Imperial College London

Department of Mathematics

Harvest Volatility Risk Premia using Deep Reinforcement Learning

Author: Zhihao Xu (CID: 02274977)

A thesis submitted for the degree of

MSc in Mathematics and Finance, 2022-2023

Declaration

The work contained in this thesis is my own work unless otherwise stated.

Signature: Zhihao Xu

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Prof. Vladimir Lucic, for his unwavering support, guidance and invaluable insights throughout my research journey. His expertise and guidance have been instrumental in shaping this project. In addition, I would like to express my heartfelt gratitude to my parents and my girl friend, whose love and encouragement have been the foundation of all my hard working. Their faith in me has been a source of strength.

Abstract

In this project, we focus on how to capture the Volatility Risk Premia (VRP), which can be simply described as the difference between implied and realised volatility. We outlined the two main methods of capturing VRP. One is to sell delta-hedged options strategy. Based on the Black-Scholes model, this strategy can achieve a positive PnL since the implied volatility is on average higher than the realized volatility. The second method is to sell variance swaps (Variance Swaps), which is exposed directly to the underlying volatility. This method is more direct than the former and eliminates the effect of Dollar gamma on PnL. Thanks to the static replication theorem, we can replicate variance swaps using European call and put options, making it easier and more straightforward to capture VRP. After solving the problem of how to capture VRP directly, we started to explore how to achieve best PnL within a given range of target vega leverage, based on a certain risk preferences. At this point, we used the Dueling Double Deep Q Learning (DDDQN) reinforcement learning algorithm with soft updating, since the purpose of reinforcement learning is to maximise the reward function through the agent's interaction with the environment. In the latter part of this thesis, we discuss how we accomplished our task within a reinforcement learning framework. Finally, we trained and tested on $S\&P500$ options data and achieved excellent results in the out of sample data, realizing the maintenance of returns while reducing the Max Drawdown.

Contents

List of Figures

List of Tables

Introduction

The Volatility Risk Premia(VRP) refers to two facts, one is that implied volatility is on average higher than realised volatility. The other is that higher realized volatility coincides with higher movement of underlying price. The option buyers are willing to pay an option premium to protect against a great decline in the underlying. This is a form of compensation by which the buyer mitigates the seller's significant downside risk inherent in the underlying[1]. In practice, VRP is most significant in important in large-cap equity indices such as S&P 500 [1, Ge, 2016], S&P 100 and Dow Jones Industrial [2, Carr and Wu, 2009]. In the market, there are 3 dominant methods for harvesting VRP and Ge [1, Ge, 2016] provides a detailed explanation. We will now analyze these 3 methods in the following.

- Option Strategies: One approach is to sell index options directly and hedge by buying index forward contracts which is driven by VRP and the direction of underlying movement [1, Ge, 2016]. Many research papers have examined this approach as a fundamental methodology. According to [3, Bakshi and Kapadia, 2001], this methodology leads to positive PnL empirically, especially in periods of high volatility, and leads to greater VRP. We can construct a portfolio that doesn't require considering the direction of the underlying movement. We can construct a portfolio that does not need to consider the direction of the underlying movement. This portfolio consists of selling straddles and straddle options, both of which consist of call and put options with the same expiry date. The difference is that straddle options have the same strike price while straddle options have different strike prices. Both of these portfolio strategies are essentially investing around volatility. From payoff perspective, positive PnL can be achieved when actual volatility is low and price fluctuations are minimal. In addition, when implied volatility exceeds actual volatility, options become relatively more expensive, resulting in higher premiums. However, significant losses can occur during times of high realized volatility.

  The cause of these situations lies in the difference between implied and realised volatility. We can further analyse the types of options we need to trade by examining the concept of the volatility smile. For most options, the volatility smile tends to take a 'smiling' shape. Implied volatility is relatively low when the initial underlying asset price is close to the strike price. This means that the probability of implied volatility exceeding actual volatility are relatively small. This means that there is a smaller chance for implied volatility to exceed realized volatility. Additionally, as the time to maturity becomes larger, there is a higher probability that the OTM option will become ITM, potentially requiring option sellers to make payments, and vice versa. The advantages of this strategy are simplicity, ease of trading, high liquidity, and customisability. But the disadvantages are also obvious. It cannot participate directly in volatility and the final profit depends on the difference between the price of the underlying asset and the strike price. Although in theory it converges on the difference between realised and implied volatility which is path-dependent, it is different from direct exposure to volatility.

- VIX: The VIX index is a tool introduced by CBOE in 1993 for measuring market volatility. It is calculated using the implied volatility of the at the money S&P 500 index in the near term. Trading The VIX Index is a purer volatility trading instrument, known for its good liquidity and availability of exchange trading. The downside is also clear: short-term changes

[1]For example in OTM European call option, the payoff is $[S_T - K]^+$, greater the volatility, the larger the price movements in S, which increases the likelihood of the underlying asset achieving extreme values. Consequently, the option seller is exposed to significant risk when S has particularly large positive movements but will not be paid more for S decreases deeply. Thus, option sellers will bear significant risk with large value of S.

in the VIX can be significant, so portfolios need to be constructed carefully to reduce risk. Portfolio construction of the VIX Index has become more complex over time. Similar to the drawbacks of variance swaps, it has a notional value expressed in Vega notional, which can be difficult to determine and understand.

- Swaps: There are two financial instruments, variance swaps and volatility swaps, available for us to trade in order to harvest VRP. Variance swaps are different from traditional swaps in many ways, as they do not involve periodic exchange of cash. IIt is a structured contract that specifies an initial strike price at inception and only pays out at maturity based on the difference between the realised variance (the square of realised volatility) and the strike price (the square of implied volatility) of a particular asset. Similar to other forward contracts, this strike price is chosen to ensure that the value of the contract is zero at the time of contract initiation. Similar to traditional swaps, variance swaps have a theoretical notional value that is used to calculate PnL. However, calculating and understanding the notional value of a Variance Swap can be challenging because the notional value of a Variance Swap is derived from another theoretical value, the Vega notional. The advantages of this strategy include direct exposure in the difference between realised and implied volatility (VRP), the ability to specify a notional exposure value, and not having to keep an eye on volatility at all times. Another major advantage is that, through the static replication theorem, we can replicate variance swaps with European call and put options, making them easier to trade with greater liquidity. However, the drawbacks are that it is an over-the-counter (OTC) trade that is relatively opaque, and the size of Vega notionals can be challenging, often relying on the trader's risk tolerance.

In summary, the essence of all the above methods lies in the difference between realised and implied volatility. Therefore, we conclude that VRP refers to the difference between realised and implied volatility. Thanks to the static replication theorem, we will now explore how to capture the VRP of a variance swap using simple European call and put options.

Chapter 1

Mathematical Definitions and Preliminaries

1.1 Definition and Notation

To assist readers in better understanding of this article, we will proceed to provide clear definitions and annotations.

Definition 1.1.1 (Returns)**.** Two popular ways of defining returns are arithmetic returns and logarithmic returns. Formally, for the price process S_t, arithmetic returns R_t and logarithmic returns Z_t are defined as following:

$$R_t = \frac{S_t - S_{t-1}}{S_{t-1}} = \frac{S_t}{S_{t-1}} - 1 \text{ and } Z_t = \log \frac{S_t}{S_{t-1}} = \log(1 + R_t)$$

Normally, we assume the price process follow a Geometric Brownian motion where $\frac{S_t}{S_{t-1}}$ follows independent log-normal distribution. Thus, $Z_t = \log \frac{S_t}{S_{t-1}}$ follows independent normal distribution. The more insightful mathematical connection between these two types of returns, can be explained as follows:

Using a Taylor expansion of $f(x) = log(1 + x)$ at some point x_1:

$$\begin{aligned} f(x) &= f(x_1) + f'(x_1)(x - x_1) + ... \\ \implies f(x) &\approx f(x_1) + f'(x_1)(x - x_1) \end{aligned}$$

We expand at $x_1 = 0$, we have

$$\log(1 + x) \approx x$$

$$Z_t = \log \frac{S_t}{S_{t-1}} = \log(1 + R_t) \approx R_t$$

Thus, log-return is approximate as arithmetic return when arithmetic return is close to 0 which is coincide in the real world and we will use log-returns as our return measure going forward.

Definition 1.1.2 (Implied Volatility)**.** Implied Volatility σ_{IV} refers to the expected value of future volatility of the underlying price process. This quantity is obtained based on a certain model, then input the market price and then inversely derive volatility. Meanwhile, σ_{IV} is often used to price the option with high volatility implies high premiums and vice versa.

Definition 1.1.3 (Realised Volatility)**.** Realised volatility σ_{RV} refers to the actual volatility of an asset's price over a specific period, typically defined as the standard deviation of the asset's returns with rolling windows. It's worth noting that the definitions of daily $D-\sigma_{RV}$ and annualised realised volatility $A - \sigma_{RV}$ differ.

$$D - \sigma_{RV} := \text{Var}\sqrt{(\text{Daily Return})} \text{ and } A - \sigma_{RV} := \text{Var}\sqrt{(\text{Annualised Return})}$$

There are many different ways to estimate the realised volatility, we will primarily focus on introducing three types.

- Direct: Suppose we have a rolling windows with length N, for each time t, we observe N previous iid underlying assert log-return $Z_t, Z_{t-1}, ..., Z_{t-(N-1)}$, we forecast the standard deviation of log-return based on these N previous point by an unbiased estimator as following:

  $$D-\sigma_{RV} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(Z_{t-(n-1)}-\mu)^2}$$

  where $\mu = \frac{1}{N}\sum_{n=1}^{N} Z_{t-(n-1)}$ refers to the mean of return.

  Assume that there are 252 trading days a year, we have

  $$A-\sigma_{RV} = \text{Std(Annualised Return)} = \sqrt{\text{Var}(\sum_{t=1}^{252} Z_t)}$$

  Since returns are iid:

  $$A-\sigma_{RV} = \sqrt{252\text{Var}(Z_t)} = \sqrt{252}(D-\sigma_{RV})$$

- Log-normal model: Suppose the assert price follows a log-normal model such as Geometric Brownian motion model under real world measure $\mathbb{P}$.

  $$dS_t = \mu S_t dt + D-\sigma_{RV} S_t dW_t$$

  where W_t is a Brownian motion. If we want to calibrate such model, remember that, log-return $Z_t = \log\frac{S_t}{S_{t-1}} \sim \mathbb{N}(\mu, D-\sigma_{RV})$(We will give a rigorous proof later in section Variance Swap). Thus the result is similar to direct method.

  $$D-\sigma_{RV} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(Z_{t-(n-1)}-\mu)^2}$$

  and

  $$A-\sigma_{RV} = \sqrt{252}(D-\sigma_{RV})$$

- Normal model: Suppose the assert price follows a normal model such as

  $$dS_t = \mu dt + D-\sigma_{RV} dW_t$$

  Then we have

  $$S_t - S_{t-1} \sim \mathbb{N}(\mu, D-\sigma_{RV})$$

  This model make a bit different since we have

  $$D-\sigma_{RV} = \sqrt{\text{Var}(S_t - S_{t-1})}$$

  Thus an unbiased estimator for $D-\sigma_{RV}$ is given by

  $$D-\sigma_{RV} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}((S_{t-(n-1)}S_{t-1-(n-1)})-\hat{\mu})^2}$$

  where $\hat{\mu} := \frac{1}{N}\sum_{n=1}^{N}(Z_{t-(n-1)} - Z_{t-1(n-1)})$.

  For the sake of convenience and close alignment with reality, we choose the first and second methods to estimate realized volatility.

1.2 Project Structure

Our project is divided into three main chapters. The first part describes how we obtain VRP using certain financial instruments such as delta-hedged options and variance swaps. it also elaborates how to replicate variance swaps using the static replication theorem. The second part explains how to use Deep Q learning(DQN) to harvest VRP while reducing the maximum retracement. It also describes Dueling Double Deep Q learning(DDDQN), an upgraded version of DQN. In the last section, we test the accuracy of the algorithm by letting DDDQN make predictions on BS-delta for the purpose of testing the accuracy of the algorithm, and then also backtest the S&P 500 option data and evaluate the pros and cons of our model by some performance metrics.

1.3 Feynman-Kac Theorem

The Feynman-Kac Theorem allows us to view the solution of a parabolic partial differential equation PDE, like the Black-Scholes PDE, through the lens of expected values derived from a stochastic diffusion process.

Theorem 1.3.1 (Feynman-Kac Theorem)**.** *Given suitable regularity and integrability conditions, the solution of the PDE*

$$\frac{\partial V}{\partial t}(t,x)+\frac{\partial V}{\partial x}(t,x)b(x)+\frac{1}{2}\frac{\partial^2 V}{\partial x^2}(t,x)\sigma^2(x)=rV(t,x), \quad V(T,x)=f(x),$$

can be expressed as

$$V(t,x)=e^{-r(T-t)}\mathbb{E}_{t,x}^{Q}\left\{f\left(X_T\right)\mid\mathcal{F}_t\right\}$$

where the diffusion process X has dynamics starting from x at time t

$$dX_s=b\left(X_s\right)ds+\sigma\left(X_s\right)dW_s^Q, s\geq t, X_t=x$$

where under the expectation $\mathbb{E}_{t,x}^{Q}\{\cdot\}$ is taken under the probability measure $\mathbb{Q}$. The process W^Q is a standard Brownian motion under $\mathbb{Q}$.

- A sufficient condition is:
$$\mathbb{E}\left[\int_0^T D_s^2 V_x^2\sigma^2\left(s,X_s\right)ds\right]<\infty$$
where $D_s:=e^{-\int_t^s r(u,X_u)du}$ and $V_x:=\frac{\partial V(t,x)}{\partial x}$. One possible condition is we can impose that $r(t,x)$ is bounded from below, V_x is bounded and $\sigma(t,x)$ has some suitable growth condition. There are more possible approaches can be found in [4, Karatzas and Shreve, 1988].

1.4 Girsanov's theorem

Theorem 1.4.1 (Girsanov's theorem)**.** *Let $\{W_t\}$ be a Wiener process defined on the filtered Wiener probability space $\{\Omega,\mathcal{F},\{\mathcal{F}_t^W\}_{t=0}^{\infty},\mathbb{P}\}$ with the natural filtration of the Wiener process $\left\{\mathcal{F}_t^W\right\}_{t=0}^{\infty}$. Let X_t be a process adapted to $\left\{\mathcal{F}_t^W\right\}_{t=0}^{\infty}$ and define an exponential martingale*

$$\mathcal{E}(X)_t=L_t:=\exp\left(X_t-\frac{1}{2}[X]_t\right),$$

where $[X]_t$ denotes the quadratic variation of the process X. We can easily check that $E^{\mathbb{P}}[L_t]=1$ and L_t is a positive(Equivalent Measure) martingale with a suitable condition have been satisfied.

Thus, a probability measure $\mathbb{Q}$ can be defined on $\{\Omega,\mathcal{F}\}$ such that Radon-Nikodym derivative

$$\left.\frac{d\mathbb{Q}}{d\mathbb{P}}\right|_{\mathcal{F}_t}=\mathcal{E}(X)_t$$

Then for each t the measure $\mathbb{Q}$ restricted to the $\mathcal{F}_t^W$ is equivalent to $\mathbb{P}$ restricted to $\mathcal{F}_t^W$, i.e $\mathbb{Q}\sim\mathbb{P}$ by the fact that Radon-Nikodym is postitive. Furthermore if Y_t is a local martingale under P then the process

$$\tilde{Y}_t=Y_t-[Y,X]_t$$

is a $\mathbb{Q}$ local martingale on the filtered probability space $\{\Omega,\mathcal{F},\{\mathcal{F}_t^W\}_{t=0}^{\infty},\mathbb{Q}\}$.

