, 3, and 4. When $\alpha = 1 \times 10^{-3}$, the Delta agent gives the smallest standard deviation of PL, while the DDPG gives the largest mean of PL. The KDE is overlapped with each other (Fig. 5) and the difference among the agents is not distinct. When $\alpha = 4 \times 10^{-3}$ or

$\alpha = 8 \times 10^{-3}$, the delta hedging is obviously not a good policy. Comparing the DDPG and the DQN agent, the DDPG tends to have a smaller standard deviation of PL and a larger 1 percentile PL. The DQN agent has the largest mean of PL when $\alpha = 4 \times 10^{-3}$ likely due to the largest standard deviation, while the mean of PL by the DDPG is largest when $\alpha = 8 \times 10^{-3}$. We further investigate the case of $\alpha = 8 \times 10^{-3}$.

Agent	Mean PL	Std. Dev. PL	1 percentile PL
Delta	-26.72	38.30	-137.83
DDPG	-24.80	40.76	-144.94
DQN	-27.18	46.44	-159.23

Table 2: Statistics of PL at the expiry ($t = T$) with $\alpha = 1 \times 10^{-3}$.

Agent	Mean PL	Std. Dev. PL	1 percentile PL
Delta	-108.67	50.60	-262.38
DDPG	-94.00	52.49	-254.46
DQN	-83.44	64.35	-254.72

Table 3: Statistics of PL at the expiry ($t = T$) with $\alpha = 4 \times 10^{-3}$.

Agent	Mean PL	Std. Dev. PL	1 percentile PL
Delta	-217.94	74.32	-437.02
DDPG	-154.43	64.98	-326.55
DQN	-167.59	75.63	-389.44

Table 4: Statistics of PL at the expiry ($t = T$) with $\alpha = 8 \times 10^{-3}$.

Figure 6 shows nine sample paths of the number of shares of stock by the Delta (red), the DDPG (green), and the DQN (blue) agent, together with the time series of stock price (black). Interestingly, the DDPG or DQN agent does not have 100 shares of stock when the option is in the money at $t = T$. The agent reduces the number of shares of stock to cut the transaction costs, which makes a profit over the period. In fact, Table 4 shows the 1 percentile PL of the DDPG or DQN agent is much larger than that of the Delta. Figure 6 also indicates the following two points:

- The DDPG agent tends to have a small number of shares of stock in both cases of ITM (in the money) and OTM (out of the money).
- The DQN agent changes a number of shares of stock not frequently.

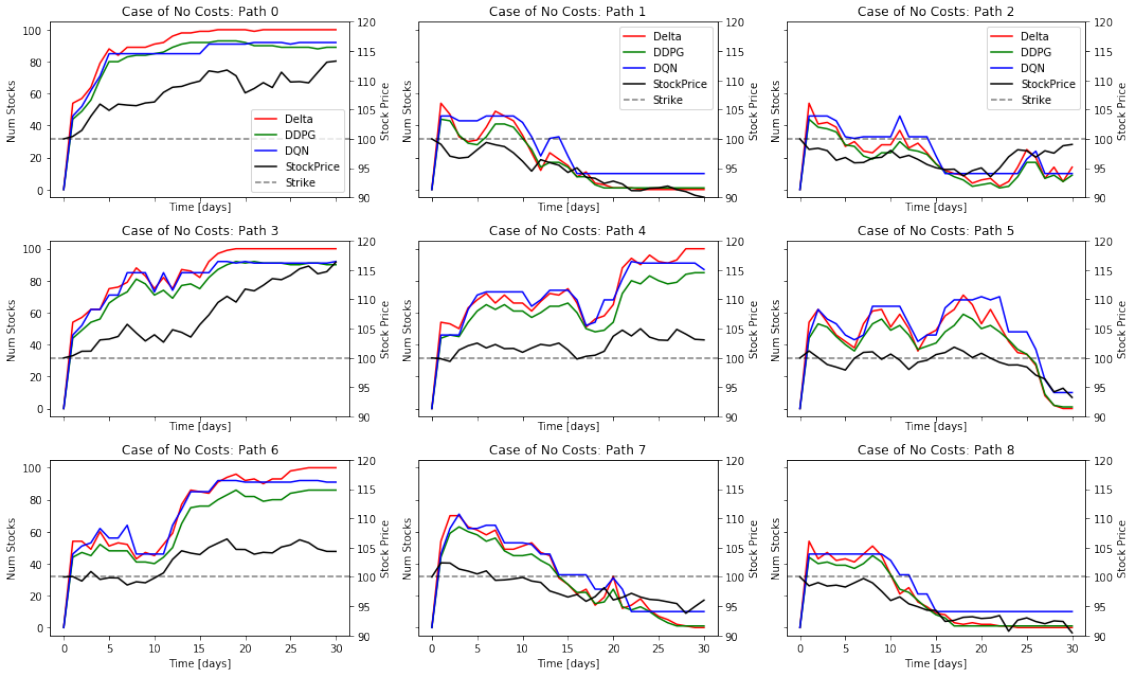


Figure 6: Sample paths with $\alpha = 8 \times 10^{-3}$. Each figure shows the time series of the number of shares of stock by the Delta (red), the DDPG (green), and the DQN (blue) agent. The time series of stock price is also shown (black). The horizontal dashed lines represent the strike K ($= 100$).

