DELTA HEDGING OF FINANCIAL OPTIONS USING REINFORCEMENT LEARNING AND AN IMPOSSIBILITY HYPOTHESIS.

by

Ronak Tali

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Statistics

Approved:

Kevin R. Moon, Ph.D. Major Professor	Tyler J. Brough, Ph.D. Committee Member
Jia Zhao, Ph.D. Committee Member	Janis L. Boettinger, Ph.D. Acting Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2020

ABSTRACT

Delta hedging of Financial Options using Reinforcement Learning and an Impossibility Hypothesis.

by

Ronak Tali, Master of Science

Utah State University, 2020

Major Professor: Kevin R. Moon, Ph.D.
Department: Mathematics and Statistics

In this thesis we take a fresh perspective on delta hedging of financial options as undertaken by market makers. The current industry standard of delta hedging relies on the famous Black Scholes formulation that prescribes continuous time hedging in a way that allows the market maker to remain risk neutral at all times. But the Black Scholes formulation is a deterministic model that comes with several strict assumptions such as zero transaction costs, log normal distribution of the underlying stock prices, etc. In this paper we employ Reinforcement Learning to redesign the delta hedging problem in way that allows us to relax the strict assumption of risk neutrality and allows us to embed market realities such as transaction costs right at the outset. Our main argument is that by taking a controlled amount of risk and encouraging some uncertainty (referred to as exploration) in the hedged position, the market maker is able to generate incremental profit in the entire operation. Our model does not assume any parametric distribution for the underlying stock prices and is fundamentally online in nature i.e. learns on the go.

(91 pages)

To my grand parents, whom I owe everything.

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Kevin Moon for the continuous support and investment of time in my research process, for his patience, motivation, enthusiasm. His guidance has been invaluable for this research and writing of the thesis. I could not have asked for more from an advisor and mentor for my MS study.

I would also like to record my special thanks to Dr. Brough for motivating the problem and sharing his insights that helped me navigate through the complexities of the topic of research.

I would like to thank Dr. Zhao for being a part of my thesis committee and supporting this effort.

Finally, I would like to thank my friend Yogen Shah for his continous support and motivation especially in these testing times when we are all affected by the COVID shutdown.

Ronak Tali

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The world of option trading is run by the "market makers". Market makers are agents that underwrite and sell options to the buyers. In this thesis, we are concerned with the market makers perspective and how we can generate greater profit for the market maker without taking on too much risk. To achieve our goal, we show that it is possible to take the industry-standard Black-Scholes model of option pricing and use reinforcement learning to overcome some of that model's weaknesses ,and as a result, we can generate incremental profit for the market maker.

This thesis is structured in the following way:

1. Define and explain the fundamentals of financial options and option pricing.

2. Study in detail various historical models developed over the course of the last fifty years for option pricing.

3. Explain in depth the famous Black-Scholes model for option pricing by both looking at the mathematical underpinnings and discussing in detail the various assumptions and implications that arise out of this model.

4. Discuss various algorithms in reinforcement learning. We also modify the standard Monte Carlo reinforcement learning algorithm to create a new variant that we can use for our own delta hedging model.

5. Develop the framework and propose our own delta hedging model that overcomes some of the critical limitations of the Black - Scholes model and, in the end, show that we can generate incremental profit for the market maker.

1.1 Reinforcement learning in finance

While a large number of articles have been published in financial literature using the concepts of reinforcement learning, the vast majority have focused on two topics, forecasting future asset prices and portfolio optimization [1]. Also, most of the research in this area has focused on a limited number of reward functions such as maximizing the Sharpe ratio or minimizing the root mean square error(RMSE). This shows that there is plenty of room to explore other interesting areas of financial trading and research. Meng and Khushi [1] note that many simulations have suggested that reinforcement learning methods perform poorly when stock prices in the training period are markedly different from those in the test period. In this thesis, we therefore, normalize prices such models created using data from a certain time period remain valid for other test periods. Finally, they note that most studies in reinforcement learning tend to ignore friction factors such as trading costs, which could be a problem in case of high-frequency trading or large trading volumes. In this thesis, we address this issue by incorporating transaction costs to mimic real-world trading.

1.2 Basics of financial options

Options are defined as financial instruments that give a buyer of an option the right, with no obligation, to trade a given stock at a predetermined price (strike price) on or before a future date (Expiration date). Depending on the type of trade (buy or sell) and time of trade execution (at expiry or through any date including the expiration date), options are classified as follows:

1. European call option - The buyer can only exercise their right on the date of expiry to buy the stock at the strike price. The implicit assumption here is that the buyer would only do so if the actual price of the stock is greater than the strike price, allowing them to collect a net profit.

2. European put option - The buyer can only exercise their right on the date of expiry to sell the stock at the strike price. Similarly, the implicit assumption here is that

they would only do so if the actual price of the stock is lower than the strike price, allowing them to collect a net profit.

3. American call option - Similar to the European call option with just one important differentiation that the buyer of the option can choose to trade anytime before and including the expiration date.

4. American put option - Similar to the European put option with just one important difference that the buyer of the option can choose to trade anytime before and including the expiration date.

Financial options have the following characteristics:

1. Payoff: Defined as [Stock Price - Strike Price] for call options if the buyer chooses to exercise the option and [strike price - stock price] for put options. Logically the payoffs are never negative. The nature of payoffs can be plotted as a function of the stock price and the strike price, as in Figure 1.1.

2. Value of an option: Depends on the volatility. Further, the value of an option goes up as volatility goes up and vice versa.

3. Price processes: We assume that the underlying stock price of the option follows a predetermined price process such as Geometric Brownian Motion. This is necessary, so that seller of the option formulates some measure of risk associated with selling the option.

4. No Arbitrage: Though debatable in practice, most option pricing models assume the absence of arbitrage opportunities a necessary condition in order to simplify their model formulations. Arbitrage is defined as an opportunity that exists in a market, perhaps only momentarily, to buy an asset at a certain price and sell it in a different market for a higher price, thereby making money risk free by taking advantage of the price difference.

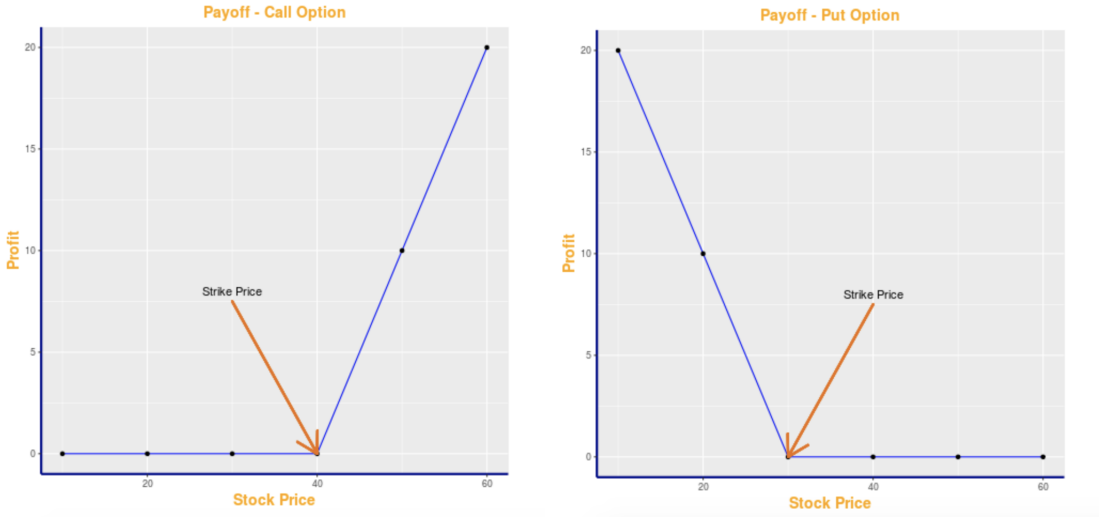

Fig. 1.1: Payoffs ($) from call and put options.

Figure 1.1 shows the range of possible payoffs for the buyer of the option generated from European call and put options respectively. It can be seen that a call option generates a positive payoff for the buyer only when the stock price at expiration exceed the strike price of the option and vice versa for put options.