The primary concept of Girsanov's theorem is to explore the behaviour of the semimartingales after change of measure.

Corollary 1.4.2. *If X_t is continuous and adapted and W_t is Brownian motion under measure $\mathbb{P}$ then*

$$\tilde{W}_t = W_t - [W,X]_t$$

is Brownian motion under $\mathbb{Q}$.

Proof.

- $[W,X]_t$ is cross variation between W and X. We know that cross variation is right continuous and finite variation, thus, $\tilde{W}_t$ is right continuous.
- $[\tilde{W},\tilde{W}]_t = [W,W]_t = t$ by an easy corollary of cross variation of continuous function and finite variation function equals 0.(Proof is in the appendix)
- By Girsanov's theorem, $\tilde{W}_t$ is a $\mathbb{Q}$ local martingale. Thus by levy's characterization, $\tilde{W}_t$ Brownian motion under $\mathbb{Q}$.

□

1.5 The Black-Scholes model

In this chapter, we aim to introduce the well-known Black-Scholes model for pricing theory especially European option.

Theorem 1.5.1. *Define a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_n\}_{n=0}^{\infty}, \mathbb{P})$. Assume that the underlying stock price S_t follows a Geometric Brownian Motion such that*

$$dS_t = \mu S_t dt + \sigma S_t dW_t^{\mathbb{P}}$$

where $W_t^{\mathbb{P}}$ is a standard Brownian Motion under measure $\mathbb{P}$ and μ is drift. Assume a constant risk free rate r such that depositing 1 unit of money in the bank at time $t = 0$ will yield $B_t = \exp(rt)$ at time t.

In order to satisfy the fundamental theory of option pricing, the absence of arbitrage is equivalent to the existence of an equivalent measure $\mathbb{Q}$, We have to find the $\mathcal{E}(X)_t$ such that underlying process over a numeraires is a martingale under measure $\mathbb{Q}$, i.e $\frac{S_t}{B_t}$.

We can easily see that $\frac{S_t}{B_t}$ is a martingale if

$$dS_t = rS_t dt + \sigma S_t dW_t^{\mathbb{Q}}$$

where $W_t^Q := W_t^{\mathbb{P}} + \frac{\mu - r}{\sigma} t$ such that S_t follows Geometric Brownian motion under measure $\mathbb{P}$.

By Corollary 1.4.2, W_t^Q is a Brownian motion if

$$[W,X]_t = -\frac{\mu - r}{\sigma} t \Rightarrow X_t = \int_0^t \frac{r-\mu}{\sigma} dW_t$$

where we have used the Kunita-Watana property.

Thus, we find the risk neutral measure $\mathbb{Q}$ such that Radon-Nikodym derivative

$$\left.\frac{d\mathbb{Q}}{d\mathbb{P}}\right|_{\mathcal{F}_t} = \mathcal{E}\left(\int_0^t \frac{r-\mu}{\sigma} dW_t\right)$$

Denote the value of the call option at time t as C_t with payoff $\left[(S_T - K)^+\right]$. by no arbitrage theorem,

$$C_t = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}\left(S_T - K\right)^+ | \mathcal{F}_t\right]$$

thus we can say C_t is a function of S_t and t since S_t is a markov process, i.e $C_t := C(S_t, t)$.

Assume the function $C(S_t, t)$ of time t and of the stock price S_t to have regularity $C \in C^{1,2}\left([0,T] \times \mathbb{R}^+\right)$. Apply Ito's Lemma to C to obtain

$$dC\left(t, S_t\right) = \left(\frac{\partial C}{\partial t}\left(S_t, t\right) + \mu S_t \frac{\partial C}{\partial S}\left(S_t, t\right) + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 C}{\partial S^2}\left(S_t, t\right)\right) dt + \sigma S_t \frac{\partial C}{\partial S}\left(S_t, t\right) dW_t \quad (1.5.1)$$

We consider a self-financing trading strategy for $0 \leq t \leq T$,

$$\phi_t^S = \frac{\partial C}{\partial S}(S_t, t), \quad \phi_t^B = \left(C_t - \phi_t^S S_t\right)/B_t$$

By construction, the value of this strategy at time t is C_t, since clearly $C(S_t, t) = \phi_t^B B_t + \phi_t^S S_t$. Thus,

$$dC_t = \phi_t^B dB_t + \phi_t^S dS_t = \left[C(S_t, t) - \frac{\partial C}{\partial S}(S_t, t) S_t\right] r dt + \frac{\partial C}{\partial S}(S_t, t) S_t (\mu dt + \sigma dW_t) \qquad (1.5.2)$$

Then by equating 1.5.1 and 1.5.2 (ITO + SELF FINANCING), we obtain the BS valuation equation:

$$\frac{\partial C}{\partial t}(S_t, t) + r S_t \frac{\partial C}{\partial S}(S_t, t) + \sigma^2 S_t^2 \frac{1}{2} \frac{\partial^2 C}{\partial S^2}(S_t, t) = r C(S_t, t) \qquad (1.5.3)$$

with terminal condition $C(S_T, T) = (S_T - K)^+$.

One can deduced that the solution of BS valuation equation 1.5.3 is

$$C(S, t) = S_t N(d_1) - K e^{-r(T-t)} N(d_2)$$

with

$$\begin{aligned} d_1 &= \frac{1}{\sigma\sqrt{T-t}}\left[\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right] \\ d_2 &= d_1 - \sigma\sqrt{T-t} \end{aligned}$$

where N is the standard cumulative normal distribution function:

$$N(x) = \mathbb{P}(X \leq x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{x^2}{2}} dx$$

By relationship of put-call parity for European options, the corresponding put option price deduced as follows:

$$\begin{aligned} P(S_t, t) &= K e^{-r(T-t)} - S_t + C(S_t, t) \\ P(S_t, t) &= K e^{-r(T-t)} N(-d_2) - S_t N(-d_1) \end{aligned}$$

1.5.1 Greeks

Definition 1.5.2 (Greeks)**.** In the realm of options trading, the term "Greeks" encompasses a collection of mathematical metrics or indicators. These metrics serve to elucidate how an option's price is expected to fluctuate in reaction to a variety of factors. These factors basically encompass alterations in the underlying asset's value (S_t), the passage of time (t), shifts in volatility (σ), and changes in interest rates (r).

Thus, greeks under Black-Scholes model can be found by taking the partial derivative of the option price w.r.t. to the corresponding components.

- BS Delta(Δ): The deltas of calls and puts which represent the sensitivity of an option price to changes in the price of underlying assert are shown as follows:

$$\begin{aligned} \Delta_{\text{call}} &= \frac{\partial C(S, t)}{\partial S} = N(d_1) \\ \Delta_{\text{put}} &= \frac{\partial P(S, t)}{\partial S} = N(d_1) - 1 = \Delta_{\text{call}} - 1 \end{aligned}$$

- BS Vega(ν): The vegas of calls and puts which measures the sensitivity of an option's price to changes in volatility are shown as follows:

$$\begin{aligned} \nu_{\text{call}} &= \frac{\partial C(S, t)}{\partial \sigma} = S\sqrt{T-t} N'(d_1) \\ \nu_{\text{put}} &= \frac{\partial P(S, t)}{\partial \sigma} = S\sqrt{T-t} N'(d_1) = \nu_{\text{call}} \end{aligned}$$

- BS Gamma(Γ): The gammas calls and puts of which represent changes of deltas concerning changes in the underlying asset's price are shown as follows:

$$\begin{aligned}
\Gamma_{\text{call}} &= \frac{\partial \Delta_{\text{call}}}{\partial S} = \frac{N'(d_1)}{S\sigma\sqrt{T-t}} \\
\Gamma_{\text{put}} &= \frac{\partial \Delta_{\text{put}}}{\partial S} = \frac{N'(d_1)}{S\sigma\sqrt{T-t}} = \Gamma_{\text{call}}
\end{aligned}$$

- BS Theta(Θ): The gammas calls and puts of which represents the sensitivity of an option's price to the time to maturity τ are shown as follows:

$$\begin{aligned}
\Theta_{\text{call}} &= \frac{\partial C(S,t)}{\partial \tau} = -\frac{\sigma S N'(d_1)}{2\sqrt{\tau}} - rKe^{-r\tau}N\left(d_2\right) \\
\Theta_{\text{put}} &= \frac{\partial P(S,t)}{\partial \tau} = -\frac{\sigma S N'(d_1)}{2\sqrt{\tau}} + rKe^{-r\tau}N\left(-d_2\right)
\end{aligned}$$

Chapter 2

Harvest VRP by Delta-Hedged Variance Swap

2.1 VRP harvesting via selling Delta-Hedged portfolio

In this section, we will clarify the methodology for acquiring VRP through selling Delta-Hedged portfolio. Before diving into the specifics, we must clarify two key questions. Firstly, which theoretical framework is our methodology based on? Secondly, in calculating delta ($\Delta = N(d_1)$), do we use actual or implied volatility? For convenience, we will use the Black-Scholes (BS) model and make a distinction between using implied and actual volatility to calculate delta.

2.1.1 Trading liquid option and Hedging with actual volatility

In order to formalize the main idea, we denote the option price $C(S_t, t)$ and stock price S_t and Delta-Hedged portfolio as $C(S_t, t) - \phi_t^B B_t - \phi_t^S S_t$ at time t . where

$$\phi_t^S = \frac{\partial C}{\partial S}(S_t, t) = \Delta_t, \quad \phi_t^B = \left(C_t - \phi_t^S S_t\right)/B_t$$

The gain process of this portfolio from time t to $t+\tau$ $G_{t,t+\tau}$ with cash earns the risk-free rate r is defined as

$$G_{t,t+\tau} := C_{t+\tau} - C_t - \int_t^{t+\tau} \Delta_u dS_u - \int_t^{t+\tau} r\left(C_u - \Delta_u S_u\right) du \tag{2.1.1}$$

where r is the constant risk-free rate.

Assume the stock price follow the geometric Brownian Motion under the real world probability measure $\mathbb{P}$:

$$dS_t = \mu S_t dt + \sigma S_t dW_t^{\mathbb{P}}$$

Apply ito's formula to option price $C(S_t, t)$, we can derive following equation:

$$\begin{aligned} dC_t &= \frac{\partial C_t}{\partial S_t} dS_t + \left(\frac{\partial C_t}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 C_t}{\partial S_t^2}\right) dt \\ C_{t+\tau} - C_t &= \int_t^{t+\tau} \frac{\partial C_u}{\partial S_u} dS_u + \int_t^{t+\tau} \left(\frac{\partial C_u}{\partial u} + \frac{1}{2}\sigma^2 S_u^2 \frac{\partial^2 C_u}{\partial S_u^2}\right) du \end{aligned} \tag{2.1.2}$$

Notice that option price satisfies the BS valuation equation,

$$\frac{\partial C}{\partial t}(S_t, t) + r S_t \frac{\partial C}{\partial S}(S_t, t) + \sigma^2 S_t^2 \frac{1}{2} \frac{\partial^2 C}{\partial S^2}(S_t, t) = rC(S_t, t) \tag{2.1.3}$$

By equations 2.1.2 and 2.1.3, we deduce that

$$C_{t+\tau} - C_t = \int_t^{t+\tau} \Delta_u dS_u + \int_t^{t+\tau} r\left(C_u - \Delta_u S_u\right) du \tag{2.1.4}$$

Comparing to equation 2.1.1, it is apparent that if we continuously delta-hedged, the gain equals to 0 ($G_{t,t+\tau} = 0$) over every period. Actually, following the similar derivation, $G_{t,t+\tau} = 0$ appears up for all one-dimensional Markov price process $dS_t = \mu(S_t)S_t dS_t + \sigma(S_t)S_t dW_t^{\mathbb{P}}$.

It is noteworthy that the gain process $G_{t,t+\tau}$ is zero only under the Black-Scholes (BS) model. If under a stochastic volatility model, the $G_{t,t+\tau}$ manifests as a stochastic process. Its expected value $E(G_{t,t+\tau})$ can be interpreted as the excess return of the delta-hedged portfolio. Furthermore, it has been demonstrated that $E(G_{t,t+\tau}) = 0$ unless volatility risk is priced and symmetric in [3, Bakshi and Kapadia, 2001].

2.1.2 Trading option priced by implied volatility and Hedging with implied volatility

In this section, we aim to elucidate the nexus between harvesting VRP and delta-hedged options, which stands as our primary objective. Concurrently, we operate under the assumption that we are trading an option priced by implied volatility with value process $C_t^{(i)}$ and subsequently define the delta-hedged portfolio as Delta-Hedged portfolio as $C^{(i)} - \phi_t^{B(i)} B_t - \phi_t^{S(i)} S_t$ at time t. Thus we have [5, D. Olivier and M. Abhishek, 2022]

$$\phi_t^{S(i)} = \frac{\partial C^{(i)}}{\partial S}(S_t, t) = \Delta_t^{(i)}, \quad \phi_t^{B(i)} = \left(C_t^{(i)} - \phi_t^{S(i)} S_t\right)/B_t$$

which corresponds to buy the option and delta hedged the option using implied volatility $\sigma_{(iv)}$ while the actual volatility of the stock prices is $\sigma_{(rv)}$. Thus, we can define the pnl $\Pi_t^{(i)}$ using the self-financing condition of the trading strategy $(\phi_t^{S(i)}, \phi_t^{B(i)})$:

$$\mathrm{d}\Pi_t^{(i)} = \mathrm{d}C_t^{(i)} - \Delta_t^{(i)} \mathrm{d}S_t - r\left(C_t^{(i)} - \Delta_t^{(i)} S_t\right)\mathrm{d}t \tag{2.1.5}$$

Applying ito's formula to option price $C_t^{(i)}$, we can derive following equation:

$$\mathrm{d}C_t^{(i)} = \frac{\partial C^{(i)}}{\partial t}\,\mathrm{d}t + \underbrace{\frac{\partial C^{(i)}}{\partial S}}_{=\Delta_t^{(i)}}\mathrm{d}S_t + \frac{1}{2}\sigma_{(rv)}^2 S_t^2 \underbrace{\frac{\partial^2 C^{(i)}}{\partial S}}_{=\Gamma_t^{(i)}}\mathrm{d}t, \tag{2.1.6}$$

Combining equation 2.1.5 and 2.1.6, we have

$$\mathrm{d}\Pi_t^{(i)} = \left(\frac{\partial C^{(i)}}{\partial t} + \frac{1}{2}\sigma_{(rv)}^2 S_t^2 \frac{\partial^2 C^{(i)}}{\partial S^2} - r\left(C_t^{(i)} - \Delta_t^{(i)} S_t\right)\right)\mathrm{d}t.$$

where $dS_t = \mu S_t dt + \sigma_{(rv)} S_t dW_t^{\mathbb{P}}$. Notice that by Feynman-Kac theorem, option price $C_t^{(i)}$ satisfies the following BS valuation equation:

$$\frac{\partial C^{(i)}}{\partial t} + \underbrace{\frac{\partial C^{(i)}}{\partial S}}_{=\Delta_t^{(i)}} r S_t + \frac{1}{2}\underbrace{\frac{\partial^2 C^{(i)}}{\partial S^2}}_{=\Gamma_t^{(i)}}\sigma_{(iv)}^2 S_t^2 - rC_t^{(i)} = 0$$

Thus, we obtain

$$\mathrm{d}\Pi_t = \frac{1}{2}\left(\sigma_{(rv)}^2 - \sigma_{(iv)}^2\right) S_t^2 \Gamma^{(iv)} \mathrm{d}t$$

Over each short duration, the pnl of delta-hedged portfolio is contingent upon the discrepancy between realized and implied variance, scaled by the gamma (calculated utilizing the implied volatility). Although $d\Pi_t$ is deterministic, the cumulative hedging gain throughout the option's tenure exhibits path-dependency. Its absolute value is higher when the paths of stock fluctuate around the strike such that the gamma is higher.