These two points suggest that the characteristics of the DQN hedging strategy is statistically different from that of the DDPG strategy, though the PL distributions of the DQN and the DDPG are similar when $\alpha = 8 \times 10^{-3}$ (Fig. 5).

Figure 7 shows the histograms of the difference between the numbers of shares of stock at various times. The DDPG agent (green) tends to have a smaller number of shares of stock compared to the Delta agent. After $t = 15$ days, the histogram of the DDPG (green) becomes bimodal. The larger peak consists of ITM paths, while the smaller peak (close to zero) consists of OTM paths. As discussed before, the DDPG agent tends not to have 100 stocks even when the option is ITM. The DQN agent (orange) follows the different hedging strategy. Before $t = 20$ days, the DQN tends to have a larger number of shares of stock compared to the Delta agent. The histograms become bimodal after $t = 20$ days: the left peak consists of OTM paths, while the right one consists of ITM paths. When the option is ITM, the DQN agent has a smaller number of stock compared to the Delta agent (like the DDPG agent). In contrast, when the option is OTM, the DQN agent has a larger number of stock compared to the Delta agent. The DQN agent has stocks even when the option is OTM at the expiry. This characteristic can also be confirmed in Fig. 6. This result is likely to reflect that the DQN agent sells or buys stocks not frequently.

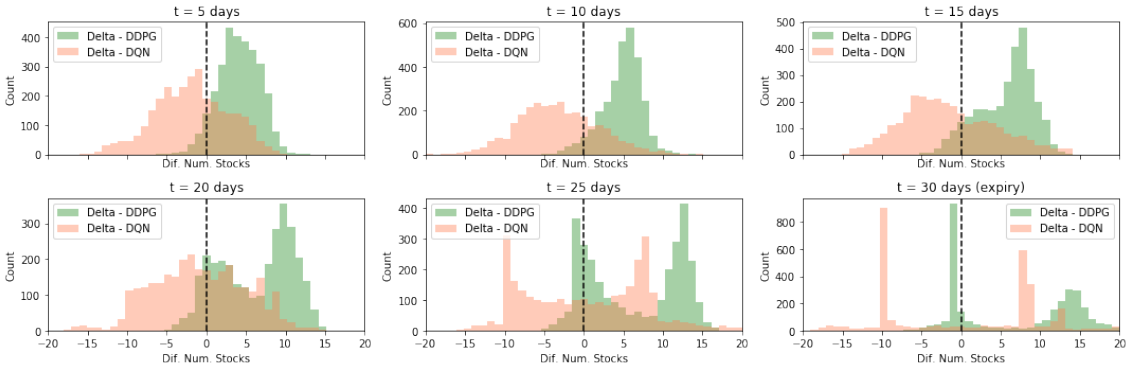


Figure 7: Histograms of the difference between the numbers of shares of stock when $\alpha = 8 \times 10^{-3}$. The histograms are made from newly sampled 3000 paths.

Figure 8 shows the histograms of change in the number of shares of stock over a day. We exclude the data within $t = 5$ days because each agent rapidly changes its portfolio from the initial position, which is common to all agents. The Delta agent changes its position most frequently, while the DQN seldom changes its position. Figure 8 also suggests that the DQN agent tends to make a large change in the number of shares of stock when rebalancing its portfolio (its histogram has the heaviest tail).

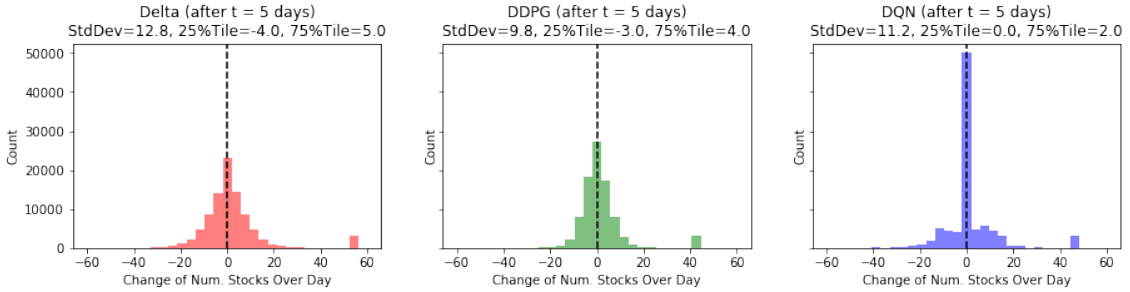


Figure 8: Histograms of change in the number of shares of stock over a day when $\alpha = 8 \times 10^{-3}$ after $t = 5$ days. The histograms are made from newly sampled 3000 paths.