1.3 The market makers perspective

Most financial options in the stock markets are executed by agents called "market makers". The role of a market maker is to "take the other side of the retail trade". In other words, when a retail trader wants to buy an option X in the stock market, for that option trade to be possible, a market maker will have to "write" an option contract and assume all the risks associated with the writing of the option contract. As a result of this process, the market maker assumes different kinds of market risks during the life cycle of X ,and in return charges, an upfront premium, akin to the idea of a premium charged to the buyer of an insurance policy, to the retail trader.

Using the call option to explain the concepts discussed in Section 1.2, here are some standard notations:

- Strike price = K
- Expiration date = T
- Stock price at any time $t \leq T = S_t$
- Payoff for retail investor = Π_{inv}
- Payoff for market maker = Π_{MM}
- Upfront premium paid the option buyer = p
- Transaction cost for the market maker = C_X
- Trading cost for the market maker = C_T

The retail trader makes a profit Π_{inv}, such that

$$\Pi_{inv} = S_T - K - p.$$

Therefore, the retail trader purchasing a call option expects the price of the underlying asset (stock, ETF, bond, commodity, etc.) to exceed the strike price K at the time of expiry T of the option contract. This phenomenon is often referred to as "In the Money" in financial literature. On the contrary if at time of expiry T, $S_T < K$, the retail trader makes a loss given by $\Pi_{inv} = -p$. This phenomenon is often referred to as "Out of the Money".

The market maker also expects a profit, but they have a slightly more complicated formulation. Similar to the retail trader, we have to consider both the cases i.e., when the option ends up in the money and when it ends up out of the money.

For in the money options:

$$\Pi_{MM} = p - (S_T - K) - C_T - C_X.$$

For out of the money options:

$$\Pi_{MM} = p - C_T - C_X.$$

The significance of trading costs and transaction costs

Trading cost (C_T) is incurred when the market maker continuously rebalances (i.e., hedge) their portfolio in response to the volatility of the underlying asset. This act of rebalancing is critical to the study of options. In the process of rebalancing, a market maker is constantly buying and selling the underlying asset. Selling low and buying high is the cardinal mistake in stock trading (simulations over the years have suggested that this is the precise reason why individual retail investors lose money over time because they tend to sell low and buy high, especially during periods of high volatility, such as the 2008 market crash). Market makers are subject to this same threat of losing money as they constantly try to rebalance the position of the underlying asset held by them.

Transaction Cost (C_X) constitutes all costs that the market maker incurs to maintain a hedged position. These include interest on borrowed money, commissions, regulatory fees, etc. While these are typically smaller than the trading reward C_T, they are not negligible. Market makers borrow money from banks to hold underlying asset positions. Hence they are incurring a daily interest cost because of holding any asset positions. It is fair to assume that the interest rates of borrowing are fixed, at least over short periods of time. As a result, we can assume them to be constant while developing the algorithms. We denote interest rates by r in this thesis. Again, in most financial literature, these costs are usually ignored.

In this thesis, we are looking at the market makers problem, which is how to maximize profits in the presence of all kinds of market forces. Putting all the concepts we have discussed so far together, we note that if the market maker were to find a way to rebalance their asset position held against the option (delta hedge) that maximizes their trading cost minus transaction cost i.e., $C_T - C_X$, they would show that it is possible to generate incremental profit in the entire operation when compared to another market maker that is using only Black-Scholes for delta hedging.

We now clearly state the lifecycle of a financial call option from a market makers perspective:

1. Underwrite an Option Contract O for an asset X with a Strike Price K and expiration time T for a retail trader and collect a premium p
2. Against the sold option O, hold Δ units of X such that $0 < \Delta \leq 1$ using a Delta Hedging Algorithm for every time t such that $0 \leq t \leq T$
3. Incur trading costs C_T^t and transactions cost C_X^t at each time t such that

$$C_T = \sum_{t=1}^{T} C_T^t$$

$$C_X = \sum_{t=1}^{T} C_X^t$$

4. Meet any obligations at contract expiry and liquidate all asset positions realizing a final profit Π_{MM}^{+} or loss Π_{MM}^{-}

1.4 Review of Selected Literature

In this section, we will discuss some historical option pricing models and their significance. We will be discussing the various assumptions, limitations, and implications of these models. We note that our goal is to overcome some critical limitations that we will use to design our own model to perform delta hedging.

1.4.1 Black Scholes Merton Formulation

This Nobel Prize winning idea was originally proposed by Fisher Black and Myron Scholes [2] in 1973 and provided perhaps the first rigorous treatment of valuing risks in options. Despite its several limitations, this model is still used widely in the industry to manage options. The following are some of the key observations upon which the authors formulated their model.

Under the assumption of constant volatility:

1. Higher the price of the asset compared to the strike price of the option, greater the value of the option. This is because of the probability that the stock price at time t, denoted as S_t, exceeds the strike price K, is greater. I.e. $P[S_t > K]$ is monotonically increasing with increasing S_t.

2. An option value is more volatile than the underlying stock. For a given expiration time T, a change in the stock price of the underlying asset will result in a dilated change in option value.

3. The value of an option declines with time. For maturity farther into the future, the value of an option is nearly equal to the current stock price but, as we move closer towards maturity, the value of the option declines.

4. And finally the no-arbitrage rule. For a correctly priced option, it is impossible to find a combination of long and short positions that leads to arbitrage opportunities.

Black & Scholes made several other restrictive assumptions that never meet the realities of option trading. Yet, their model empirically performs well except for the known situation where the underlying asset of the option is experiencing high volatility.

1.4.2 Cox, Ross and Rubinstein (1979) Binomial Option Pricing Model

In their highly influential 1979 paper "Option Pricing - A simplified Approach", Cox, Ross, and Rubinstein [3] presented, "a simple discrete-time model for valuing options. They note in this paper that the original Black- Scholes model "is a special limiting case". This paper, despite its very simplistic treatment, works pretty well in most scenarios. Following are some key ideas of the CRR model:

1. Underlying asset price, S_t follows a multiplicative binomial process over discrete time periods.

2. The rate of return on S_t can assume only two values over a single discrete time step. With probability q, $S_{t+1} = uS_t$, or, with probability $(1-q)$, $S_{t+1} = dS_t$. Here $u-1$ is the rate of return over a single time step in upward direction and $d-1$ is the rate of return over a single time step in the downward direction.

3. At every time step, the market maker holds a combination of underlying asset Δ and B units of a risk free bond for optimal hedging.

4. The model calculates that under constant interest rate r , no-arbitrage, and absence of transactions costs, optimal single period values of Δ and B as follows:

$$\Delta = \frac{C_u - C_d}{(u-d)S_t}$$

and,

$$B = \frac{uC_d - dC_u}{(u-d)r}$$

Here C_u is the payoff from a call option at expiry if the stock price of the underlying stock goes up in future (single time step), and similarly C_d is the corresponding payoff if the market instead goes down.

5. For multi period formulations, the above formula holds but the values of C_u and C_d are calculated by calculating the conditional expectation (under a steady state transition matrix) of higher order values such as C_{uu}, C_{ud}, C_{dd}, etc. Here e.g. $C_{uu} = u^2 S_t$ and so forth.

The authors also show that as t is divided into minute sub-intervals, the CRR model approaches the Black - Scholes formulation and the multiplicative binomial distribution of S_t approaches the Log-Normal distribution.

1.4.3 Clewlow and Hodges

Clewlow and Hodges [4] is one of the few studies in the area of delta hedging that takes into account the transaction costs but not the trading rewards of options. The authors solve an optimal control problem akin to classical quality control formulations that plot different Control Limits UCL and LCL on the control chart. In this model, rebalancing happens when the delta of the underlying Asset x_t (as suggested by Black - Scholes) crosses these control bands in either direction. Essentially the authors are creating the equivalent of a high pass filter. The core idea is that by providing these restrictions, the frequency of rebalancing can be reduced, and with that, we can also expect the transaction costs C_X will also decline over the entire option period. It is critical to note that the width of these control bands needs to be tuned to control the risk at any time t. Following are the main ideas of the model:

1. The authors define a surplus function for the market maker defined at maturity T

$$w_T = y_T - C(S_T)$$

   Here,

   y_T : Cash generated by liquidating all positions at maturity T

   $C(S_T)$: Liability of market maker at maturity T i.e., if the call option is in the money, the market maker will have to have to pay the contract holder.