As previously articulated, harvesting VRP is tantamount to the difference between implied volatility and realised volatility($\sigma_{iv} - \sigma_{rv}$). The aforementioned portfolio entails purchasing options and subsequently selling stocks. Hence, to capture VRP, we would sell options and buy stocks for delta-hedging, rendering our VRP's gain process as

$$\text{VRP}_t := \frac{1}{2}\int_0^t \Gamma_u^{(iv)} S_u^2 \left(\sigma_{(rv)}^2 - \sigma_{(iv)}^2\right) \mathrm{d}t = \int_0^t \Gamma_u^{\$(iv)} \left(\sigma_{(rv)}^2 - \sigma_{(iv)}^2\right) \mathrm{d}t$$

where we denote Dollar Gamma as $\Gamma_t^{\$(iv)} := \frac{1}{2}\Gamma_t^{(iv)} S_t^2$.

2.2 Introduction to Variance Swap

This chapter draws on the research from [6, JP Morgan, 2005] for a comprehensive study of variance swaps. We clarify and summarise the preliminary content on the topics of variance swaps and static replication, aiming to harvesting VRP through static replication of variance swap.

Definition 2.2.1 (Variance Swap)**.** The variance swap is an over-the-counter (OTC) customised financial derivative designed to speculate on or hedge against the volatility (degree of fluctuation) of a specific underlying asset (e.g., an equity index, currency or interest rate). Essentially, this instrument provides a payoff based on the difference between the realised variance of the underlying asset (square of the realised volatility) and a strike price (square of the implied volatility) determined at the start of the agreement. The payoff is given by

$$\text{Variance Swap Payoff} = \text{Variance Notional} \times (\text{Realised Volatility at maturity}^2 - \text{Strike}_{\text{Var}}^2)$$
$$\text{Volatility Swap Payoff} = \text{Volatility Notional} \times (\text{Realised Volatility at maturity} - \text{Strike}_{\text{Vol}})$$

where variance notional is defined as:

$$\text{Variance Notional} = \frac{\text{Vega Notional(USD)}}{2 \times \text{Strike}}$$

such that the realsied volatility is 1 'vega' above the strike at maturity, the payoff is approximately equal Vega Notional by

$$\frac{\text{Vega Notional(USD)}}{2 \times \text{Strike}} \times ((\text{Strike} + 1)^2 - \text{Strike}^2) \approx \text{Vega Notional(USD)}$$

We can easily see $\text{VS}_{\text{payoff}}$ is convex on Realised Volatility at maturity, as illustrated in Figure 2.1. It means the investor who longs a variance swap will have faster gain and slower loss.

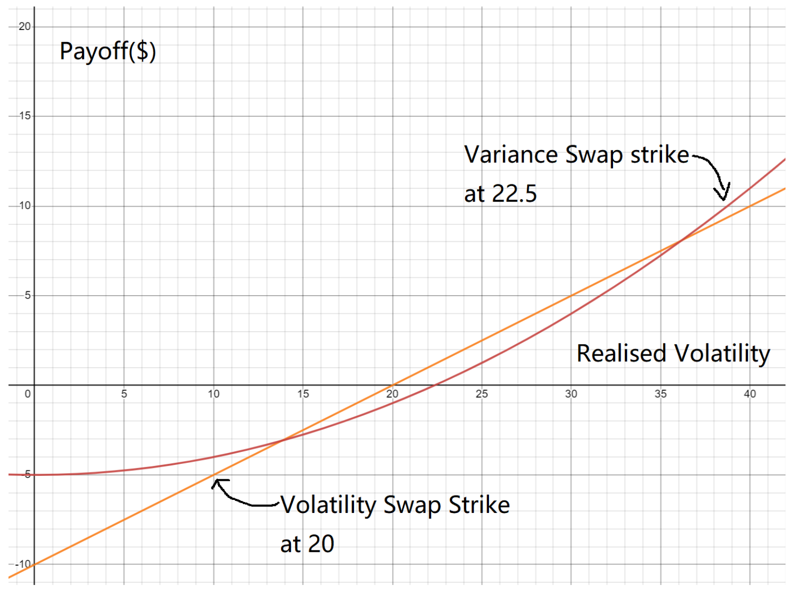


Figure 2.1: Payoff of Variance Swap and Volatility Swap with y-axis payoff($)

where we can see the Variance Swap strike is larger than volatility Swap strike by the Jensen's inequality(concave) as following:

$$\text{Strike}_{\text{Vol}} = E\left[\sqrt{\text{Realised Volatility}^2}\right] <= \sqrt{\left(E\left[\text{Realised Volatility}^2\right]\right)} = \text{Strike}_{\text{Var}}$$

2.2.1 Relationship between Variance Swap payoff and delta-hedged Pnl

Denote that the option price as a function of S, t and σ, i.e $C(S, t, \sigma)$. By applying ito's formula, we have:

$$\begin{aligned} dC &= \frac{\partial C}{\partial S}dS + \frac{\partial C}{\partial t}dt + \frac{1}{2}\frac{\partial^2 C}{\partial S^2}dS^2 + \frac{\partial C}{\partial \sigma}d\sigma + \dots \\ &= \Delta dS + \Theta dt + \frac{1}{2}\Gamma dS^2 + \nu d\sigma + \dots \\ &= \text{ Delta PnL } + \text{ Theta PnL } + \text{ Gamma PnL } + \text{ Vega PnL } + \dots \end{aligned}$$

Consider the portfolio Π consisting of delta-hedged option. Thus

$$\Pi = C - \frac{\partial C}{\partial S}S = C - \Delta S$$

By self-financing condition of portfolio and applying ito's formula to option C, we have

$$\begin{aligned} d\Pi = dC - \Delta dS &= \Theta dt + \frac{1}{2}\Gamma dS^2 + \nu d\sigma + \dots \\ &= \text{ Theta PnL } + \text{ Gamma PnL } + \text{ Vega PnL } + \dots \end{aligned}$$

By assuming the volatility is constant like BS model and risk free interest rate equals 0, we can express the Daily Pnl of delta-hedged portfolio $d\Pi$ as

$$d\Pi = \Theta dt + \frac{1}{2}\Gamma dS^2 \tag{2.2.1}$$

As shown in 1.5.3, any contingent claims with value $f(S, t)$ at time t is the function of underlying S, satisfy the BS valuation equation also linear combination of them. Then

$$r\Pi = \Theta + r\Delta S + \frac{1}{2}\Gamma\sigma^2 S^2$$

since from stochastic differential equation 2.2.1 of Π, we have

$$\frac{\partial \Pi}{\partial t} = \Theta \text{ and } \frac{\partial^2 \Pi}{\partial S^2} = \Gamma$$

Meanwhile, we have $\frac{\partial \Pi}{\partial S} = 0$ from equation 2.2.1, thus

$$\Theta = r\Pi - \frac{1}{2}\Gamma\sigma^2 S^2$$

By assuming risk free interest rate equals 0 as before, then

$$\Theta = -\frac{1}{2}\Gamma\sigma^2 S^2 \tag{2.2.2}$$

By substituting equation 2.2.2 into 2.2.1, then

$$\begin{aligned} d\Pi &= \frac{1}{2}\Gamma(dS^2 - \sigma^2 S^2 dt) \\ &= \frac{1}{2}\Gamma S^2 \times \left[\left(\frac{dS}{S}\right)^2 - \sigma^2 dt\right] \end{aligned} \tag{2.2.3}$$

Now, we partition both the t axis and the S axis, with step sizes of $\Delta t = 1$ and daily changes of S ΔS respectively. Thus, the Daily Pnl of delta-hedged portfolio is given by

$$\text{Daily PnL} = \frac{1}{2}\Gamma S^2 \times \left[\left(\frac{\Delta S}{S}\right)^2 - \sigma^2 \Delta t\right] \tag{2.2.4}$$

where we can interpret $\frac{\Delta S}{S}$ as the daily stock return and square it can be seen as daily squared realised volatility. Furthermore, $\sigma^2 \Delta t$ can be interpreted as the daily squared implied volatility. Therefore, from equation 2.2.4, it can be inferred that the delta-hedged daily return is driven by the difference between the squared realized volatility and the squared implied volatility.

Now we sum up all the daily PnL till the maturity of the option, we obtain the total PnL as following:

$$\text{Total PnL} = \sum_{t=0}^{n} \Gamma_t^{\$} \times \left[r_t^2 - \sigma^2 \Delta t\right] \tag{2.2.5}$$

where $r_t := \left(\frac{\Delta S}{S}\right)^2$ is the stock daily return and $\Gamma_t^{\$} := \frac{1}{2}\Gamma_t S_t^2$ is the dollar Gamma.

Equation 2.2.5 closely mirrors the payoff structure of a variance swap, being dollar Gamma weighted sum of the difference between squared realized returns and a constant strike. However, a key distinction lies in the weighting. In a variance swap, these weights remain consistent, while in this context, they vary based on the option's dollar gamma over its duration which is called path-dependency.

2.2.2 Constant Dollar Gamma portfolio construction

In formula 2.2.5, it prompts us to consider whether we can apply a certain scaling ratio to our delta-hedged portfolio to ensure that the dollar gamma value of the options in the portfolio remains in time (t) and stock price (S) such that the PnL of delta-hedged portfolio has constant exposure to squared realised volatility. Next, we will explore the feasibility of scaling by plotting the curve of dollar gamma with respect to t and S. A rigorous mathematical proof will be provided at the end of this discussion. The Dollar Gamma across strikes and underlying is given by Figure 2.2 where we can see that the peak dollar gamma exhibits an increasing relationship with the strike. Furthermore, options with lower strikes contribute minimally compared those with higher strikes. Thus, it's essential to amplify the weights of options with lower strikes while reducing the weights of those with higher strikes.

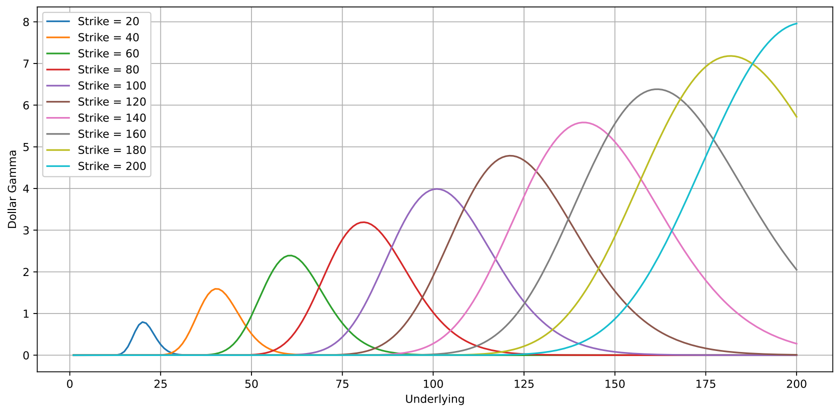


Figure 2.2: The Dollar Gamma with different strike and underlying

The simplest scaling involves multiplying by $\frac{1}{K}$ which yield the following graph 2.3 where we can see that with the escalation in strike, there's an amplification in the spread of Dollar Gamma.

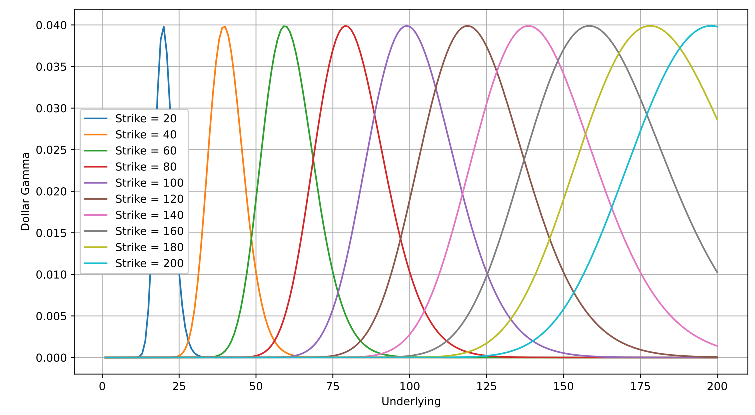


Figure 2.3: The Dollar Gamma weighted by $\frac{1}{K}$

The $\frac{1}{K^2}$ weighted sum is given by Figure 2.4 which results in constant Dollar Gamma.

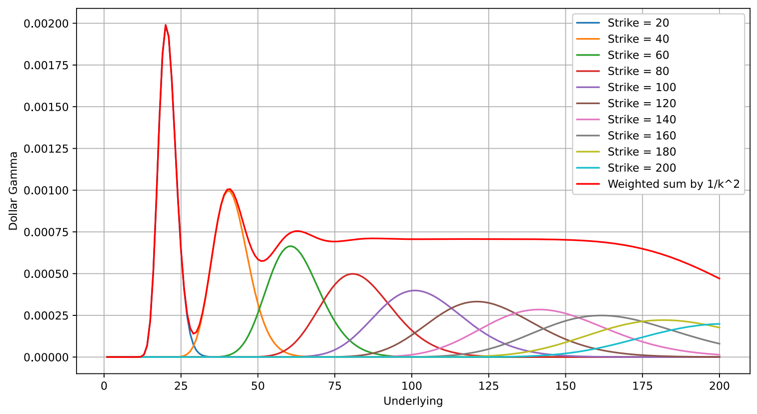


Figure 2.4: The Dollar Gamma weighted by $\frac{1}{K^2}$

The weighted parameters $\frac{1}{K^2}$ is motivated by finding a portfolio with value process $\Pi(S,t)$ such that

$$\Gamma = \frac{\partial^2 \Pi}{\partial S^2} = \frac{c}{S^2}$$

Then Dollar Gamma becomes constant

$$\Gamma^{\$} = \frac{c}{S^2} S^2 = c$$

By Solving above second order differential equation, we get:

$$\pi(S,t) = -a\ln(S_t) + bS_t + c \tag{2.2.6}$$

where a, b and c are constant. Thus, the replication of Variance Swap is

While we cannot trade log contract directly in the market, thankfully due to static replication theory, we can replicate it using option strips.

From the equation 2.2.6, it can be inferred that in order for our PnL to have a constant exposure to volatility, we need to construct a portfolio consisting of a log-contract, the underlying stock, and cash. We will later demonstrate that the log contract can actually be replicated by options such that the replication is a direct exposure to realised variance. Meanwhile, the position of underlying stock essentially replicates the stock part in the delta-hedged portfolio, providing exposure to implied variance. A more clear representation is provided below.

$$\text{Variance Swap Payoff} = \text{Variance Notional} \times (\underbrace{\text{Realised Volatility at maturity}^2}_{\text{log contract}} - \underbrace{\text{Strike}^2_{\text{Var}}}_{\text{stock}})$$

$$\text{Delta-hedged payoff} = \underbrace{C}_{\text{log contract}} - \underbrace{\Delta S}_{\text{stock}}$$

Subsequently, we will delve into static replication theory and provide a rigorous proof for the replication of variance swap.