Figure 9 shows the scatter plots of the position delta vs the change in the number of shares of stock over a day. The position delta is defined by the number of stock minus the option delta multiplied by 100. For instance, when the position delta is -50 [$x = -50$ in Fig. 9], an agent lacks 50 stocks of being delta neutral. The delta agent then buys 50 stocks to make the delta neutral position [$y = 50$ in Fig. 9]. In contrast, the DDPG agent tends to buy or sell smaller number of shares of stock than it would need to make its position delta neutral. The DQN agent does not show

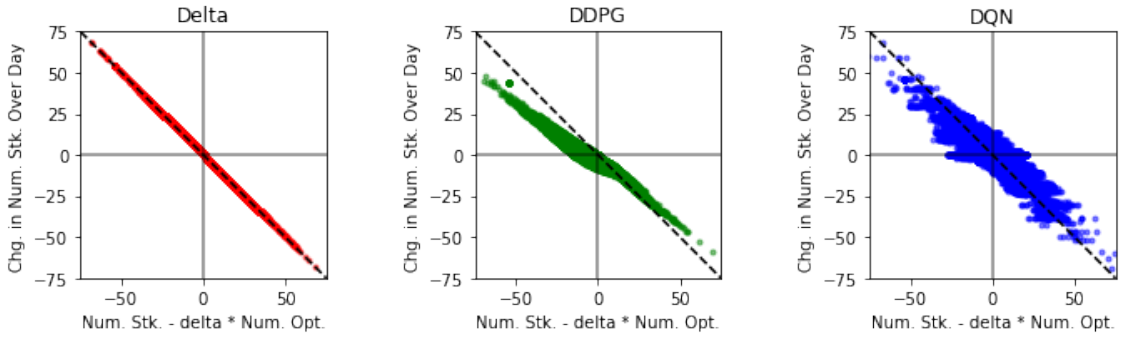


Figure 9: Scatter plots of the position delta vs the change in the number of shares of stock over a day. The position delta is defined by the number of stock minus the option delta multiplied by 100. The plots are made from newly sampled 3000 paths.

such a distinct characteristic and its scatter plot is broad. Although not very clear, sample points by the DQN agent are concentrated around $y = 0$, which reflects that the agent buys or sells stocks not frequently.

5 Summary

We have investigated option hedging strategies with the Black-Scholes-Merton (BSM) model with transaction costs, using two reinforcement learning (RL) algorithms: Deep Q Network (DQN) [6, 10] and Deep Deterministic Policy Gradient (DDPG) [8, 9]. The DQN is simple and applied to a problem with a discrete action space, while the DDPG is complicated and applied to a problem with a continuous action space. An agent hedges 100 call options with strike K and expiry T, where the underlying stock price obeys the geometric Brownian motion. We have first considered the case of no transaction costs to examine the convergence of policy. In this case, the optimal policy is delta hedging [7]. The DDPG policy is closer to the delta hedging policy than the DQN. We have then examined the cases of non-zero transaction costs. The following characteristics of the DDPG and the DQN agent have been revealed:

- The DDPG and the DQN agent tend to reduce the number of shares of stock in their positions to cut the transaction costs when the option is in the money (ITM).
- The DDPG agent tends to have a smaller number of shares of stock, in the case of not only ITM but also OTM (out of the money).
- The DQN agent tends to change a number of shares of stock not frequently.

The DQN agent determines a number of shares of stock in its position on the basis of the action-value function $q^*(s,a)$. This number (i.e., action a) must be treated as an integer even if its

value is quite large [e.g., $O(10^5)$]. It would be unrealistic to regard the action space as discrete and then the DDPG would be more practical because the DDPG algorithm regards the action space as continuous. Moreover, the DDPG agent has a smaller standard deviation of profit and loss (PL) than the DQN agent in all numerical experiments presented in this report. If we used reinforcement learning techniques in a trading desk, we would construct some weighted averaged policy from various agents including the DQN and the DDPG (i.e., model stacking). The results of this report suggest that a weight of the DDPG is larger than a weigh of the DQN, however, the DQN weight is not zero because the hedging strategy of the DQN is statistically different from that of the DDPG. Stacking agents with statistically different policies would reduce the variance of rewards and PL. The stacked agent would then determine the amount of extra charge that is added to the BSM fair price from the distribution of PL at the expiry.

References

[1] Steven Shreve. *Stochastic Calculus for Finance II: Continuous-Time Models*. Springer, 2 edition, 2008.

[2] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2 edition, 2018.

[3] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[4] Igor Halperin. QLBS: Q-Learner in the Black-Scholes(-Merton) Worlds. *arXiv e-prints*, page arXiv:1712.04609, Dec 2017.

[5] Igor Halperin. The QLBS Q-Learner Goes NuQLear: Fitted Q Iteration, Inverse RL, and Option Portfolios. *arXiv e-prints*, page arXiv:1801.06077, Jan 2018.

[6] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[7] Petter N. Kolm and Gordon Ritter. Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, 1(1):159–171, 2019.

[8] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[9] Timothy P. Lillicrap, Jonathan J. Hunt, Alexand er Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv e-prints*, page arXiv:1509.02971, Sep 2015.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[11] Emanuel Derman and Michael B. Miller. *The Volatility Smile*. Wiley, 2016.