2. The optimal control problem seeks to maximize this surplus. But to give more structure to this problem, the authors rather maximize a negative exponential utility function of w_T given by

$$U(w_T) = -exp(-\lambda w_T)$$

   Here λ is a parameter that defines the degree of risk aversion of the market maker. Bigger the λ, the lesser the risk aversion.

3. With this set up, the authors define a special cost function that encapsulates the transaction costs associated with hedging at each time step.

4. Finally, the authors derive the control limits discussed previously in the form of x_+ and x_-.

5. The authors, therefore, prescribe the following delta hedging strategy:

   If underlying delta of the asset x_t exceeds x_+ or recedes x_-, trigger a re-balancing. Otherwise, do nothing.

6. Using this simple rule, a low pass filter is created that reduces transaction costs without taking on too much risk.

One critical assumption of this model is that of constant drift μ^* in the underlying asset price S_t over time. This limits the ability of this model to remain responsive to shocks. Nevertheless, this model gives us the important idea that we can choose not to do anything at certain times to optimally hedge a portfolio.

1.4.4 Heston and Nandi - Closed-Form GARCH Model

In the paper "Closed-Form GARCH Option Pricing Model," Heston and Nandi [5] were able to replace implied volatility (volatility inferred from the extant price of an option) used in the Black-Scholes model with actual estimates of volatility by assuming that the volatility of an asset follows the GARCH process. Before we proceed, it is worthwhile to briefly touch upon the theory of GARCH processes.

Generalized Autoregressive Conditional Heteroskedasticity (GARCH) are a class of Time Series models that are used to model non-constant volatility in a stochastic process. GARCH models have become extremely popular in both economics and finance because of this ability. GARCH processes are specified by having both a conditional mean as well as a conditional variance equation. GARCH significantly improves upon ARCH by adding lagged volatility terms to ARCH's lagged residual error terms. This modification allows GARCH

to handle heteroscedasticity and volatility clustering in a stochastic process. GARCH (1,1) seems to be the most popular model currently being used in analyzing financial time-series data (GARCH(1,1) models are time series models that model a value (a_t) as function of value yesterday (a_{t-1}) as well as volatility yesterday (σ_{t-1}) i.e. $a_t = f(a_{t-1}, \sigma_{t-1})$).

As we briefly discussed, Heston and Nandi's cleverly modified Black-Scholes by embedding GARCH as a running volatility estimator as opposed to using the notion of Implied Volatility. Another significant innovation in this paper was finding a closed-form solution to GARCH at each time step.

Some of the key ideas in this model are as follows:

1. The paper assumes that a GARCH(1,1) process is appropriate to model volatility.
2. The value of a Call option one-time step before expiration obeys the Black-Scholes solution.
3. The paper modifies the Black - Scholes equation by adding to Asset Price at time t, S_t an additional conditional variance term $h(t + \Delta)$
4. $h(t + \Delta)$ is computed by the observed path of Asset Prices S_t. This removes the requirement to use implied volatility and instead uses the history of observed asset prices to directly compute the conditional variance term.

The authors observe that while GARCH models almost always outperform the Black - Scholes formulation, they do not perform very well with short period options. Nonetheless, to this model's credit, it significantly builds on the original ideas of Black-Scholes by employing the power of a GARCH(1,1) process to automatically estimate volatilities from previously observed data.

1.4.5 Longstaff and Schwartz(LS algorithm)

Longstaff and Schwartz [6] originally developed their framework to price American options; however, we have used their key ideas to modify their model to hedge European options so that we can compare their results with Black-Scholes as well as with the model that we have proposed. The key ideas of their model are as follows:

1. An American option is exercised if the payoff from immediate exercise is greater than the expected conditional payoff resulting from continuation.

2. The conditional expectation can be computed using cross-sectional data generated from multiple simulated paths using ordinary least squares(OLS).

3. Choosing appropriate basis functions improves the accuracy of the least-squares fitting.

4. The idea can be extended to price even more exotic options such as Bermudan or Asian options.

The implementation of this algorithm can be expressed in the form of the following steps:

1. Simulate various option paths and for each path compute the discounted cash flow.

2. Choose only those paths that end "in the money".

3. Choose a set of basis functions for the inputs and regress discounted cash flow from step - 1 against these basis functions. Denote the fitted value as $\hat{F}(\omega; t_{K-1})$.

4. It can be shown that in theory that $\hat{F}(\omega; t_{K-1})$ approaches $F(\omega; t_{K-1})$ in mean square and probability.

5. It can also be proved that $\hat{F}(\omega; t_{K-1})$ is best linear unbiased estimator (BLUE) of $F(\omega; t_{K-1})$.

The authors also establish that value of an American option V(x) obeys the following convergence property:

$$\lim_{N\to\infty} P\left[\left|V(x) - \frac{1}{N}\sum_{i=1}^{N} \underbrace{LS(\omega_i; M, K)}_{\text{Cash flow using LS algo.}}\right| > \epsilon\right] = 0$$

Here M indicates the number of basis functions used. Obtained through grid search, K indicates the number of available discrete exercise times. N is the total number of simulated paths.

Adapting the LS algorithm to hedge call options:

1. At each discrete time point $k \in K$, the LS algorithm calculates an exercise price. Over multiple k, we get an exercise boundary.
2. This boundary indicates that if the stock price is below exercise price, any delta hedging will lead to undervaluation and that we can expect a market correction that will drive the stock prices up. To avoid selling low and later buying high, we should hold off any delta hedging.
3. Create a decision rule such that we hedge to the Black-Scholes delta if the current stock price is at or above exercise boundary and do nothing otherwise.

1.4.6 Carr and Madan - Fourier transform

Carr and Madan [7] use the following key ideas to construct their model:

1. The characteristic function of the risk-neutral density of the logarithm of the stock price is known.
2. The risk-neutral density can also be used to calculate the option price.
3. The option price from above needs to be modified to make it square-integrable such that a Fourier transform can be taken and create an analytical expression for the option price.

4. We then take an inverse Fourier transform of the above analytical expression giving us the numerical option price.

The sequence of steps in their derivation is as follows:

1. Characteristic function - Let s represent the support for stock price and let q_T represent the log density of stock price over the support s. Thus characteristic function can be represented by:

$$\phi_T(u) = \int_{-\infty}^{\infty} e^{ius} q_T(s) ds$$

2. Option price as a function of the density - Let k denote the density of logarithm of the strike price. Assuming a risk-free rate of r, we have

$$C_T(k) = \int_{k}^{\infty} e^{-rT}(e^s - e^k) q_T(s) ds$$

3. Making option price square-integrable for Fourier transform by assuming a $\alpha > 0$ such that:

$$c_T(k) = e^{\alpha k} C_T(k)$$

4. Take Fourier transform

$$\psi_T(v) = \int_{-\infty}^{\infty} e^{ivk} c_T(k) dk$$

Solving the above expression we get:

$$\psi_T(v) = \frac{e^{-rT}\phi(T)(v - (\alpha + 1)i}{\alpha^2 + \alpha - v^2 + i(2\alpha + 1)v}$$

5. Finally, to get the numerical option price we take the inverse Fourier transform given by:

$$C_T(k) = \frac{e^{\alpha k}}{\pi} \int_{0}^{\infty} e^{ivk} \psi(v) dv$$

1.4.7 Artificial Neural Networks (ANN)

While ANNs have been used to replicate various option pricing models, including one such implementation to replicate the Black-Scholes formulation by the author, Liu et al. [8] take a more general approach by replicating not just an option pricing model but also modeling a hyper-parameter such as implied volatility using ANNs. As the paper notes, one of the reasons behind using ANNs is the computational speed while generating predictions.

The general approach to using an ANN for option pricing is as follows:

1. Either use observed data or generate simulated data such that we have a dataset that maps a set of input parameters to the corresponding output variable e.g., the option price.
2. Split the above data in a predefined proportion and randomly to create a training and test set.
3. Train the ANN using the training set generated in step 2.
4. Evaluate the ANN using the test set generated in step 2.
5. Use the trained ANN instead of the original option pricing model.

ANN implementations are available out of the box in several programming languages. One of the key contributions of the paper is that the authors have trained multiple ANNs using hyper-parameter search and were able to make some important recommendations about the ideal parameters for the option pricing problem. Some of these observations are listed below:

1. Fully connected deep network (4 layers) with a large number of nodes per layer (400) - This indicates that we are solving a highly non-linear problem.
2. Large batch size (1024).
3. Low learning rate between 0.001 and 0.00001.