2.2.3 Static Replication of Variance Swap by option strips

Theorem 2.2.2 (Static Replication)**.** *A European options with payoff of $g(S_T)$ can be replicated using the underlying asset S, cash, and European Calls and Puts option strips. Furthermore, this replication is static in nature and does not necessitate position rebalancing.*

Let $g : \mathbb{R} \to \mathbb{R}$ be a $\mathcal{C}^2$ function, H a non-negative constant, $P(S_t, K, T)$ and $C(S_t, K, T)$ are European Puts and Calls option at time t with underlying S,strike K and expiry time T and define a filtered risk netural probability space $(\Omega, \mathcal{F}, \{\mathcal{F}_n\}_{n=0}^{\infty}, \mathbb{Q})$, the payoff $g(S_T)$ satisfies

$$\begin{aligned} e^{-r(T-t)} E^{\mathbb{Q}}\left[g\left(S_T\right) \mid \mathcal{F}_t\right] &= e^{-r(T-t)} g(H) + e^{-r(T-t)} g'(H)\left(E[S_T] - H\right) \\ &\quad + \int_0^H g''(K) P\left(S_t, K, T\right) dK + \int_H^{\infty} g''(K) C\left(S_t, K, T\right) dK \end{aligned} \tag{2.2.7}$$

Before proceeding with the proof, our preliminary task is to determine the first and second order derivatives of the European option with respect to the strike, and understand their implications.

Proposition 2.2.3. *Suppose we are pricing the European call and put option with payoff $(S_T - K)^+$ and $(K - S_T)^+$ respectively. Suppose $\delta(x)$ is dirac delta function and $f_{S_T|\mathcal{F}_t}$ is conditional pdf of random variable S_T.*

$$e^{-r(T-t)} f_{S_T|\mathcal{F}_t}(K) = \frac{\partial^2 P\left(S_t, K, T\right)}{\partial K^2} = \frac{\partial^2 C\left(S_t, K, T\right)}{\partial K^2}$$

Proof. Without loss of generality, we prove the following using European call option $C(S_T, K, T)$.

By dominated convergence theorem,

$$\begin{aligned} \frac{\partial C(S_t, K, T)}{\partial K} &= e^{-r(T-t)} E^{\mathbb{Q}}\left[\partial \frac{(S_T - K)^+}{\partial K} | \mathcal{F}_t\right] \quad K > 0 \\ &= e^{-r(T-t)} E^{\mathbb{Q}}\left[-\mathbb{1}_{K \le S_T} | \mathcal{F}_t\right] \\ &= -e^{-r(T-t)} \mathbb{P}(S_T \ge K | \mathcal{F}_t) \\ \frac{\partial P(S_t, K, T)}{\partial K} &= e^{-r(T-t)} E^{\mathbb{Q}}\left[\partial \frac{(K - S_T)^+}{\partial K} | \mathcal{F}_t\right] \quad K > 0 \\ &= e^{-r(T-t)} E^{\mathbb{Q}}\left[\mathbb{1}_{K \ge S_T} | \mathcal{F}_t\right] \\ &= e^{-r(T-t)} \mathbb{P}(S_T \le K | \mathcal{F}_t) \\ \frac{\partial^2 C(S_t, K, T)}{\partial K^2} &= e^{-r(T-t)} E^{\mathbb{Q}}\left[\partial^2 \frac{(S_T - K)^+}{\partial K^2} | \mathcal{F}_t\right] \quad K > 0 \\ &= e^{-r(T-t)} E^{\mathbb{Q}}\left[\partial \frac{-\mathbb{1}_{K \le S_T}}{\partial K} | \mathcal{F}_t\right] \\ &= e^{-r(T-t)} \int \delta(x - K) f_{S_T|\mathcal{F}_t}(x) dx \\ &= e^{-r(T-t)} f_{S_T|\mathcal{F}_t}(K) \\ \frac{\partial^2 P(S_t, K, T)}{\partial K^2} &= e^{-r(T-t)} E^{\mathbb{Q}}\left[\partial^2 \frac{(K - S_T)^+}{\partial K^2} | \mathcal{F}_t\right] \quad K > 0 \\ &= e^{-r(T-t)} E^{\mathbb{Q}}\left[\partial \frac{\mathbb{1}_{K \ge S_T}}{\partial K} | \mathcal{F}_t\right] \\ &= e^{-r(T-t)} \int \delta(x - K) f_{S_T|\mathcal{F}_t}(x) dx \\ &= e^{-r(T-t)} f_{S_T|\mathcal{F}_t}(K) \end{aligned} \tag{2.2.8}$$

□

where we have defined the derivative in the sense of distribution since the derivative in the usual sense doesn't exist at discontinue point.

Now we begin to prove the static replication theorem by above proposition 2.2.3.

Proof. Denote $f_{S_T|\mathcal{F}_t}(K)$ as the condition probability density function of S_T, then by proposition 2.2.3

$$e^{-r(T-t)}f_{S_T|\mathcal{F}_t}(K) = \frac{\partial^2 P(S_t,K,T)}{\partial K^2} = \frac{\partial^2 C(S_t,K,T)}{\partial K^2}$$

Thus by the martingale pricing theorem, we deduce that the payoff $g(S_T)$ of European option satisfies:

$$\begin{aligned}
e^{-r(T-t)}E^{\mathbb{Q}}\left[g(S_T) \mid \mathcal{F}_t\right] &= e^{-r(T-t)}\int_0^\infty g(K)f_{S_T|\mathcal{F}_t}(K)dK \\
&= e^{-r(T-t)}\int_0^\infty g(K)e^{r(T-t)}\frac{\partial^2 P(S_t,K,T)}{\partial K^2}dK \\
&= \int_0^\infty g(K)\frac{\partial^2 P(S_t,K,T)}{\partial K^2}dK \\
&= \mathbb{1}_{S_T\le H}\int_0^H g(K)\frac{\partial^2 P(S_t,K,T)}{\partial K^2}dK + \mathbb{1}_{S_T\ge H}\int_H^\infty g(K)\frac{\partial^2 C(S_t,K,T)}{\partial K^2}dK
\end{aligned}$$

where H is a non-negative constant. Integrating by parts yields:

$$\begin{aligned}
e^{-r(T-t)}E^{\mathbb{Q}}\left[g(S_T) \mid \mathcal{F}_t\right] = \mathbb{1}_{S_T\le H}&\left(g(K)\frac{\partial P(S_t,K,T)}{\partial K}\bigg|_0^H - \int_0^H g'(K)\frac{\partial P(S_t,K,T)}{\partial K}dK\right) \\
+ \mathbb{1}_{S_T\ge H}&\left(g(K)\frac{\partial C(S_t,K,T)}{\partial K}\bigg|_H^\infty - \int_H^\infty g'(K)\frac{\partial C(S_t,K,T)}{\partial K}dK\right)
\end{aligned}$$

By the calculation of first order derivative in equation 2.2.8, then

$$e^{-r(T-t)}E^{\mathbb{Q}}\left[g(S_T) \mid \mathcal{F}_t\right] = e^{-r(T-t)}g(H) - \int_0^H g'(K)\frac{\partial P(S_t,K,T)}{\partial K}dK - \int_H^\infty g'(K)\frac{\partial C(S_t,K,T)}{\partial K}dK$$

Applying integration by parts again to yield:

$$\begin{aligned}
e^{-r(T-t)}E^{\mathbb{Q}}\left[g(S_T) \mid \mathcal{F}_t\right] = e^{-r(T-t)}g(H) &- g'(K)P(S_t,K,T)\big|_0^H + \int_0^H g''(K)P(S_t,K,T)\,dK \\
&- g'(K)C(S_t,K,T)\big|_H^\infty + \int_H^\infty g''(K)C(S_t,K,T)\,dK
\end{aligned}$$

Finally, $S_t > 0$ almost surely under BS model, then

$$\begin{cases}
P(S_t,K,T) = e^{-r(T-t)}E^{\mathbb{Q}}\left[(0-S_T)^+ \mid \mathcal{F}_t\right] = 0 \text{ if } K = 0 \\
C(S_t,K,T) = e^{-r(T-t)}E^{\mathbb{Q}}\left[(S_T-\infty)^+ \mid \mathcal{F}_t\right] = 0 \text{ if } K = \infty
\end{cases}$$

Recall that the put call parity for European option as following:

$$P(S_t,H,T) + e^{-r(T-t)}E[S_T] = C(S_t,H,T) + e^{-r(T-t)}H$$

Thus,

$$\begin{aligned}
e^{-r(T-t)}E^{\mathbb{Q}}\left[g(S_T) \mid \mathcal{F}_t\right] &= e^{-r(T-t)}g(H) + e^{-r(T-t)}g'(H)\left(E[S_T]-H\right) \\
&+ \int_0^H g''(K)P(S_t,K,T)\,dK + \int_H^\infty g''(K)C(S_t,K,T)\,dK
\end{aligned}$$

□

This formula indicates that any contingent claim with a payoff $g(S_T)$ $g:\mathbb{R}\to\mathbb{R}$ a $\mathcal{C}^2$ function, can be replicated using cash, the underlying asset S, and an infinite number of out-of-the-money(OTM) European call and put options strips with different strike prices.

A simpler approach to this proof is to employ the Carr-Madan-formula, which allows us to avoid the tedious integration by parts.

Lemma 2.2.4 (Carr-Madan-formula)**.** *Let $g:\mathbb{R}\to\mathbb{R}$ be a $\mathcal{C}^2$ function, H a non-negative constant,*

$$g(S) = g(H) + g'(H)(S-H) + \mathbb{1}_{\{S>H\}}\int_H^\infty g''(u)(S-u)_+ \mathrm{d}u + \mathbb{1}_{\{S<H\}}\int_0^H g''(u)(u-S)_+ \mathrm{d}u$$

Proof. For $S>0$ and non-negative constant H, by the fundamental theorem of calculus

$$g(S) - g(H) = \mathbb{1}_{\{S>H\}}\int_H^S g'(u)\mathrm{d}u \;-\; \mathbb{1}_{\{S<H\}}\int_S^H g'(u)\mathrm{d}u$$

Then

$$\begin{aligned}
g(S) &= g(H) + g(S) - g(H)\\
&= g(H) + \mathbb{1}_{\{S>H\}}\int_H^S g'(u)\mathrm{d}u \;-\; \mathbb{1}_{\{S<H\}}\int_S^H g'(u)\mathrm{d}u\\
&= g(H) + \mathbb{1}_{\{S>H\}}\int_H^S \left[g'(H) + \int_H^u g''(v)\mathrm{d}v\right]\mathrm{d}u - \mathbb{1}_{\{S<H\}}\int_S^H \left[g'(H) - \int_u^H g''(v)\mathrm{d}v\right]\mathrm{d}u\\
&= g(H) + g'(H)(S-H) + \mathbb{1}_{\{S>H\}}\int_H^S\int_v^S g''(v)\mathrm{d}u\mathrm{d}v + \mathbb{1}_{\{S<H\}}\int_S^H\int_S^v g''(v)\mathrm{d}v\mathrm{d}u\\
&= g(H) + g'(H)(S-H) + \mathbb{1}_{\{S>H\}}\int_H^S g''(v)(S-v)\mathrm{d}v + \mathbb{1}_{\{S<H\}}\int_S^H g''(v)(v-S)\mathrm{d}v\\
&= g(H) + g'(H)(S-H) + \mathbb{1}_{\{S>H\}}\int_H^\infty g''(v)(S-v)_+\mathrm{d}v + \mathbb{1}_{\{S<H\}}\int_0^H g''(v)(v-S)_+\mathrm{d}v\\
&= g(H) + g'(H)(S-H) + \int_H^\infty g''(v)(S-v)_+\mathrm{d}v + \int_0^H g''(v)(v-S)_+\mathrm{d}v
\end{aligned}$$

□

We only need to expand the payoff $g(S_T)$ of the European option into the Carr-Madan-formula and compute the discounted expectation under measure $\mathbb{Q}$ which will give us the same result as 2.2.7.

Proposition 2.2.5 (Realised Variance estimation under BS model)**.** *Recall that the randomness in the payoff of Variance Swap is Realised Variance. Realised variance estimation is given by*

$$Var(r_T) = \frac{1}{T}[\ln S, \ln S]_T$$

Proof. Assume S_t follows a geometric brownian motion model. Then

$$d\ln S_t = \mu dt + \sigma dW_t$$

where μ is drift, σ is volatility and W_t is a Brownian motion.

Define the daily log-return

$$r_t := \ln S_t - \ln S_{t-1} = \mu + \int_{t-1}^t \sigma dW_s$$

.

Thus, the variance of daily log-return(using ito's isometry) and quadratic variation of log stock price are given by

$$Var(r_t) = Var(\ln S_t - \ln S_{t-1}) = \int_{t-1}^t \sigma^2 ds = \sigma^2$$

$$[\ln S, \ln S]_t = \int_0^t d\;\ln S_s d\ln S_s = \int_0^t \sigma^2 ds = \sigma^2 t$$

Hence we can conclude that

$$Var(r_T) = \frac{1}{T}[\ln S, \ln S]_T \tag{2.2.9}$$

□

Applying ito's formula to $f(S) = \ln(S)$

$$d\ln(S_t) = \left(r - \frac{1}{2}\sigma^2\right) dt + \sigma dW_t$$

Then

$$\frac{1}{2}\sigma^2 dt = \frac{dS_t}{S_t} - d\ln(S_t)$$

By equation 2.2.9, the Realised Variance V_T is given by

$$\begin{aligned}
V_T = \frac{1}{T}[\ln S, \ln S]_T = \frac{1}{T}\int_0^T \sigma^2 ds &= \frac{2}{T}\left[\int_0^T \frac{dS_t}{S_t} - \int_0^T d\ln(S_t)\right] \\
&= \frac{2}{T}\left[\int_0^T \frac{dS_t}{S_t} - \ln\frac{S_T}{S_0}\right] \\
&= \frac{2}{T}\left(rT + \sigma W_T - \ln\frac{S_T}{S_0}\right) \\
&= \frac{2}{T}\left(\ln\left(e^{rT}\right) + \sigma W_T - \ln\frac{S_T}{S_0}\right) \\
&= \frac{2}{T}\left(\sigma W_T - \ln\frac{S_T}{e^{rT}S_0}\right) \\
&= \frac{2}{T}\left(\sigma W_T - \ln\frac{S_T}{E^{\mathbb{Q}}\left[S_T\right]}\right) \\
&= \frac{2}{T}\left(\sigma W_T - \ln\frac{S_T}{E^{\mathbb{Q}}\left[S_T\right]}\right)
\end{aligned}$$

Now we take the expectation under risk netural measure $\mathbb{Q}$ on both sides:

$$E^{\mathbb{Q}}\left[V_T\right] = -\frac{2}{T}E^{\mathbb{Q}}\left[\ln\frac{S_T}{E^{\mathbb{Q}}\left[S_T\right]}\right]$$

Thus, we can replicate the Realised Variance part by a log-contract $\ln\frac{S_T}{E^{\mathbb{Q}}[S_T]}$. More specifically, $E^{\mathbb{Q}}\left[S_T\right] = F$ where F is the forward price of the underlying stock S. Now we denote this log-contract as $\ln\frac{S_T}{F}$.