4. Little over-fitting even with no regularization.

5. Very high degree of accuracy. Average error over a large number of simulations is only about 0.009%

Note: The authors have run simulations to replicate a large number of option pricing models, and in all cases, they were able to get really low errors on a test dataset.

1.4.8 Conic option pricing

Madan and Schoutens [9], in this paper, show that by abandoning the law of one price that is based on the classical economics of matching demand with supply and assuming instead a law of two prices that allow for selling for example at a lower price and buying at a higher price, the risk set of the option is contained in a convex cone that has non-negative random variables. The end goal is to create two different valuations of the options i.e., lower and upper valuation, and then computing the suitably weighted average (midquote) that gives a mixture risk-neutral model. To understand this idea more clearly:

Let $L(C)$ and $U(C)$ denote the lower and upper valuations resulting from the law of two prices. These can be represented mathematically as:

$$L(C) = \inf_{Q \in M} E^Q(C)$$

$$M(C) = \sup_{Q \in M} E^Q(C)$$

Here M represents the family of probabilities that separate the non-negative claims from a set of zero-cost traded claims. Moreover, we can use a distortion function $\Phi(u), 0 \leq u \leq 1$ such that for any random variable X, we can get $L(X) < E(X) < U(X)$.

The authors go on to calculate the expressions for $L(C)$ and $U(C)$ given by:

$$L(C) = \int_K^\infty \hat{\Psi}\Big(F\Big(log\Big(\frac{s}{S_0}\Big)\Big)\Big)ds$$

and

$$U(C) = \int_K^\infty \Psi\Big(F\Big(log\Big(\frac{s}{S_0}\Big)\Big)\Big)ds$$

Here, $\hat{\Psi}(F) = 1 - \Psi(F)$ and Ψ is the distortion function that we defined previously. s is the support for stock price and F is the CDF function.

The authors then go on to use these L and U to create a fair mid quote as a log weighted average of L and U.

1.5 Chapter Summary

In this chapter, we introduced the relevant concepts with regard to the fundamentals of options. We also discussed the work done in almost fifty years in this area. Through the rest of the thesis, we will use many of these concepts to develop our own algorithm to perform delta hedging on financial options.

CHAPTER 2

BLACK - SCHOLES & DELTA HEDGING OVERVIEW

2.1 The Theory behind Black-Scholes

In this chapter, we will look at the Black-Scholes model in detail. There are several reasons to do so:

1. The path-breaking model has a rich history associated with it.
2. The model captures most of the fundamentals that drive the options market. Hence knowing this model well ensures we have understood the various market forces at play.
3. This thesis builds on the ideas of Black & Scholes and uses several of the concepts used by Black & Scholes in their seminal paper.

2.1.1 A Brief Sojourn into CAPM

The Capital Asset Pricing Model (CAPM) [10] suggested around the mid-1960s, is a foundational idea in Mathematical Finance. CAPM gives us the expected return that a risky asset will generate given the prevailing interest rates and the relative risk (i.e., covariance with the broad market e.g., the S&P 500) associated with that asset. This model, despite its simplicity, is still used widely in industry to calculate the "Net Present Value" (NPV) of an asset in a given portfolio. This notion of relative risk is widely referred to as the "Beta of an Asset" and is denoted by β. Assets that are more volatile than the reference market have $\beta > 1$ and vice versa. In later sections we will test our model on ETFs that have $\beta > 1$, $\beta < 1$ and $\beta = 1$.

2.1.2 The History of Black Scholes

Duffe [11] wrote in the Scandinavian Journal of Economics about the significance of the Black - Scholes model in modern mathematical finance. He quotes some sources to throw some light on the history of how this famous idea came to be.

The idea was to apply the Capital Asset Pricing Model (CAPM) at every time step during the option period, so that we can value the option in continuous time. Fisher Black observed [11] that as time steps decrease in size, it is possible to define a stochatic PDE for the option price $c(x,t)$ at any time $t < T$ at a fixed interest rate r given by:

$$c_t(x,t) + c_x(x,t)rx + \frac{1}{2}c_{xx}\sigma^2x^2 = rc(x,t),$$

where subscripts t, x in the equation indicate the partial derivatives. The above equation has the familiar boundary condition (i.e. defined option payoff) at expiration time T given by:

$$c(x,T) = max\{x - K, 0\},$$

where K is the strike price of a call option. It is interesting to note that while Fisher Black proposed this PDE around 1969, he could not solve it. We note here that the PDE does not contain the Expected Return from an asset μ, indicating that μ does not play a role in evaluating the value of the option.

Black subsequently collaborated with Scholes and Merton, and together they realized that if both the underlying asset and the associated option can be treated to have a riskless rate of return, then over infinitesimal intervals of time, the movement of the asset price and the value of the option are perfectly correlated and that the stochastic PDE has a solution of the form proposed by Sprenkle in 1961 [12]. But to ensure a continuously riskless rate of return, one must hold a hedged position between the asset and the option, which is now popularly known as the delta of an option denoted by Δ. It is worthwhile to note that this seminal paper was originally rejected twice only to be eventually accepted by the Journal

of Political Economy for their Spring 1973 edition after Eugene Fama and Robert Merton himself wrote detailed reviews of the paper, highlighting its breakthrough importance [11].

2.1.3 Stock prices as Itô processes

Black, while developing his original idea, modeled asset price as following an Itô process with the following form:

$$dS_t = \mu(S_t, t)dt + \sigma(S_t, t)dB_t$$

Here, the term dB_t comes from an independent increments property of Brownian motion and by definition $dB_t \sim N(0, dt)$ and thus has the property that $E[dB_t] = 0$

Taking Expectations on both sides of the equation and dividing by S_t, under the critical assumptions that both the drift and volatility of the asset is constant, we have

$$E\left[\frac{dS_t}{S_t}\right] = \mu dt.$$

We can thus express:

$$\mu(S_t, t) = \mu S_t$$

and

$$\sigma(S_t, t) = \sigma S_t$$

Therefore, we can express asset price at any time t as,

$$\frac{dS_t}{S_t} = \mu dt + \sigma dB_t.$$

The solution to this Brownian motion using Itô's Lemma indicates that S_t is Log-Normally distributed [13].

2.1.4 Key assumptions used for solving the Black - Scholes PDE

1. The asset price follows the Geometric Brownian Motion model.
2. The risk-free interest rate r and the asset volatility σ are known *apriori*.
3. Absence of transaction costs.
4. The asset pays no dividends during the life of the option.
5. There are no arbitrage opportunities.
6. Trading of the asset takes place in continuous time.
7. Short selling is allowed.
8. We can buy or sell any number of units (including fractions) of the asset.

2.1.5 Derivation of the solution to the Black-Scholes PDE

We start by assuming that the Value of an option $V(S,t)$ is dependent only on S and t [14]. Also assuming that $\frac{\partial V}{\partial t} \in C^1$ and $\frac{\partial V}{\partial S} \in C^2$ (set C^1 indicates that first derivative exist and is continuous. Similarly, set C^2 indicates that both first and second derivatives exist and are continuous.), and applying the Itô's Lemma we have:

$$dV = \left(\mu S V_s + \frac{1}{2}\sigma^2 S^2 V_{ss} + V_t\right) dt + (\sigma S) V_s \cdot dB_t$$

As we discussed in the previous section, the idea is to construct a hedged position $-1 < \Delta < 1$ so that the rate of return on the option becomes riskless, and the above equation becomes solvable. Thus, for a call option, we can write the value of the hedged position Π as

$$\Pi = V - \Delta \cdot S.$$

Taking a time derivative of the above expression gives

$$d\Pi(t) = dV - \Delta dS.$$

Making substitutions for dV and the Brownian motion evolution of asset price dS_t:

$$d\Pi(t) = \left(\mu S V_s + \frac{1}{2}\sigma^2 S^2 V_{ss} + V_t - \mu S\right) dt + \sigma S(V_s - \Delta) dB_t.$$

We make some important observations here.

1. In the equation above, the first term is deterministic, and the second term is stochastic because of the Brownian motion's evolution term dB_t.
2. If we set $\Delta = V_s$ in the second term, we can get rid of the entire stochastic term, i.e., the market maker can operate risk-free albeit in an idealized world with no arbitrage opportunities.

These observations form the crux of the idea behind delta hedging.

We also notice that because we have whittled away all risks from our hedged position, at least in theory, we can safely claim that the return on an investment Π over a time period dt can be expressed as $r\Pi dt$. This return is risk-free. It also means that an arbitrage opportunity presents itself in a case $d\Pi(t)$ (that we just derived) is significantly different (above a certain tolerance) from $r\Pi dt$. Since the Black-Scholes model assumes no-arbitrage, we can say that

$$d\Pi(t) \approx r\Pi dt.$$

Now we are going to substitute back $\Pi = V - \Delta S$ and $\Delta = V_s$ to reach the final form that we discussed, but with a slightly different notation in our introduction to this chapter i.e.

$$V_t + \frac{1}{2}(\sigma S)^2 V_{ss} + rSV_s = rV.$$

This is the final form of the celebrated Black-Scholes partial differential equation.

It is pertinent to make a few consequential remarks:

1. To avoid taking any risks at all, the market maker needs to hold Δ units of the underlying asset for every option sold at all times. At expiration, however, for a call

option, the market maker needs to hold $\Delta = 1$ if the option is in the money or $\Delta = 0$, if it's out of money. In this thesis, one of our main goals is to refute this idea that the market makers need to be risk-neutral at all times. Instead, we argue that by taking a controlled amount of risk, the market maker can create an arbitrage opportunity at best and avoid large losses at worst.

2. Of particular interest here is the differential operator given by

$$L_{BS} = \frac{\partial}{\partial t} + \frac{1}{2}(\sigma S)^2 \frac{\partial^2}{\partial S^2} + (rS)\frac{\partial}{\partial S} - r.$$

The operator signifies the difference in returns between our hedged position and a risk-free bond. Because of the no-arbitrage principle, this value should be ≈ 0. However, in this thesis, we will show that by taking a calculated amount of additional risk, we can generate incremental returns from our hedged position, compared to say investing in a risk-free treasury bond.

3. Finally, we do not have a drift term for the asset μ. This means that our expectation of μ has no bearing on the value of an option. Further, two different market makers with different expectations of μ would still value the option identically, everything else remaining the same.

2.1.6 Discussion on the boundary conditions of the Black - Scholes PDE

It is important to note that the Black - Scholes PDE, has many possible solutions. However, it is critical for a market maker to get a unique value for the model to be usable. Hence we need to impose some boundary conditions to make sure that we are able to find unique solutions that are meaningful.

The task of establishing the boundary conditions become easier once we realize that the Black - Scholes PDE is indeed a Backward Parabolic PDE and resembles in the form to the classical heat equation (which is a forward-parabolic PDE originally proposed by

Fourier that represents the mechanism of heat flow in a metal rod) given by:

$$u_t = u_{xx} \quad \forall t > 0.$$

The usual boundary conditions for the heat equation are:

$$u(x, 0) = u_0(x) \quad \forall -\infty < x < \infty,$$

i.e. We know the temperature of the rod at time $t = 0$ and

$$u(-\infty, t) = u_-(t) \quad u(\infty, t) = u_+(t) \quad \forall t > 0,$$

i.e. we are able to measure the temperature at the ends of the rod at all times.

Comparing this with the Black-Scholes PDE, similar to the case of a standard heat equation, two conditions are commonly imposed on S and one condition on t of the form

$$V(S, t) = V_a(t) \ \ at \ \ S = a,$$

$$V(S, t) = V_b(t) \ \ at \ \ S = b.$$

$V_a(t), V_b(t)$ are known functions of time t known *apriori*. And finally,

$$V(S, T) = V_T(S).$$

Again, $V_T(S)$ is a known function of S.

We will now take the example of a call option and define these boundary conditions in a more concrete manner. Since this is a backward evolving PDE, we start with the terminal boundary condition, i.e., the option payoff at expiry T given by:

$$c(S, T) = max\{S - K, 0\} \quad \forall S > 0.$$

Here K is the strike price of the call option. We next look at the non triviality condition. i.e. Consider the case $S = 0$:

$$c(0,t) = 0 \quad \forall t > 0.$$

Finally we look at the condition $S \to \infty$. With very large asset prices, it becomes almost certain the option will end up in the money and that the Market Maker will have to payout. This condition yields our final boundary condition:

$$c(S,t) \sim S - Ke^{-r(T-t)} \quad , S \to \infty.$$

Adding these boundary conditions, we are now in a position to solve the Black-Scholes PDE and derive unique solutions along the path of the option. This is a very important result [14] because it ensures the availability of a solution all through the option time frame.

2.2 Solution of the Black-Scholes Equation

We will try to solve the Black-Scholes PDE using Dunbar's [15] interpretation of J Michael Steele's approach. We start by setting

$$t = T - \frac{2\tau}{\sigma^2} \quad and \quad S = Ke^x,$$

such that we can now write

$$V(S,t) = Kv(x,\tau).$$

Let's calculate the various partial derivatives associated with the original PDE

$$V_t = -(\frac{\sigma^2}{2})Kv_\tau,$$

$$V_s = \frac{K}{S}v_x,$$

$$V_{ss} = -\frac{K}{S^2}v_x + \frac{K}{S^2}v_{xx}.$$

Similarly the terminal boundary condition takes the following form

$$v(x,0) = max\{e^x - 1, 0\}.$$

Making substitutions into the original PDE, we get,

$$v_\tau = v_{xx} + (k-1)v_x - kv \quad where \quad k = \frac{2r}{\sigma^2}.$$

Looking closely at the above equation, we notice a few things:

1. The equation has constant coefficients.
2. $-\infty < x < \infty$.
3. k is dimensionless and represents ratio of interest rate to volatility.
4. Rescaled time to expiry is denoted by $\frac{\sigma^2}{2}T$.

After final substitutions we can get to the following form of the heat equation:

$$v = e^{\alpha x + \beta \tau} u(x, \tau).$$

Here α and β are constants that depend on k. Thus,

$$v_x = \alpha e^{\alpha x + \beta \tau} u + e^{\alpha x + \beta \tau} u_x,$$

and,

$$v_{xx} = \alpha^2 e^{\alpha x + \beta \tau} u + 2\alpha e^{\alpha x + \beta \tau} u_x + e^{\alpha x + \beta \tau} u_{xx}.$$

Substituting this into the simplified form of our PDE, we get,

$$u_\tau = u_{xx} + (2\alpha + (k-1))u_x + (\alpha^2 - (k-1)\alpha - k - \beta)u.$$

Now we can see that by setting

$$\alpha = \frac{k-1}{2} \quad , \quad \beta = \alpha^2 - (k-1)\alpha - k,$$

we have reached the heat equation

$$u_\tau = u_{xx}.$$

Observing the initial boundary condition

$$u(x,0) = max\{e^{\frac{(k+1)x}{2}} - e^{\frac{(k-1)x}{2}}, 0\},$$

we invoke the standard solution of the Heat Equation and continue the calculations, reaching the final form of the Black-Scholes solution:

$$V(S,t) = S\Phi(d_1) - (Ke^{-r(T-t)})\Phi(d_2),$$

where,

$$d_1 = \frac{log(\frac{S}{K} + (r + \frac{\sigma^2}{2})(T-t))}{\sigma\sqrt{T-t}},$$

$$d_2 = \frac{log(\frac{S}{K} + (r - \frac{\sigma^2}{2})(T-t))}{\sigma\sqrt{T-t}}.$$

Note that

$$d_2 = d_1 - \sigma\sqrt{T-t}.$$

This is the final **closed form** solution of the Black - Scholes PDE.

2.3 Greeks

In this section, we will again be following Dunbar's interpretation [16] of the various Greek Symbols associated with the Black - Scholes PDE. The various "Greeks" represent the sensitivity of the PDE to its parameters. The knowledge of sensitivity is critical for the market makers because by monitoring the sensitivity of the Greeks, the market maker is able to make important decisions with respect to hedging against the option.