Although we successfully replicated Realised Variance using log-contract, unfortunately, we cannot trade log-contract. Therefore, the theory of static replication becomes crucial. We will demonstrate how to apply this theory to log-contracts.

$$\begin{aligned}
e^{-r(T-t)}E^{\mathbb{Q}}\left[g\left(S_T\right) \mid \mathcal{F}_t\right] = e^{-r(T-t)}g(H) + e^{-r(T-t)}\left(E^{\mathbb{Q}}[S_T] - H\right) + \int_0^H g''(K)P\left(S_t, K, T\right) dK \\
+ \int_H^\infty g''(K)C\left(S_t, K, T\right) dK
\end{aligned}$$

By choosing non-negative constant $H = F$, $g(S_T) = \ln\frac{S_T}{F}$ and notice that

$$g'(K) = \frac{1}{K} \text{ and } g''(K) = -\frac{1}{K^2}$$

Thus

$$\begin{aligned}
e^{-r(T-t)}E^{\mathbb{Q}}\left[\ln\frac{S_T}{F_T} \mid \mathcal{F}_t\right] &= e^{-r(T-t)}\ln\frac{F_T}{F_T} + e^{-r(T-t)}\left(E^{\mathbb{Q}}[S_T] - F_T\right) - \int_0^F \frac{1}{K^2}P\left(S_t, K, T\right) dK \\
&\quad - \int_F^\infty \frac{1}{K^2}C\left(S_t, K, T\right) dK \\
&= -\int_0^F \frac{1}{K^2}P\left(S_t, K, T\right) dK - \int_F^\infty \frac{1}{K^2}C\left(S_t, K, T\right) dK
\end{aligned}$$

Thus, we can conclude that Realised Variance part in Variance Swap can be replicated by European Calls and Puts option strips weighted by $\frac{1}{K^2}$ which is coincide the figure we showed before with constant Dollar Gamma.

In summary, we do not trade variance swaps directly because the European call and put option markets are large, liquid and easy to operate, and we can replicate variance swaps by trading weighted delta-hedged options with different strike prices continuously for the purpose of capturing VRP. In the following, we will use reinforcement learning to adjust our target vega to ensure a utility function by trading weighted delta-hedged options to harvest VRP within prescribed leverage limits. Detailed discussions will be presented in the next chapter.

Chapter 3

Reinforcement Learning Prerequisites

3.1 Introduction of Reinforcement Learning

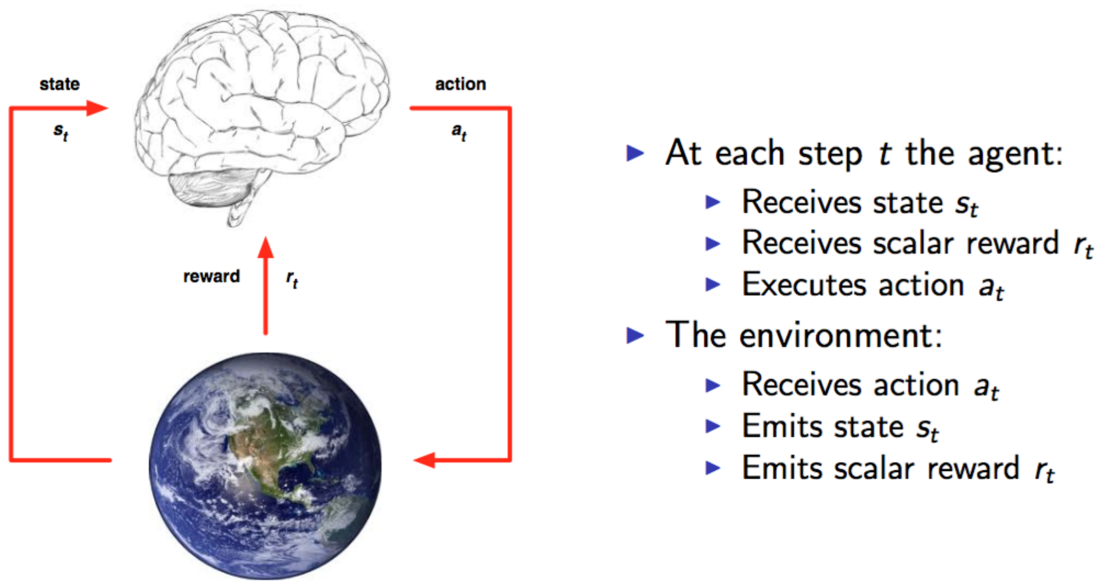
This chapter aim to introduce deep Q learning, which is a fusion of reinforcement learning and deep learning. We will delve into the essence and application of the algorithm in this chapter, in particular its application in harvesting VRP. The following reinforcement learning knowledge is based on [7, silver, 2015] and detailed algorithm construction will be introduced later.

Imagine guiding someone to invest in stocks without explicitly informing them of the number of stocks they are investing in. When they make a profit in the stock market, we consider the money earned as a positive return. Conversely, when they lose money, we view the money lost as a negative return. Over time, and after countless stock investing efforts, this person gradually learns how to invest in stocks to maximise returns. This process of learning driven by rewards and punishments encapsulates the essence of reinforcement learning.

In reinforcement learning, there are several representative terms: Agent, Action, Environment, State, and Reward.

- The "Agent" typically represents an entity capable of taking actions, such as the individual in the aforementioned example.
- The "Environment" refers to the surroundings perceived by the Agent during its activities, like the stock market or the computer it operates in the given example.
- At its core, reinforcement learning is a process of interaction between the Agent and the Environment. Throughout this process, the concepts of Action, Observation, and Reward are indispensable.
- The "Action" a_t denotes specific activities performed by the Agent to interact with the Environment. In the stock market example, the quantity of stocks purchased is the action.
- The "State" s_t encompasses a set of metrics observable from the Environment. Since the Agent cannot perceive all the information within the Environment, we use the term "Observation" to represent the available information. Stock prices, stock volume, and news are the observable information in the given example.
- The "Reward" r_t serves as a criterion to judge the quality of the actions taken. If the stocks generate profit, the reward is positive, and if there's a loss, the reward is negative which indicating the quality of the action.

We can find the entire interaction process of reinforcement learning in Figure 3.1 sourced from [7, silver, 2015]. Now that we understand how the whole process works, the next step is to determine the goal of the task. Typically, we set the goal as obtaining as many rewards as possible. The more rewards we get, the better the agent will be able to complete the task. As can be seen from 3.1, at each time step t, we determine the action based on the observed state. Thus, there is a mapping relationship between states and actions. A straightforward idea is that a state corresponds



- At each step t the agent:
  - Receives state s_t
  - Receives scalar reward r_t
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_t
  - Emits scalar reward r_t

Figure 3.1: Interaction process of reinforcement learning

to an action, which means that when an agent observes a particular state, it takes a particular action. Another approach is probabilistic, viewing this mapping relationship as the probabilities corresponding to different actions. The higher the probability associated with a particular action, the more that action should be taken. Therefore, we call to this mapping relationship as the policy π

$$a = \pi(s) \text{ or } \pi(a|s)$$

Initially, we do not know what the optimal policy is. Therefore, we can start experimenting with a random policy to obtain a series of samples of state, action and reward.

$$\{s_1, a_1, r_1, s_2, a_2, r_2, ...\}$$

Thus, reinforcement learning can improve the policy by learning from these samples, aiming to achieve increasingly better Rewards.

3.1.1 Markov Decision Process

We have covered the structure of reinforcement learning. Next up is the model's assumption — the Markov Decision Process (MDP). The Markov Decision Process describes the entire environment, i.e., the next state s_{t+1} is entirely determined by the current state s_t.

Definition 3.1.1 (Markov Process)**.** A process S_t is Markov if and only if

$$\mathbb{P} = [S_{t+1}|S_1, ..., S_t] = \mathbb{P} = [S_{t+1}|S_t]$$

Markov process means the current state s_t encompasses all pertinent details from the past and once the state is identified, the historical data becomes redundant.

Definition 3.1.2 (Markov Reward Process)**.** A Markov Reward Process is a tuple

$$\langle S, \mathcal{P}, \mathcal{R}, \gamma \rangle$$

- S is a finite set of states
- $\mathcal{P}$ is a state transition probability matrix with $\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s'|S_t = s\right]$
- $\mathcal{R}$ is a reward function with $\mathcal{R}_s = E\left[R_{t+1}|S_t = s\right]$
- γ is a discount factor with $\gamma \in [0, 1]$

3.1.2 Value function and Bellman Equation

Definition 3.1.3 (Return)**.** The return G_t is the total discounted reward from time t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where the discount factor $\gamma \in [0,1]$ represents the present value of future rewards.

The reason for introducing γ discount factor is that people are usually more interested in immediate rewards than in delayed rewards. If we represent rewards only in terms of return, then we cannot fully observe the entire process unless it is over. Therefore, we introduce the value function $v(s)$ to represent the potential future value of a state.

Definition 3.1.4 (State value function)**.** The state value function v(s) of an Markov decision process is the expected return conditional on state s

$$v(s) = E\left[G_t | S_t = s\right]$$

We can obtain the optimal policy π^* by using the state value function $v(s)$. If we know the value associated with each state, then we prefer states with higher values. Therefore, it is crucial to estimate the state value function. Hence, we introduce the Bellman equation to estimate the state value function.

Definition 3.1.5 (Bellman Equation)**.** The state value can be decomposed into 2 parts as following

$$\begin{aligned} v(s) &= E\left[G_t \mid S_t = s\right] \\ &= E\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\ &= E\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\ &= E\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= E\left[R_{t+1} + \gamma v\left(S_{t+1}\right) \mid S_t = s\right] \end{aligned}$$

where we have used tower property $E\left[G_{t+1} \mid S_t = s\right] = E\left[E\left[G_{t+1} | S_{t+1}\right] | S_t = s\right]$ in the last equality.

More specifically, in the Markov Reward Process $\langle S, \mathcal{P}, \mathcal{R}, \gamma \rangle$

$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v\left(s'\right)$$

Typically, when we define a reward function, the premise is that given a certain state and taking a specific action, we can determine the amount of reward. Therefore, we need to introduce an action set to the Markov reward process.

Definition 3.1.6 (Markov Decision Process)**.** A Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{A}$ is the action set
- $\mathcal{P}$ is a state transition probability matrix with $\mathcal{P}_{ss'}^a = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s, A_t = a\right]$
- $\mathcal{R}$ is a reward function with $\mathcal{R}_s^a = E\left[R_{t+1} \mid S_t = s, A_t = a\right]$
- γ is a discount factor with $\gamma \in [0,1]$

Definition 3.1.7 (Policy)**.** In the environment $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, a policy is the conditional distribution of action given state

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right]$$

After introducing the policy, the probability of taking action in each state will follow the policy, which is different from the previous random action. Therefore, we define the State Value function and the Action Value function under the policy.

The State value function

$$v_\pi(s) = E_\pi\left[G_t | S_t = s\right] = E_\pi\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s\right]$$

The Action value function

$$q_\pi(s,a) = E_\pi\left[G_t|S_t = s, A_t = a\right] = E_\pi\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \mid S_t = s, A_t = a\right]$$

The State value function can be represent by Action value function

$$\begin{aligned} v_\pi(s) = E_\pi\left[G_t|S_t = s\right] &= E_\pi\left[E_\pi\left[G_t|S_t = s|A_t = a\right]\right] \\ &= E_\pi\left[E_\pi\left[G_t|S_t = s, A_t = a\right]\right] \\ &= E_\pi\left[q_\pi(s,a)\right] \\ &= \sum_{a\in\mathcal{A}} \pi(a|s) q_\pi(s,a) \end{aligned}$$

The Action value function can be represent by State value function

$$\begin{aligned} q_\pi(s,a) = E_\pi\left[G_t|S_t = s, A_t = a\right] &= E_\pi\left[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a\right] \\ &= E_\pi\left[R_{t+1} + \gamma E_\pi\left[G_t|S_t = s\right]|S_t = s, A_t = a\right] \\ &= E_\pi\left[R_{t+1} + \gamma v_\pi(s)|S_t = s, A_t = a\right] \\ &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi\left(s'\right) \end{aligned}$$

Thus, combining with these 2 equation, we get the Bellman Equation for $v_\pi(s)$ and $q_\pi(s,a)$

$$\begin{aligned} v_\pi(s) &= \sum_{a\in\mathcal{A}} \pi(a \mid s)\left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi\left(s'\right)\right) \\ q_\pi(s,a) &= \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a'\in\mathcal{A}} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right) \end{aligned}$$

3.1.3 Optimal value function and iteration method

Definition 3.1.8 (Optimal value function)**.** The optimal state value function $v^*(s)$ is the maximum state value function over all policies

$$v^*(s) = \max_\pi v_\pi(s)$$

The optimal action value function $q^*(s,a)$ is the maximum action value function over all policies

$$q^*(s,a) = \max_\pi q_\pi(s,a)$$

Definition 3.1.9 (Partial ordering over policies)**.** Suppose there are 2 policies π_1, π_2, if $v_{\pi_1} \geq v_{\pi_2}$ for $\forall s$, then

$$\pi_1 \geq \pi_2$$

For any Markov Decision Process, there exists an optimal policy π^* that is better than or equal to all other policies, i.e. $\pi^* \geq \pi, \forall\pi$.

Definition 3.1.10 (Policy Iteration)**.** From the Bellman equation, we can infer that we can update our state value function through iterative methods

$$v_{k+1}(s) = \sum_{a\in\mathcal{A}} \pi(a \mid s)\left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_k\left(s'\right)\right)$$

Policy Iteration typically consists of two steps:

- Policy Evaluation: Aiming to update the Value Function, or to better estimate the value based on the current policy.
- Policy Improvement: Utilizing a greedy policy to generate new samples for the first step of policy evaluation.

Algorithm 1: Policy iteration

```
Initialization: Randomly set $V(s) \in \mathbb{R}$ and $\pi(a \mid s) \in [0,1]$ for all $s \in \mathcal{S}$;
Policy Evaluation;
while True do
    $\delta \leftarrow 0$ ;
    for each $s \in S$ do
        $v \leftarrow V(s)$;
        $V(s) \leftarrow \sum_{a \in \mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s')\right)$;
        $\delta \leftarrow \max(\delta, |v - V(s)|)$;
    if $\delta < \theta$(a small positive number) then
        break;
Policy Improvement;
policy $\leftarrow$ true;
for each $s \in S$ do
    $a \leftarrow \pi(s)$;
    $\pi(s) \leftarrow \arg\max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V(s')$;
    if $a \neq \pi(s)$ then
        policy $\leftarrow$ false;
if policy then
    stop and return $v^* = V$ and $\pi^* = \pi$ ;
else
    go to Policy Evaluation ;
```

Essentially, it involves generating new samples using the current policy and then using these new samples to better estimate the policy values. Subsequently, the policy itself is updated using the policy values, and this process is repeated. Theoretically, it can be shown that the policy will eventually converge to the optimal value. The Policy iteration pseudo code

Definition 3.1.11 (Value Iteration)**.** From the Bellman equation, we can define the optimal state value function

$$\begin{aligned} v^*(s) &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right] \\ &= \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^*(s') \end{aligned}$$

The iteration formula is given by

$$v_{k+1}(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s')$$

Algorithm 2: Value iteration

```
Initialization: Randomly set $V(s) \in \mathbb{R}$ and $\pi(a \mid s) \in [0,1]$ for all $s \in \mathcal{S}$;
while True do
    $\delta \leftarrow 0$ ;
    for each $s \in S$ do
        $v \leftarrow V(s)$;
        $V(s) \leftarrow \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V(s')$;
        $\delta \leftarrow \max(\delta, |v - V(s)|)$;
    if $\delta < \theta$(a small positive number) then
        break;
$\pi(s) = \arg\max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V(s')$;
Return $\pi(s)$;
```

3.2 Q learning algorithm

The concept of Q Learning is derived entirely from value iteration. However, it's important to note that value iteration updates the Q-values for all state-action pairs in each iteration. In practical scenarios, we cannot enumerate all states and actions; we can only obtain a limited set of samples. Thus, Q-learning provides us a new method for updating Q value based on Bellman Equation

$$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right)+\alpha\left(R_{t+1}+\lambda \max_a Q\left(S_{t+1}, a\right)-Q\left(S_t, A_t\right)\right)$$

Algorithm 3: Q learning

```
Initialization: Step size $\alpha \in (0,1], \epsilon > 0$, randomly set $Q(s,a) \in \mathbb{R}$ for all $s \in \mathcal{S}$ except that
  Q(terminal state,·) = 0;
for each episode do
    Initialize S;
    for each step do
        Choose A from S using policy derived from Q(s,a) (i.e. $\epsilon$-greedy);
        Obtain action A, reward R, next state $S'$ ;
        $Q(S,A) \leftarrow Q(S,A) + \alpha(R + \lambda \max_a Q(S',a) - Q(S,A))$;
        $S \leftarrow S'$;
```

3.2.1 Exploration and Exploitation

From the Q-learning algorithm, it's evident that generating actions is a crucial step. Consequently, two methods emerge for generating actions:

- Random generation: Using random actions corresponds to exploration, which means probing the effects of unknown actions, beneficial for updating Q-values and obtaining a better policy.
- Greedy policy: Using a greedy policy equates to exploitation. This approach might not be as effective for updating Q-values, but it can yield better test results to evaluate the algorithm's efficacy.
- Combining both approaches results in ϵ-greedy.

$$\pi(s)=\begin{cases}\arg\max_{a \in \mathcal{A}(s)} Q^*(S,a) & \text{probability } 1-\epsilon \\ \text{randomly select action from } \mathcal{A}(s) & \text{probability } \epsilon\end{cases}$$

3.3 Deep Q learning(DQN)

In Q-learning, we need to record the Q-value of each action corresponding to each state. However, in practice, this is challenging because the number of states can be very large. For example, consider training an agent to play an Atari game [8, M. Volodymyr et al, 2013] where the input is raw image data, specifically an image of 210x160 pixels, and the output is a set of key actions. In this case, how many different states are there? Theoretically, if each pixel can have 256 different values (due to the 8-bit colour depth), then the total number of states is $256^{(210\times160)}$, which is an astronomical number! Faced with a high-dimensional state space, these states must be efficiently compressed or represented. Value Function Approximation (VFA) is one solution.