2.3.1 Delta(Δ)

We had observed during the formulation of Black-Scholes that to take the risk out of the equation; we had to set

$$\Delta = V_s.$$

Now that we have a closed form solution to the PDE, we can take derivatives of $V(S,t)$ to give us a closed-form solution to Δ.

$$\Delta = \Phi(d_1) + \underbrace{[S\Phi^{'}(d_1)](d_1)_s - [(Ke^{-r(T-t)})\Phi^{'}(d_2)](d_2)_s}_{=\ 0}.$$

Following are some key observations about Δ:

1. Since $0 < \Phi(d_1) < 1$, $\Delta > 0$ for a call option.
2. As the stock price S_t increases, Δ also increases and vice versa.

Formally Δ is defined as the rate of change of the value of an option w.r.t the price of the underlying asset. In this thesis, our central goal is to define a control policy for Δ such that we are able to generate incremental profit out of holding a hedged position. Therefore the original idea is to hold a hedged position [16], ΔS_t called the "Hedge Ratio" at all times so that the market maker remains risk-neutral at all times during the option period.

2.3.2 Gamma(Γ)

Gamma(Γ) is also referred to as the "Convexity Factor" and indicates the sensitivity of Δ to underlying asset price S_t. It is given by:

$$\Gamma = V_{ss}.$$

Gamma is an indicator of how frequently a market maker needs to rebalance his Hedge Ratio. The idea being that if Γ is large, Δ is highly sensitive to S_t and vice versa. Formally, Γ can be shown to be:

$$\Gamma = \frac{1}{S\sqrt{2\pi}\sigma\sqrt{T-t}} e^{-\frac{d_1^2}{2}}.$$

$\Gamma > 0$, which means it concave w.r.t S_t.

2.3.3 Theta(Θ)

Theta(Θ) represents the time derivative of the option value function. It is represented as:

$$\Theta = V_t.$$

We can solve for Θ, and it has the following form:

$$\Theta = -\frac{S\sigma}{2\sqrt{T-t}} \cdot \frac{e^{-\frac{d_1^2}{2}}}{\sqrt{2\pi}} - rK[exp(-r(T-t))]\Phi(d_2).$$

A few important observations about Θ:

1. For a call option, $\Theta < 0$. Thus the value of the option diminishes as we approach maturity.

2. We don't hedge against time because it varies linearly and we can't control it.

2.3.4 Rho(ρ)

Rho(ρ) indicates the change of value of an option w.r.t a change in the interest rate r. It is formally given by:

$$\rho = K(T-t)[exp(-r(T-t))]\Phi(d_2).$$

Following are some important properties of ρ:

1. $\rho > 0$
2. As interest rates go up, the value of the option also goes up. The intuition here is that by applying the Black - Scholes equation, we are able to earn money as if we were investing in a risk-free bond. In this regard, it is easy to argue that the higher the interest rate, the higher the payoff for the market maker by holding the option for time T.

2.3.5 Vega(Λ)

Vega(Λ) is an indicator of the rate of change of the value of an option w.r.t the change in the underlying volatility of the asset. Formally, Λ is given by:

$$\Lambda = S\sqrt{T-t} \cdot \frac{e^{-\frac{d_1^2}{2}}}{\sqrt{2\pi}}.$$

Following are some important properties of Λ:

1. $\Lambda > 0$.
2. An increase in volatility gives rise to an increase in the value of the option, provided there are no transaction costs. In the presence of friction factors, any value gains can be eaten away by trading losses and transaction costs.

2.3.6 An Alternative Formulation of Black - Scholes PDE using Greeks

It is interesting to note that the Black - Scholes PDE can be expressed in the form of the first three Greeks that we just described. This special form is as follows:

$$\Theta + rS\Delta + \frac{1}{2}\sigma^2 S^2 \Gamma = rV.$$

2.4 Criticisms of Black - Scholes

With all its fame and glory, the Black - Scholes algorithm came under heavy criticism during the financial crash of 2008-09 [17]. Some quarters have explicitly blamed Black-Scholes as the very basis that financial engineers used to construct complex derivative products such as Collateralized Debt Obligations (CDOs) and other synthetic products. Financial engineers used all sorts of unrealistic assumptions to build these products with one on top of the other, ultimately showing that these complex instruments were essentially risk-free. These products were the results of wide-scale misuse of the Black - Scholes PDE. When you price the risk of a derivative for which the underlying asset is another derivative, you are *defacto* taking too much risk, which is akin to the idea of bullwhip effect [18] studied in supply chains. Volatilities have a tendency to explode in such a multilayer structure. Therefore even the smallest of shocks in the derivatives markets can cause massive volatility swings for a complex derivative built on top. When that happens, the Black - Scholes algorithm is no longer valid, and all the risk pricing previously undertaken has little meaning.

2.5 Chapter Summary

In this chapter, we summarized the main ideas from the Black-Scholes model. We will use these concepts to achieve the following objectives:

1. Understand and embed the idea of risk into our algorithms.

2. Use the idea of Black - Scholes in conjunction with Reinforcement Learning to identify improved risk-taking strategies

3. Provide recommendations from our algorithm in a way that helps a market maker.

CHAPTER 3
REINFORCEMENT LEARNING OVERVIEW

3.1 Foundational Ideas of Reinforcement Learning

3.1.1 Introduction

Reinforcement learning is a kind of machine learning technique in which our goal is to maximize the total rewards for an agent of interest while the agent is interacting with an environment by way of taking actions and receiving and evaluating the feedback from the environment in the form of rewards or penalties resulting from a particular action of the agent. Further, we try to maximize rewards over a sequence of interactions rather than a single interaction. That way reinforcement learning enables us to find global solutions to sequential decision-making problems. It is this ability of reinforcement learning algorithms that we employ to redefine the Black-Scholes delta hedging problem and ultimately show that we can overcome some of the limitations of the Black-Scholes model and, as a result, generate incremental profit for the market maker. The focus of this chapter is to discuss various reinforcement learning algorithms and, in the end, propose a variant of the Monte Carlo reinforcement learning algorithm that we use in our own delta hedging model.

3.1.2 A Brief History of reinforcement learning

Sutton and Barto state in their book [19] that before the 1980s reinforcement learning was studied in the form of two separate fields i.e., learning through trial and error and the more rigorous optimal control. But these ideas came together in the 1980s as computer scientists started to see the connection. To put things into perspective, some of the popular concepts studied in machine learning, such as the Bellman equation [20] and MDPs - Markov Decision Processes [21] have been known long before they were adopted extensively into

reinforcement learning literature. Sutton and Barto attribute the name "reinforcement" to have been inspired by the very famous "Pavlov's experiment on dog" that studied the ability to elicit desired behavior from the dog by setting up an appropriate reward structure. This is also sometimes referred to as the "law of effect." Turing (1948) and Shannon (1952) studied the law of effect independently and published results that show that ideas such as learning a reward structure for reaching a goal had been known. However, the first major paper that discussed reinforcement learning in a way that matched our current interpretation of the field was by Klopf (1972) [22] that brought together the concepts of an environment, goal and rewards, and showed that adaptive behavior could be induced. Both Sutton and Barto claim that it was, in fact, this paper that influenced their decision to pursue reinforcement learning as opposed to supervised learning.

3.1.3 Definitions

The following is a list of important notations widely used in Reinforcement Learning literature.

1. State (s)
2. Action (a)
3. Set of non-terminal states (S)
4. Set of all states including Terminal State (S^+)
5. Set of all possible Rewards (R)
6. Discrete Time Steps (t)
7. Final Time Step (T)
8. Reward generated at t (R_t)
9. Action taken at t (A_t)
10. Returns - cumulatively discounted rewards between t and T (G_t)

11. Policy/Decision Making Rule (π)
12. Action taken in State s following π ($\pi(s)$)
13. Probability of transitioning to state $s^{'}$, receiving reward r from state s by taking action a ($P(s^{'}, r|s, a)$)
14. Expected Value of being in state s under policy π ($v_{\pi}(s)$)
15. Expected Value of being in state s under optimal policy ($v_{*}(s)$)
16. Expected Return by taking action a in state s under policy π ($q_{\pi}(s, a)$)
17. Expected Return by taking action a in state s under optimal policy ($q_{*}(s, a)$)
18. Random variable representing $v_{\pi}(s)$ or $v_{*}(s)$ ($V_t(s)$)
19. Random variable representing $q_{\pi}(s, a)$ or $q_{*}(s)$ ($Q_t(s, a)$)
20. Probability of a Random Action in an $\epsilon - greedy$ policy (ϵ)
21. Discounting Rate (γ)

3.1.4 Concepts in reinforcement learning

The following are some of the key concepts that we often use to construct reinforcement learning algorithms.

1. Agent - The agent(s) in a reinforcement learning problem can be defined in analogous terms as the protagonist of a story. Our entire goal in the learning problem centers around the fact that we need to learn policies and actions (definitions follow) in such a way that we ensure that the agent receives as much reward as possible. Similarly, in a multi-agent problem, we want to learn policies and actions in such a way that all the agents as a group benefit the most by finding the optimal solution (i.e., maximize rewards) to the learning problem. It is also possible to formulate a problem where we only maximize gains for a subset of agents and simultaneously minimize gains for the

complementary subset of agents. By imposing a special structure on the nature of rewards, we can effectively redesign our problem in such a way that we can effectively maximize rewards of the individual agent(s). Given this notion of rewards, it is only apt for us to mention that in multi-agent learning problems, various game-theoretic equilibria would invariably pop up, giving us very useful insights into the structure of the task.

2. Environment - In ideal terms, the environment can be defined as a black box that is capable of reacting to the actions of an agent(s) and be able to provide feedback, either immediate or delayed, back to the agent(s). In reality, the environment itself has its own limitations. It can only understand those actions that are pre-configured into its domain. Similarly, when it comes to rewards, it again is dependent on the distribution of rewards that have been pre-fed to it. In short, the environment in some ways can be considered as a static spectator, which in most real-world situations, is not true. Even if one were to argue that we can make the environment adaptive, even then, the rules of adaptive behavior need to be specified *apriori*, rendering the idea of a self-learning environment moot.

3. Policy - Sutton and Barto [19] define policy as the agent's way of behaving at any given time. Policies might be stochastic or deterministic (in the latter case, we are not learning anything). But the core idea behind the policy is to provide the agent with a set of rules either in a functional form ($action = f(state)$), in tabular form ($state \rightarrow action$) or maybe something else, that the agent can use to decide the best action. It is often the case that we start with random policy, but during the learning process through exploration and exploitation and by observing the subsequent rewards from the environment, we are able to improve the original random policy until we reach a point where a policy can not be improved any further, and we have in our hands the optimal policy.

4. Episodes - Episodes are analogous to the idea of epochs in deep learning. As we are

going to see later, for any given reinforcement learning problem, the learning does not happen at once; rather we have to let the agent go through from initial state to final state multiple times, interacting all the while with the Environment, to be finally able to learn the optimal policy. Each time the agent goes through the sequence of states from initial to terminal, we call it an episode. It is important to note here that not all problems are episodic, sometimes we have problems that continue over time with no final states whatsoever.

5. Rewards - This is by far the most complicated concept in reinforcement learning and. in some ways, considered to be as much an art as science. While the reward system is built into the environment, ultimately, it is we who design the environment itself while trying to solve the problem at hand. It is imperative, therefore, that we, as the designers of the learning problem put enough thought into the design of the reward system. The motivation for the reward function might come from domain knowledge, experience, examples, or even from best practices. Whatever the case may be, the designer needs to thoroughly understand the implications of the reward mechanism design. There seem to be some widely accepted notions about the rewards structure, which is worth considering:

   (a) $Reward \in (-\infty, \infty)$.

   (b) The only way for an agent(s) to learn about the environment is by observing the sequence of rewards that accrue from the sequence of its own actions.

   (c) The higher the reward, the higher the incentive for the agent to reach/pass through that state.

   (d) A negative reward from a state acts as a penalty signal for the agent. It disincentivizes the agent to ever pass through the corresponding states.

   (e) Rewards can be immediate. The environment can send out the reward to the agent(s) immediately following an action. Alternatively, rewards might be delayed. The environment might only choose to send out a reward signal at the

very end of an episode or at intermediate time points in a continuing game. In case of delayed rewards, we often indicate that as $R = 0$ to be used in the various equations.

(f) There is a notion of discounting that is associated with rewards in a learning problem. The idea is that we want to attach greater importance to rewards that are generated in the vicinity of the current time, looking forward, as opposed to rewards that accrue much farther out into the future. This allows the Learning algorithm to be optimal both in a local as well as global sense. We want the policy to be good not just immediately but also good overall for the problem. The magnitude of the discounting parameter γ controls this tradeoff between the local and global significance of rewards.

6. Value Function - As we just discussed, the agent(s) learns by observing the reward. But there is an operational question here, where are all the rewards that an agent(s) obtains from the environment stored? The answer is that an agent(s) stores these rewards in the form of a specially designed function called the value function. Reinforcement learning prescribes special equations to handle rewards and store them through this notion of value functions. In fact, Sutton and Barto [19] note that "the central role of value estimation is arguably the most important thing that has been learned about reinforcement learning over the last six decades".

7. Explore - Exploit tradeoff - This is again a very critical concept in reinforcement learning, especially in non-deterministic environments that are constantly evolving by design or by experience. Controlled by the parameter $\epsilon < 1$, exploration is key to the learning process in a reinforcement learning problem. Building on the concept of the value function, imagine that an agent is placed into the environment with no prior knowledge and that the values of all states for this agent have been initialized to zero. In this case, the agent has no information at all to exploit. The only way for the agent to start learning is to continuously explore the environment, get a sequence of rewards and update the values of the states until such time that it has gathered enough initial

information for exploitation to be meaningful. ϵ controls what percentage of time we are willing to explore. For example, with $\epsilon = 0.2$, we are telling the agent to explore (i.e., take a random action) once every five time-steps. This means that if values of all states are initialized to zero, the agent is learning really slowly at first until enough states start to have values associated with them, and exploitation becomes meaningful. But it is important to note that research [23] has shown that exploring at least ϵ fraction of the time throughout the learning process helps the algorithm converge faster to the steady-state values. In fact constant exploration is a *defacto* necessity in stochastic environments.

3.2 Typical formulation of a reinforcement learning problem

1. Define the key parameters of the problem i.e., states, rewards, action, ϵ, etc.

2. Instantiate both the agent(s) and environment and define the rules of engagement so that both the agent(s) and environment know what to expect in terms of definitions of states, actions, and the nature of rewards from each other.

3. For the agent(s), associate states, actions, and rewards with a value function so that storage becomes possible.

4. Implement a handler, that can play out the episodes from the initial state to end state over discrete time steps and use ϵ to maintain an exploration-exploitation.

5. Start with a policy. It is perfectly fine to start with a completely random one where actions are chosen at random at each time step.

6. Define a notion of tolerance so that you can terminate the learning problem when you notice that convergence has been reached. After a certain set of episodes, the successive updates to the values of the states fall below the tolerance value signaling that it is perhaps time to terminate.

7. Once convergence has been reached, use the value function to improve your random policy to something more informative based on the experience so far. Use this new policy to again go through several episodes and update the value functions.

8. Over multiple iterations, one can observe that the previously learned policy is identical to current policy, indicating that policy convergence has been reached, and it is time to terminate.

3.3 Success stories of reinforcement learning

The real moment of glory and worldwide appreciation for reinforcement learning came with the much-storied success of Google's "Go"-playing "AlphaGo" [24] engine that was able to comprehensively defeat the world champion of "Go" multiple times in a series of games. Reinforcement learning is also being actively tested on robots, [25] and chances are that in due course, most classical control problems will rather be solved as reinforcement learning control problems.

3.4 Important reinforcement learning algorithms

Following are some of the most important varieties of reinforcement learning problems:

3.4.1 Multi-arm Bandits

The k-armed bandit problem is perhaps the best known and the earliest studied problem in reinforcement learning. The setting of the problem is very similar to playing a slot machine at a Casino with multiple levers. As you pull different levers in the slot machine, you receive different rewards depending on what levers correspond to the best actions that maximize your gains. We can formulate the problem using the notation that we have stated previously as:

$$q_*(a) = E[R_t|A_t = a]. \tag{3.1}$$

Here, $q_*(a)$ indicates the value of choosing an action a. But unfortunately, the true value of $q_*(a)$ is unknown apriori. All we can have at best is perhaps just an estimate $Q_t(a)$. Over the long run, i.e., after playing the slot machine long enough, we want $Q_t(a)$ to converge to $q_*(a)$. To study the convergence, we need to define $Q_t(a)$ clearly,

$$Q_t(a) = \frac{\sum_{i=1}^{t-1} R_i \cdot 1_{A_i=a}}{\sum_{i=1}^{t-1} 1_{A_i=a}}.$$

The following conclusions can be made from the above equation:

1. $Q_t(a)$ is the average of all rewards generated every time the lever a is pulled.
2. As $\sum_{i=1}^{t-1} 1_{A_i=a}$ increases when you have taken an action a a large number of times, the ratio stabilizes.
3. Once the ratio becomes stable, we can claim that $Q_t(a)$ has converged and is an unbiased estimate of $q_*(a)$.
4. Convergence is guaranteed for stationary problems where the distribution of rewards does not change over time.