Definition 3.3.1 (Value Function Approximation)**.** Value Function Approximation is use a function to approximate $Q(s,a)$ by a function $f(s,a)$ where $f : \mathcal{S} \to |\mathcal{A}|$. With Value Function Approximation, we observe that regardless of the dimension of the input state, we can obtain a corresponding action vector of a length equal to the number of actions. Consequently, we can further represent this function using a unified parameter set, denoted as θ.

$$Q(s,a) \approx f(s,a,\theta)$$

In this approach, a neural network takes a state as input and produces Q-values for all potential actions. The networks becomes a function $f : \mathcal{S} \to |\mathcal{A}|$ where $|\mathcal{A}|$ represents the number of action and parameters are optimized to minimize the discrepancy between its predicted Q-values and the target Q-values, which are based on the Bellman equation.

3.3.1 Deep learning for value function approximation

In this section, we introduce the feedforward neural network, which we will employ for value function approximation. The feedforward neural network is defined as following[9, Lukas, 2022]:

Definition 3.3.2 (Feedforward neural network)**.** Let $I, O, r \in \mathbb{N}$. A function $\boldsymbol{f} : \mathbb{R}^I \to \mathbb{R}^O$ is a feedforward neural network (FNN) with $r - 1 \in \{0, 1, \dots\}$ hidden layers, where there are $d_i \in \mathbb{N}$ units in the i-th hidden layer for any $i = 1, \dots, r-1$, and activation functions $\boldsymbol{\sigma}_i : \mathbb{R}^{d_i} \to \mathbb{R}^{d_i}, i = 1, \dots, r$, where $d_r := O$, if

$$\boldsymbol{f} = \boldsymbol{\sigma}_r \circ \boldsymbol{L}_r \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{L}_1, \tag{3.3.1}$$

where $\boldsymbol{L}_i : \mathbb{R}^{d_{i-1}} \to \mathbb{R}^{d_i}$, for any $i = 1, \dots, r$, is an affine function

$$\boldsymbol{L}_i(\boldsymbol{x}) := W^i \boldsymbol{x} + \boldsymbol{b}^i, \quad \boldsymbol{x} \in \mathbb{R}^{d_{i-1}}, \tag{3.3.2}$$

parameterised by the weight matrix $W^i = \left[W^i_{j,k}\right]_{j=1,\dots,d_i, k=1,\dots,d_{i-1}} \in \mathbb{R}^{d_i \times d_{i-1}}$ and the bias vector $\boldsymbol{b}^i = \left(b^i_1, \dots, b^i_{d_i}\right) \in \mathbb{R}^{d_i}$, with $d_0 := I$. We shall denote the class of such functions f by

$$\mathcal{N}_r\left(I, d_1, \dots, d_{r-1}, O; \sigma_1, \dots, \sigma_r\right).$$

If $\boldsymbol{\sigma}_i(\boldsymbol{x}) = \left(g\left(x_1\right), \dots, g\left(x_{d_i}\right)\right), \boldsymbol{x} = \left(x_1, \dots, x_{d_i}\right) \in \mathbb{R}^{d_i}$, for some $g : \mathbb{R} \to \mathbb{R}$, we write g in place of $\boldsymbol{\sigma}_i$.

The integers $r, d_1 \dots, d_{r-1}$ are called the hyperparameters of the FNN - to distinguish them from the weights in $W^1 \dots, W^r$ and biases in $\boldsymbol{b}^1, \dots, \boldsymbol{b}^r$, which are the actual parameters of the network.

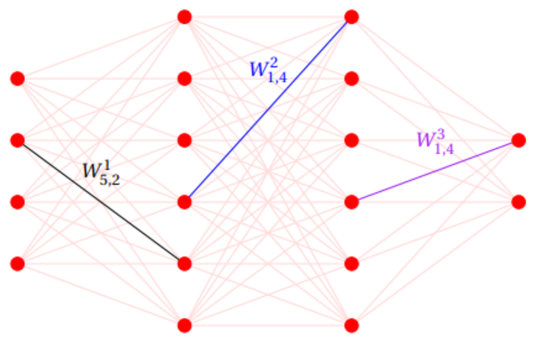


Figure 3.2: Graphical representation of a feedforward neural network

To practically apply it to our value function approximation, we need to standardize the activation function, loss function and risk, and gradient descent method.

- Activation function: In our experiment, we choose rectified linear unit (ReLU) function as an activation function in hidden layer and identity function as an activation function in output layer since ReLU is computational simplicity and performing well in practice.

$$ReLU(x) := \max\{x, 0\} \text{ and } Id(x) = x$$

- Loss function and Risk: Suppose $\boldsymbol{f} : \mathbb{R}^I \to \mathbb{R}^O$ and $\boldsymbol{f} \in \mathcal{N}_r\left(I, d_1, \dots, d_{r-1}, O\right)$, loss function is defined as

$$\ell : \mathbb{R}^O \times \mathbb{R}^O \to \mathbb{R}.$$

Given input $\boldsymbol{x} \in \mathbb{R}^I$ and target value $\boldsymbol{y} \in \mathbb{R}^O$ we compute the Loss as $\ell(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y})$.

If $\boldsymbol{x}$ and $\boldsymbol{y}$ are samples of joint random variables $(\boldsymbol{X}, \boldsymbol{Y})$, we could try to seek an optimal $\boldsymbol{f}$ by minimising the risk

$$\mathbf{E}[\ell(\boldsymbol{f}(\boldsymbol{X}), \boldsymbol{Y})]. \tag{3.3.3}$$

In practice, we often don't have access to the distribution of $(\boldsymbol{X}, \boldsymbol{Y})$, Instead, we rely on empirical methods, using sampled data points $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^N$ of $\boldsymbol{x}$ and $\boldsymbol{y}^1, \ldots, \boldsymbol{y}^N$ of $\boldsymbol{y}$ for some $N \in \mathbb{N}$. As an empirical proxy of 3.3.3, we then work with empirical risk

$$\mathcal{L}(\boldsymbol{f}) := \frac{1}{N} \sum_{i=1}^{N} \ell\left(\boldsymbol{f}\left(\boldsymbol{x}^i\right), \boldsymbol{y}^i\right) \tag{3.3.4}$$

We could define minibatch risk

$$\mathcal{L}_B(\boldsymbol{f}_\theta) := \frac{1}{\# B} \sum_{i \in B} \ell\left(\boldsymbol{f}_\theta\left(\boldsymbol{x}^i\right), \boldsymbol{y}^i\right) \tag{3.3.5}$$

- Gradient descent method: Euler approximation serves as the foundation for gradient descent, an iterative method that incrementally searches for a minimizer using gradient updates:

  $$\boldsymbol{\theta}_{\text{new}} := \boldsymbol{\theta}_{\text{old}} - \eta \nabla F\left(\boldsymbol{\theta}_{\text{old}}\right)$$

  given some initial condition θ_0 and learning rate $\eta > 0$.

  When working with neural networks, one might initially consider directly minimizing the empirical risk $\mathcal{L}_B\left(\boldsymbol{f}_\theta\right)$, as outlined in 3.3.5. However, determining the gradient of $\mathcal{L}_B\left(\boldsymbol{f}_\theta\right)$ can be computationally expensive, especially with a large value of N. Moreover, using gradient descent on $\mathcal{L}_B\left(\boldsymbol{f}_\theta\right)$ might result in an overfitting model $\boldsymbol{f}_\theta$. Due to these challenges, stochastic gradient descent (SGD) is often the favored approach for training neural networks.

  In SGD, the training data, denoted by indices 1,...,N, is partitioned into random minibatches. These minibatches are then sequentially utilized to calculate gradient updates. To this end, we fix minibatch size $m \ll N$, typically such that N is divisible by m, that is, $N = \mathrm{km}$ for some $k \in \mathbb{N}$. We then sample uniformly minibatches $B_1, \ldots, B_k \subset \{1, \ldots, N\}$, such that $\# B_i = m$ for any $i = 1, \ldots, k$, without replacement and $B_1, \ldots, B_k$ are disjoint with $\cup_{i=1}^k B_i = \{1, \ldots, N\}$. Starting from initial condition $\boldsymbol{\theta}_0$, the parameter vector $\boldsymbol{\theta}$ is updated via

  $$\boldsymbol{\theta}_i := \boldsymbol{\theta}_{i-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_{B_i}\left(\boldsymbol{\theta}_{i-1}\right), \quad i = 1, \ldots, k,$$

  where $\mathcal{L}_{B_i}(\boldsymbol{\theta})$ is minibatch empirical risk corresponding to B_i.

3.3.2 Implementation of the DQN Algorithm

We have already provided an in-depth introduction to deep learning. To train our neural network, we require a substantial amount of sample data. We then update our parameters using gradient descent through backpropagation. Consequently, for every input data pairs (s, a), we need to assign a label $Q^{\text{target}}(s, a)$ as following

$$Q^{\text{target}}(s, a) := R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$$

Thus, the risk in neural network is defined as

$$\mathcal{L}(\boldsymbol{\theta}) = E\left[\left(r + \gamma \max_{a'} Q(S', a', \theta) - Q(s, a, \theta)\right)^2\right]$$

The DQN algorithm is presented as following[8, M. Volodymyr et al, 2013]:

The algorithm is described primarily concerned with the Experience Replay technique which is the way of the samples are stored and sampled. Given that the samples collected from the Atari game are arranged chronologically, there is continuity between the samples. Due to the distribution of the samples, updating the Q-value directly as each sample is received may lead to sub-optimal results. Therefore, a straightforward approach is to store these samples first and then sample randomly, which refect the essence of "experiential playback". From a neuroscience perspective, the human brain employs a similar mechanism of learning through recall.

Thus, DQN is essentially centered around iterative experimentation and data storage. Once enough data has been accumulated, random samples are drawn for gradient descent.

Algorithm 4: Deep Q-learning with Experience Replay

```
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights $\theta$
Initialize target action-value function $Q^*$ with random weights $\theta^* = \theta$
for episode= 1, ..., $M$ do
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed $\phi_1 = \phi(s_1)$;
    for $t$ = 1, ..., $T$ do
        With probability $\epsilon$ select a random aciton $a_t$;
        otherwise select $a_t = \arg\max_{a \in \mathcal{A}(s)} Q^*(\phi(s_t), a; \theta^*)$ ($\epsilon$-greedy);
        Execute action $a_t$ in emulator and observe reward $r_t$ and state $x_{t+1}$;
        Set $s_{t+1} = x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$;
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$;
        if number of samples in $\mathcal{D}$ is large enough then
            Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$ ;
            Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$ ;
            Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to $\theta$;
```

3.3.3 Improvement of DQN by DDDQN

In this section, we will improve the DQN algorithm, thus introducing the Double Dueling DQN (DDDQN) algorithm. We will explore the DDDQN algorithm in depth and perform numerical experiments in following sections.

- Double DQN: A fundamental problem with DQN is the tendency to overestimate the Q-value. This overestimation bias stems from the fact that the Q-value is derived from the maximum target Q-value. Given that the Q-value estimates themselves contain noise, especially in the early stages of network training, this can lead to overly optimistic Q-value predictions.

  The Double DQN (DDQN) algorithm, proposed by [10, V. Hado et al,2016], mitigates this overestimation bias by separating action selection from its valuation. Rather than employing a singular network for both action selection and its assessment, DDQN utilizes the primary Q-network for action determination and the auxiliary target Q-network for gauging the value of the chosen action as following

  The Double DQN (DDQN) algorithm proposed by [10, V. Hado et al,2016] to mitigate this overestimation bias by separating action selection from its value evaluation. DDQN does not use a single network for action selection and its evaluation. Instead it use the primary Q-network to determine the action and the auxiliary target Q-network to measure the selected action which is shown as following

$$Q(s,a) = r + \gamma Q\left(s', \text{argmax}_{a'} Q^*\left(s', a'; \theta^*\right); \theta\right)$$

  Experimental results show that DDQN solves the overestimation bias problem by separating the action selection and action evaluation processes compared to the traditional DQN algorithm. This modification leads to more accurate Q-value prediction and higher training stability.

- Dueling DQN: Based on the traditional DQN framework, Dueling-DQN proposes a refined method[11, W. Ziyu et al,2016].The core principle of Dueling-DQN lies in the bifurcation of its state-value representation $V(s)$ and its corresponding action dominance function $A(s,a)$, which is designed to address the differences in computational requirements between different actions. Combining $V(s)$ and $A(s,a)$, then

$$Q(s,a) = V(s) + A(s,a)$$

  Within the same neural network, we divide the output layer into two parts: one to estimate the state-value function $V(s)$ and the other to estimate the advantage function $A(s,a)$. These two parts are then combined to produce the output for estimating $Q(s,a)$. Subsequently, these two parts are combined to produce the output used to estimate $Q(s,a)$. The rationale behind this design of the Dueling DQN is to decouple the state value function from the advantage function in the Q value function. The state value function is only responsible for assessing the quality of a state, while the advantage function only measures the importance of each action in that state. Then Q function reflects the extent to which taking a particular action increases the value of a particular state. By ensuring that each branch focuses on its respective task, the accuracy of the prediction is improved.

However, it is not feasible to train a neural network simply by adding value and advantage functions. In the equation Q = V+A, given Q, recognizing the respective values of V and A is problematic due to their "unidentifiability". To solve this problem, a technique was proposed in the research: limiting the maximum value of Q to coincide with the value of V ensures that the peak of the advantage function is zero and all other values are negative. This allows for accurately determining the value of V so that all advantage can be calculated. Here is the training methodology:

$$Q(s,a) = V(s) + \left(A(s,a) - \max_{a' \in |\mathcal{A}|} A(s,a) \right)$$

The paper also suggests another training methodology as following:(Read more details in paper[11, W. Ziyu et al, 2016]):

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a) \right)$$

Chapter 4

Implementation of DDDQN

4.1 Delta-hedging with DDDQN

In this section, we will discuss how to use the DDDQN algorithm aiming to train an agent for delta hedging through the construction of a reward function. Subsequently, we will introduce the formulation of the problem. To implement this algorithm, we require 6 components: stock price model, option parameters, state, action, reward function and neural network parameters.

- Stock price model: Assuming the stock prices follows Geometric Brownian motion

  $$dS_t = rS_t dt + \sigma S_t dW_t^{\mathbb{Q}}$$

  Assuming expiry date = T(Year) and we split the date into N+1 samples, $t_0, t_1, ..., t_N$ and simulation scheme of the stock prices is given by

  $$S_{t_i} = \exp\left\{\log(S_0) + \sum_{k=1}^{k=i}(r - \frac{\sigma^2}{2})(t_k - t_{k-1}) + \sigma\sqrt{t_k - t_{k-1}}\mathcal{N}_k\right\}$$

  where $\mathcal{N}_1, ..., \mathcal{N}_N$ are iid standard normal random variables.

- Option parameters: We aim to delta-hedged 1Y ATM call option with following parameters

Flag	call
Volatility σ_t	0.15
Step size Δt	1
Maturity T	365
Interest rate r	0
Strike price K	100
Initial Stock price S_0	100

Table 4.1: Option Parameters

- State: We define the stock prices S_t and time to maturity $\tau = T - t$ as the state at time t forming a tuple

  $$s_t = \{S_t, \tau\} \in \mathcal{S}$$

  .

- Action: We aim to learn the BS-delta and recall that $0 \leq \Delta_{call} \leq 1$, we define action $a_t \in [0, 1]$ represents the quantity of stocks used for hedging. Since DQN addresses problems with discrete actions, we partition the action set into 101 segments, corresponding to

  $$a_t \in \mathcal{A} := \{0.00, 0.01, ..., 0.99, 1.00\}$$

- Reward function: Recall that we have showed $E[\Pi_t] = 0$ for each t where Π_t is the pnl of delta-hedged portfolio. Thus, the reward is constructed as following:

  $$r_t = -E\left[|a_{t_i}(S_{t_i} - S_{t_{i-1}}) - (C_{t_i} - C_{t_{i-1}})|\right]$$

  which implies that agent will be penalised by incorrect delta-hedge.

- Neural network parameters:

hidden layers activation function	ReLU
Output layers activation function	Id
Number of input(state) dimension	2
Number of hidden layers	2
hidden layers dimension	64
Number of output(action) dimension	101

Table 4.2: Neural Network Parameters

The implementation of DDDQN algorithm is based on [12, fschur, 2021].

Algorithm 5: Deep Q-learning with Experience Replay

```
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights $\theta$
Initialize target action-value function $Q^*$ with random weights $\theta^* = \theta$
Initialize gradient decent algorithm: Adam with learning rate adjustment scheduler:
  StepLR
Initialize algorithm parameters: learning rate = 0.001, $\gamma = 0.99$, $\epsilon = 1$, $\epsilon$ - decay = 0.999,
  step - size = 1000*64, update-rate $\tau = 0.005$
for episode= 1, ..., M do
    Initialize sequence $s_0 = \{S_0\}$ where $S_0$ is stock price at time 0 ;
    for t = 0, 1, ..., T do
        With probability $\epsilon$ select a random aciton $a_t$;
        otherwise select $a_t = \arg\max_{a\in\mathcal{A}(s)} Q^*(s_t, a; \theta)$ ($\epsilon$-greedy);
        Execute action $a_t$ in emulator and observe reward $r_t$ and state $x_{t+1}$;
        Set $s_{t+1} = x_{t+1}$ and store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$;
        if number of samples in $\mathcal{D}$ is large enough then
            Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$ ;
            Set $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma Q\left(s', \operatorname{argmax}_{a'} Q^*\left(s', a'; \theta^*\right); \theta\right) & \text{for non-terminal } s_{j+1} \end{cases}$ ;
            Perform a gradient descent step on $\left(y_j - Q\left(s_j, a_j; \theta\right)\right)^2$ with respect to $\theta$ where
                $$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a)\right)$$
            Update target network parameters with polyak update (soft update):
                $$\theta' \leftarrow \tau * \theta + (1-\tau) * \theta'$$
```

After training, the expected reward is plotted against the number of episodes in Figure 4.1

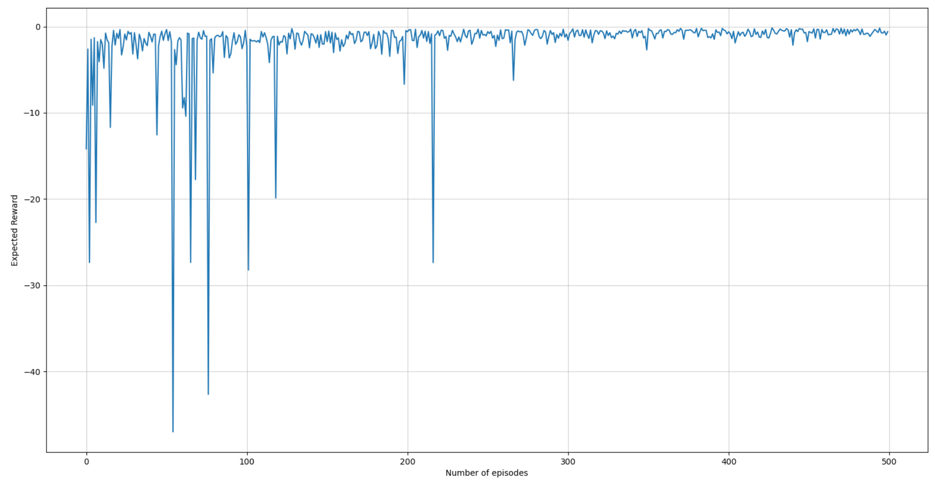


Figure 4.1: Graphical representation of expected reward against the number of episodes

As we can see from Figure 4.1, the expected reward basically converges to around 0 after 230 episodes. This indicates that the agent's training performance is particularly good, and the algorithm converges rapidly. The oscillation in the reward can be said to the fact that the states used in each episode are simulated and action is predicted from greedy algorithm, which in turn causes the reward to fluctuate.

To further assess whether our agent has accurately predicted the BS delta, we have plotted both the BS-delta and the agent's action in Figure 5.1.

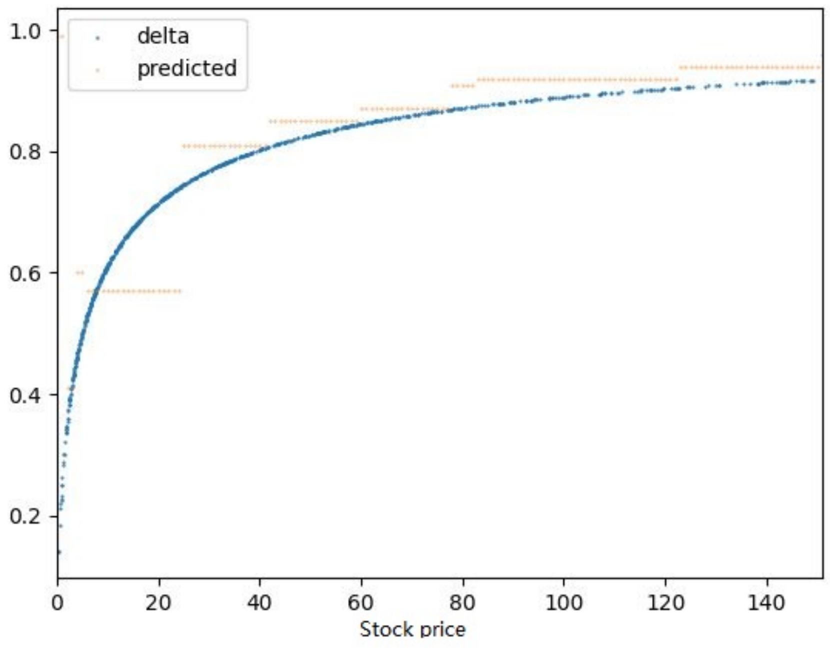


Figure 4.2: Graphical representation of agent delta and BS delta with respect to stock price

This section provides a foundational experiment on the DDDQN algorithm. In next section, we will discuss how to backtest the strategy and apply the algorithm to reality problems.

Chapter 5

Backtesting DDDQN algorithm

5.1 Performance Metrics

In this chapter, we introduce the method for calculating the index I, which consists of the total PnL(profit and loss) of all positions in a strategy. This methodology helps to more effectively compare the PnL of different strategies as a benchmark for evaluating their relative merits.

Assuming start Index $I_0 = 100$ and the index at time t+1 is given by

$$I_{t+1} = I_t(1 + \sum_{\text{legs i at time t}} \omega_{t+1}^i (V_{t+1}^i - V_t^i))$$
$$I_{t+1} = I_t + I_t \sum_{\text{legs i at time t}} \omega_{t+1}^i (V_{t+1}^i - V_t^i)$$

where V_t^i represents the \$ value of leg i at time t. Now, we will introduce several important Performance Metrics.

- Sharpe Ratio:

  Definition 5.1.1 (Sharpe Ratio). The Sharpe Ratio is a tool used to assess the return of an investment relative to its risk. This metric provides a way to directly compare the performance of one investment with another. The Sharpe Ratio quantifies the additional return per unit of risk taken and is defined as

$$\text{Sharpe Ratio } = \frac{\text{Portfolio Return-Risk-Free Rate}}{\text{Portfolio Standard Deviation}}$$

- Annual Return:

  Definition 5.1.2 (Annual Return). The annual return simply denotes the percentage variation in an investment's value over one-year period. It's a basic measure that investors rely on to measure the yearly performance of their investments which is computed as:

$$\text{Annual Return } = \left(\frac{\text{Portfolio Final Value}}{\text{Portfolio Initial Value}}\right)^{\frac{1}{\text{Number of Years}}} - 1$$

- Cumulative Returns:

  Definition 5.1.3 (Cumulative Returns). Cumulative returns provide a comprehensive perspective on the overall return of an investment for a given duration which can be expressed as:

$$\text{Cumulative Return } = \frac{\text{Portfolio Final Value}}{\text{Portfolio Initial Value}} - 1$$

- Max Drawdown:

Definition 5.1.4 (Max Drawdown)**.** Maximum drawdown is a measure that captures the most significant decline in a portfolio's value from its peak to its lowest point, prior to reaching a new peak. It represents the most unfavorable potential loss of an investment over a specified period of time. The formula is as follows

$$\text{Max Drawdown}_{\ t} = \frac{\max_{s<t} P_s - P_t}{\max_{s<t} P_s}$$

- Calmar Ratio:

  Definition 5.1.5 (Calmar Ratio)**.** The Calmar Ratio is a performance metric that contrasts the Annual Return with the max drawdown which can be expressed as:

  $$\text{Calmar Ratio } = \frac{\text{Annual Return over 3 years}}{\text{Max Drawdown over 3 years}}$$

- Value at Risk(VaR):

  Definition 5.1.6 (VaR)**.** VaR is a prevalent risk assessment methodology that estimates the potential decline in the value of the portfolio over a period of time, based on a specified confidence level α which can be defined as:

  $$\text{VaR}_\alpha(X) = -\inf\{x \in \mathbb{R} : F_X(x) > \alpha\}$$

  where X represents Portfolio PnL.

5.2 Performance of static replication of Variance Swap

In this chapter, we will provide a practical experience on the performance metric for the delta-hedged portfolio replicating a Variance swap. Furthermore, since our testing set includes the occurrence of market turbulence, we will also present the drawdown in 2020 corresponds to covid19 market crash. The following are the specific parameters of the delta-hedged portfolio.

Option underlying:	S&P 500
Option date range	From 2019 to 2021
Option type	OTM
Delta hedging frequency	Daily
Target vega	15 bps

Table 5.1: Option Parameters in S&P 500 with static replication

The statistics of delta-hedged portfolios is given by

Annual return	Cumulative returns	Annual volatility	Sharpe ratio
0.001%	0.004%	0.007%	0.21
Calmar ratio	Max drawdown	5% value at risk	
0.11	-0.013%	-0.001%	

Table 5.2: Statistics of delta-hedged portfolios

The statistics of Undelta-hedged portfolios is given by

Annual return	Cumulative returns	Annual volatility	Sharpe ratio
0.004%	0.0013%	0.011%	0.42
Calmar ratio	Max drawdown	5% value at risk	
0.29	-0.015%	-0.001%	

Table 5.3: Statistics of Undelta-hedged portfolios

As can be seen from the table, the PnL on the hedged portfolio is lower than the PnL on the unhedged portfolio on average. This is because the difference in PnL between the two portfolios is reflected in the delta PnL. Moreover, the delta-hedged portfolio is short gamma. This

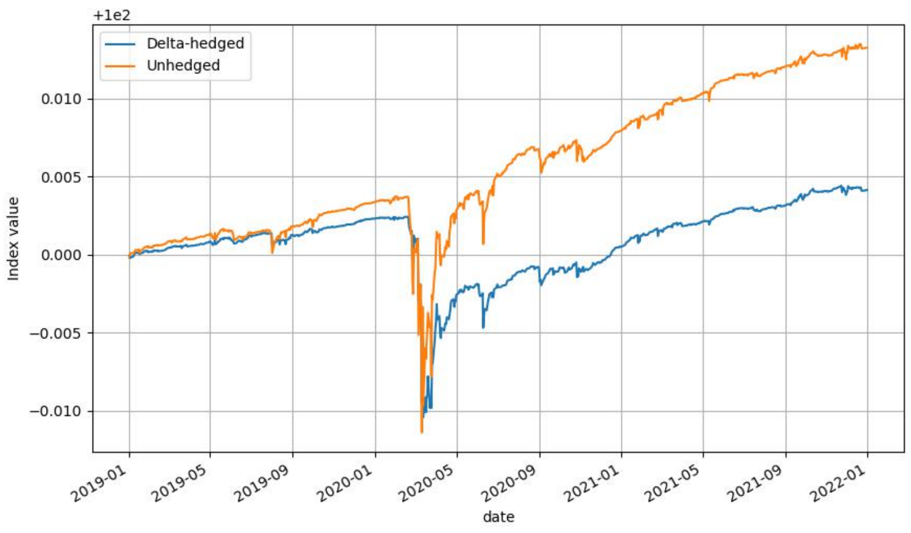


Figure 5.1: Index value of 2 portfolios

leads us to buy when the underlying price rises and sell when it falls. As a result, our pnl decreases.Undeniably, delta hedged portfolio lacks risk from delta PnL. Therefore, overall risk and volatility are reduced. While the Sharpe and Kalmar ratios indicate a higher return-to-risk ratio for the unhedged portfolio, the max drawdown for the unhedged portfolio is also higher. This is the last thing option traders, and especially asset managers want to see. And reducing the max drawdown is our primary goal later on through the reinforcement learning agent. Reducing the max drawdown is our main goal to be achieved by the reinforcement learning agent later. Lastly, we observe that at the beginning of 2020, the PnL of both strategies experienced a cliff-like drop, mainly due to the impact of the covid-19 epidemic on the financial markets, which led to the promotion of multiple meltdowns in several indices such as S&P 500, Nasdaq and so on within a month. The drop of S&P 500 even ranked third in its history.

5.3 Volatility premium harvesting with help of DDDQN

In this section, we will have the reinforcement learning agent learn how to determine our static portfolio exposure and based on a certain risk preference to harvest the VRP. In the following, we will provide the framework of the DDDQN algorithm to implement this task. Similarly, we need to define following components

- Option parameters: We aim to delta-hedged OTM option with following parameters

Underlying	S&P 500
Training option date	From 2015 to 2018
Testing option date	From 2019 to 2021
Option type	OTM
Delta hedging frequency	Daily

Table 5.4: Option Parameters in S&P 500 using DDDQN

- State: State is the value of portfolio following the bump of the options underlying with following shifts.

$$s_t = \left\{\nu_t^{-20\%}, \nu_t^{-15\%}, \nu_t^{-10\%}, \nu_t^{-5\%}, \nu_t^{0\%}, \nu_t^{5\%}, \nu_t^{10\%}, \nu_t^{15\%}, \nu_t^{20\%}\right\} \in \mathcal{S}$$

.

- Action: Recall that the static portfolio replicating Variance Swap is given by

$$\frac{1}{K^2}\left(\frac{\partial C}{\partial S}S - C\right)$$

Notice that Vega(Variance Swap) = Vega(delta-hedged portfolio), we control the leverage by specifying the target vega ν^{target} of the portfolio we want to obtain, thereby achieving the purpose of capturing the VRP as following

$$\nu^{\text{target}}\frac{\left(\frac{\partial C}{\partial S}S - C\right)}{K^2}$$

such that

$$\text{Total vega of portfolio} = \nu^{\text{target}} \times \text{Vega(weighted delta-hedged portfolio)}$$

Therefore, the agent is tasked with predicting how much vega exposure we need at each time t. Moreover, we need to set a threshold for this ν^{target} to prevent excessive use of leverage. Here, we limit the $\nu^{\text{target}} \in [0.15\%, 2.25\%]$. Meanwhile, we split the action space into 101 segments,

$$a_t = \nu_t^{\text{target}} \in \mathcal{A} := \{0.15\%, ..., 2.25\%\}$$

- Reward function: Our goal is to reduce the max drawdown. Therefore, we need to construct a utility function that makes us more averse to drawdowns. Naturally, we thought of the following function:

$$r_t = E\left[\log\left\{1 + c \times I_t \times \sum_{\text{legs i at time t}} \omega_{t+1}^i (V_{t+1}^i - V_t^i)\right\}\right]$$

where c is a positive constant to preserve enough curvature. This utility function indicates that when current Strategy Index I_t is large, we penalize more if we loss money to achieve the effet of drawdown penalization by the concave property of log function.

- Neural network parameters:

hidden layers activation function	ReLU
Output layers activation function	Id
Number of input(state) dimension	9
Number of hidden layers	1
hidden layers dimension	64
Number of output(action) dimension	101

Table 5.5: Neural Network Parameters in $S\%P$ 500

The implementation of DDDQN algorithm is based on [12, fschur, 2021].

Algorithm 6: DDDQN with soft update

```
Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights $\theta$
Initialize target action-value function $Q^*$ with random weights $\theta^* = \theta$
Initialize gradient decent algorithm: Adam with learning rate adjustment scheduler:
 StepLR
Initialize algorithm parameters: learning rate = 0.0001, $\gamma = 0.8$, $\epsilon = 1$, $\epsilon$ - decay = 0.99,
 step - size = 10*90, update-rate $\tau = 0.005$, batch-size = 32, M = 101
for episode= 1, ..., M do
    Initialize sequence $s_0 = \{S_0\}$ where $S_0$ is stock price at time 0 ;
    for t = 0, 1, ..., T do
        With probability $\epsilon$ select a random aciton $a_t$;
        otherwise select $a_t = \arg\max_{a \in \mathcal{A}(s)} Q^*(s_t, a; \theta)$ ($\epsilon$-greedy);
        Execute action $a_t$ in emulator and observe reward $r_t$ and state $x_{t+1}$;
        Set $s_{t+1} = x_{t+1}$ and store transition $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{D}$;
    if episode > 2 then
        for j = 0, 1, ..., T do
            Sample random minibatch of transitions $(s_j, a_j, r_j, s_{j+1})$ from $\mathcal{D}$ ;
            Set $y_j = \begin{cases} r_j & \text{for terminal } s_{j+1} \\ r_j + \gamma Q\left(s', \operatorname{argmax}_{a'} Q^*\left(s', a'; \theta^*\right); \theta\right) & \text{for non-terminal } s_{j+1} \end{cases}$ ;
            Perform a gradient descent step on $\left(y_j - Q\left(s_j, a_j; \theta\right)\right)^2$ with respect to $\theta$ where
                $$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a)\right)$$
            Update target network parameters with polyak update (soft update):
                $$\theta' \leftarrow \tau * \theta + (1 - \tau) * \theta'$$
```

After training, the expected reward is plotted against the number of episodes in Figure 5.2.

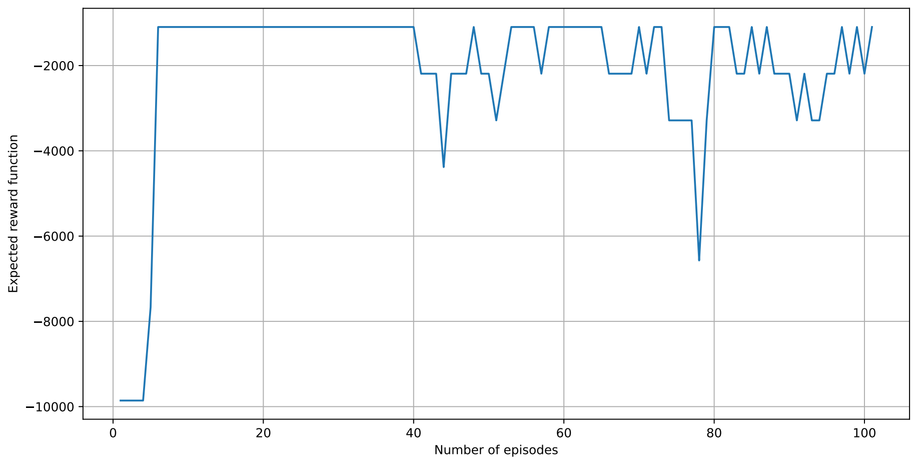


Figure 5.2: Graphical representation of expected reward against the number of episodes in training model

As we can see from Figure 5.2, we can observe that after 3-4 episodes, the algorithm begins to converge. This implies that our reward function has a good curvature. The subsequent oscillations in the Expected values are due to the use of epsilon-greedy to select actions during the training process. Thus, there will be slight fluctuations. Finally, we can demonstrate our training success through PnL and performance metrics.

The PnL of agent traded and constant target vega is given by Figure 5.3.

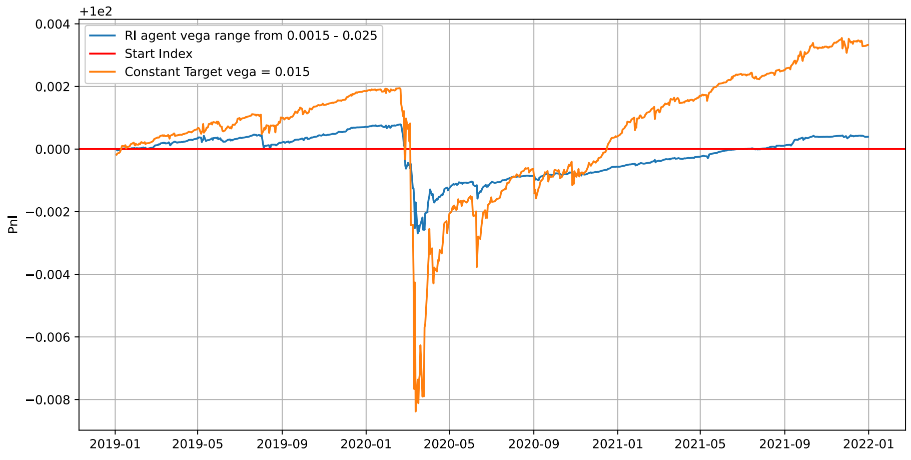


Figure 5.3: Graphical representation of PnL of agent traded and constant target vega

In order to achieve higher returns while maintaining the Sharpe ratio, we leverage up our portfolio [1] where the pnl is given by Figure 5.4.

[1]The proof is give in the Appendix

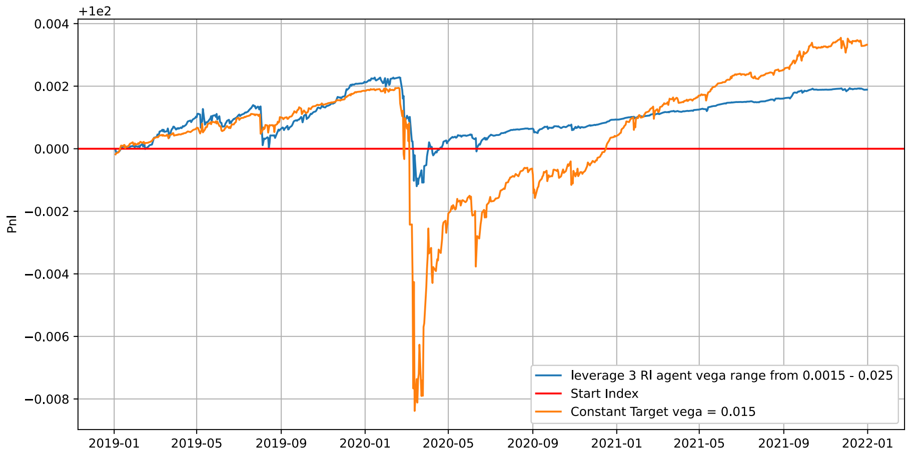


Figure 5.4: Graphical representation of PnL of agent traded and constant target vega with leverage

The statistics of the agent PnL is given by

Annual return	Cumulative returns	Annual volatility	Sharpe ratio
0.001%	0.002%	0.002%	0.38
Calmar ratio	Max drawdown	5% value at risk	
0.18	-0.003%	-0.0002%	

Table 5.6: Statistics of the agent PnL

The statistics of the constant target vega PnL is given by

Annual return	Cumulative returns	Annual volatility	Sharpe ratio
0.001%	0.004%	0.007%	0.21
Calmar ratio	Max drawdown	5% value at risk	
0.11	-0.013%	-0.001%	

Table 5.7: Statistics of the constant target vega PnL

We can see that the out of sample data has seen a significant decrease in volatility while maintaining similar annualized returns. In addition, both the Sharpe Ratio and the Calmar Ratio have improved significantly. The max drawdown has reduced substantially. One more crucial point is that our strategy keeps performing well even under market crash which is exactly what we want.

Conclusion

This thesis focuses on 3 main modules, one is how to harvest VRP and another is how to use reinforcement learning to help us harvest the most VRP for a specified level of risk aversion and the last is backeting on S&P500 option data. The success of the first part comes from the static replication theorem, which allows us to replicate the Variance Swap using a liquid and easily tradable European call and put option such that our portfolio is exposured to volatility directly.

In the second part of the reinforcement learning algorithm, we mainly focus on DQN algorithm, while in order to address the problem of the tendency of ordinary DQNs to overestimate the Q-value and to separate the state function from the value function, we introduce the DDDQN algorithm which make sure that each branch specialises in its respective task, aiming to improve the accuracy of the predictions.

During the algorithmic backtesting in the third part, we conduct two tests in total. The first test is to test whether the agent of the DDDQN algorithm predicts the BS-delta accurately. Based on the backtesting results, the algorithm converges after about 200 episodes, which is relatively fast, and the action predicted by the agent overlap with the BS-delta mostly. It can be said that the DDDQN algorithm solves this problem efficiently. In the second test we first define Index to calculate the pnl of our strategy and define performance metric to measure quality of the pnl. Then, VRP is harvested by defining a specific utility function in the DDDQN algorithm to reduce the Max Drawdown. In the training process of the algorithm, our training data is S&P500 option data from 2015-2018 and the test data is S&P500 option data from 2019-2021. It is worth to note that there are two significant benefits to selecting the 2019-2021 data as the test set. One is to test the performance of our strategy in out of sample, and the other is to test the performance of our strategy under a market crash(covid epidemic), or occurrence of turbulence. Ultimately, our algorithm converges well after fewer episodes and our pnl significantly reduces Max Drawdown and volatility. Our risk is reduced substantially, but our return is relatively not reduced a lot, as evidenced by the sharpe ratio, we gain almost double sharpe ratio compared to the constant target vega strategy.

There are many areas where our research could be improved. Since we trade a lot of call put options, we need to consider the bid-offer spread as well as the transaction cost. We can observe that there are many strategies work historically when these factors are not taken into account, so the addition of these two metrics can bring our strategy closer to reality, resulting in a more accurate PnL. We can also start to improve the model parameters of reinforcement learning, such as the number of episodes, the type of activation function, batch-size, and so on. Given our limited data, we only have 4 years of training data, and we can get more data, as well as clean and update the data to get high-quality training data. We can also test the performance of our strategy in the event of market crash such as the 2008 financial crisis.

In conclusion, our research combines a lot of mathematical proofs and coding, which is a major challenge, I need to develop deeply and rigorously into mathematical proofs and need to study the composition of the algorithm architecture to achieve our task. Furthermore, I need to optimise the data processing as well as the algorithm which greatly enhanced and honed my capabilities.

Appendix A

Technical Proofs

A.1 Leverage has no effect on Sharpe ratio

Sharpe ratio is define as

$$\frac{R-r}{\sigma}$$

where R is the return.

Assume that we leverage up n times. The return becomes $n \times R$ and

$$\sigma = \sqrt{Var(nR)} = n\sqrt{Var(R)} = n\sigma$$

At first glance, it seems our Sharpe ratio would change because r remains unchanged, but leveraging is not free, so r becomes $n \times r$. Thus, Sharpe ratio becomes

$$\frac{nR-nr}{n\sigma} = \frac{R-r}{\sigma}$$

Bibliography

[1] Wei Ge. A survey of three derivative-based methods to harvest the volatility premium in equity markets. *The Journal of Investing*, 25(3):48–58, 2016.

[2] Peter Carr and Liuren Wu. Variance risk premiums. *The Review of Financial Studies*, 22(3):1311–1341, 2009.

[3] Gurdip Bakshi and Nikunj Kapadia. Delta-hedged gains and the negative market volatility risk premium. *The Review of Financial Studies*, 16(2):527–566, 2003.

[4] Ioannis Karatzas, Steven E Shreve, Ioannis Karatzas, and Steven E Shreve. Stochastic differential equations. *Brownian Motion and Stochastic Calculus*, pages 281–398, 1988.

[5] Olivier Daviaud and Abhishek Mukhopadhyay. Linking the performance of vanilla options to the volatility premium. *Risk, July*, 2022.

[6] Sebastien Bossu, Eva Strasser, and Regis Guichard. Just what you need to know about variance swaps. *JPMorgan Equity Derivatives Report*, 4, 2005.

[7] David Silver. Lectures on reinforcement learning. URL: `https://www.davidsilver.uk/teaching/`, 2015.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[9] Lukas Gonon. Lecture notes in math70116 - deep learning 2022-2023, February 2022.

[10] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[11] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[12] fschur. Deep-reinforcement-learning-for-hedging. URL: `https://github.com/fschur/Deep-Reinforcement-Learning-for-Hedging`, 2021.