The k-armed bandit problem also gives us the opportunity to understand the explore-exploit tradeoff better. Let's say we have played the k-armed bandit machine a number of times p. A typical life cycle of moves that would arise in those p attempts is as follows:

1. At the very beginning, we have no idea what each of the k levers is worth. So let's say we set at time zero, for all k levers.

$$Q_0(a = i) = 0 \quad \forall i \in [1, k].$$

2. Since we have no information, there is nothing to exploit. We have to learn purely by exploring.

3. For some attempt $j < p$, we realize that we have reached a point where exploitation becomes possible and we can start leveraging the information that we already have in terms of estimates $Q_t(a)$

4. From this point on, we have two choices

   (a) We follow a greedy policy, i.e., we assume that the available estimates $Q_t(a)$ have converged to true values and thereafter we just choose one simple policy

   $$A_t = \underset{a}{\text{argmax}} \; Q_t(a) \quad \forall t > p.$$

   This translates to pulling the same lever that has given the highest average reward till time p for all future $t > p$. Note it is not a desirable choice.

   (b) We follow the $\epsilon - greedy$ policy where we decide on the value of $\epsilon < 1$ and use the following rule.

   $$A_t = \begin{cases} \underset{a}{\text{argmax}}\, Q_t(a), & P_{(1-\epsilon)} \\ Random\ a, & P_\epsilon \end{cases}$$

   Two conclusions are immediately obvious, first that it is always desirable to maintain at least some level of exploration throughout the learning process. This becomes especially important in cases where we have an environment without stationarity. What this means is that even if we have observed the environment long enough, we still admit the possibility of the environment changing its behavior. Without room for exploration, we would learn a policy that would probably not generalize well into future times. Second, we need to choose an optimum value of ϵ that stimulates just the right amount of exploration. A large ϵ might be desirable in some cases, especially in highly unpredictable environments, but usually, $\epsilon \leq 0.2$ does the job on most occasions.

We now formally state the algorithm for k-armed bandits as follows:

Initialize, for $a = 1$ to k

$Q(a) \leftarrow 0$, $N(a) \leftarrow 0$

while $forever$ **do**

$$A = \begin{cases} \underset{a}{\text{argmax}} Q_t(a), & P_{(1-\epsilon)} \\ 0, & P_\epsilon \end{cases}$$

R $\leftarrow$ bandit(A)

$N(A) \leftarrow N(A) + 1$

$Q(A) \leftarrow Q(A) + \frac{R - Q(A)}{N(A)}$

end

Algorithm 1: k-armed Bandit Algorithm

3.4.2 Markov decision processes(MDP)

In k-armed bandits, we dealt with problems where the rewards depend only on the action taken. But in the case of MDPs, we add another nuance to the picture, which is the idea of state. As we analyze the idea of "state", we realize that even the k-armed bandit acts as an agent with just one state. It is because of this reason that we conveniently eliminated the state term s in our calculations. A state has the following characteristics:

1. A state s reflects the granularity at which we want to take our decisions in a sequential learning problem.

2. A state can be either atomic or a vector. States can be either represented as numbers or strings, or they can also be represented as vectors of numbers and strings. This thesis defines states as a vector of strings.

3. An agent can only be in one state at any given point in time.

4. In a practical problem, each defined state needs to have an associated numerical value for learning to happen.

5. An episodic learning problem (task) has to have an initial and final state. Continuing tasks may not have a terminal state.

6. The act of an agent moving from one state to another is collectively characterized by a transition probability matrix and is in every aspect similar to identically named matrices widely employed in the study of stochastic processes.

7. The definition of states is decided *apriori*, i.e., while the agent is learning a policy, new states can't be added or existing ones deleted.

8. A state may be visited multiple times during an episode or a continuing task.

The introduction of the idea of states necessitates redefining the structure of value and policy Functions (Action-Value Functions) for MDPs.

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t|S_t = s] \\ &= E_\pi[\sum_{k=t+1}^{T} \gamma^k R_k|S_t = s] \quad \forall s \in S \end{aligned} \tag{3.2}$$

Here, R_k represents the rewards obtained at time step k, T indicates that the episode is T time steps long, π indicates the policy being followed throughout the episode, γ is the discounting parameter, $S_t = s$ represents the fact that the agent is at state s at time step t and finally G_t represents returns (discounted sum of rewards from time step t to end of the episode at time T).

Following are the key set of ideas conveyed by (3.2):

1. The calculations are being done assuming a constant policy. Just to recall, we usually start with a random policy. Then as we run through the task multiple times, we are able to make improvements to an existing policy, thereby creating a new policy. As we update the policy, the calculation for the value function will also update.

2. We decide the value of the discounting factor $0 \leq \gamma \leq 1$ *apriori*. This is, however, just a tuning parameter. We can do a grid search over various values of this tuning parameter and find the one that maximizes an objective function, such as the speed of convergence.

3. For a given episode of length T, at a particular state s at time t, we look at all future discounted rewards that accrue immediately following time t up to the time T and compute an average to estimate the value function $v_\pi(s)$.

The next important concept that we need to discuss is policy functions. Policy functions add another layer of granularity to the value functions. They go a step deeper than value Functions by adding the action dimension into the mix. In reality, we are almost always interested in policy functions because they are directly usable in solving most learning problems. The formulation of the policy function is as follows:

$$\begin{aligned} q_\pi(s,a) &= E_\pi[G_t|S_t = s, A_t = a] \\ &= E_\pi[\sum_{k=t+1}^{T} \gamma^k R_k|S_t = s, A_t = a], \quad \forall s \in S, \;\; a \in A. \end{aligned} \tag{3.3}$$

It is quite evident that the $q_\pi(s,a)$ function has all the properties that we defined for the $v_\pi(s)$ function before. The only difference is that we are calculating and updating the value of the function at a more granular level.

The next important topic of discussion in MDPs is the **Bellman Equation**. Originally proposed by Richard Bellman, this equation [20] is at the heart of dynamic programming

and hence widely employed in solving MDPs.

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t|S_t = s] \\ &= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s^{'}, r|s, a)[r + \gamma E_\pi[G_{t+1}|S_{t+1} = s^{'}]] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s^{'}, r|s, a)[r + \gamma v_\pi(s^{'}|s)], \quad \forall s \in S \\ &= E_\pi[r + \gamma v_\pi(s^{'}|s)], \quad \forall s \in S. \end{aligned} \tag{3.4}$$

The Bellman equation relates the value of one state with the subsequent state in an episode or continuing task. We refer again to the property of MDPs that calculations are performed in the reverse direction by starting with the terminal rewards and working our way backward through time steps to calculate the value of the states (in terms of discounted rewards) as they are encountered in an episode.

An analog of the Bellman equation for policy functions is as follows:

$$q_\pi(s, a) = E_\pi[r + \gamma v_\pi(s^{'}|s, a)], \quad \forall s \in S, \quad a \in A.$$

We are now going to use the Bellman equation to define some additional properties for optimal value functions and optimal policy functions in MDPs. An optimal policy π^* is defined as a policy that is better than or equal to all other policies $\pi \neq \pi^*$. In turn, a policy $\pi^{'}$ is considered better than policy π, if

$$\forall s \in S, \quad v_{\pi'}(s) \geq v_\pi(s).$$

Thus, we can formally define an optimal value as the one that corresponds to the optimal policy and satisfies the following equation

$$v_*(s) = \max_{\pi} v_\pi(s), \forall s \in S.$$

It is important to note that the optimal value may not necessarily be unique. Multiple policies may be optimal at the same time.

In a similar vein, we can define the optimal policy function and express it as

$$q_*(s,a) = \max_{\pi} q_\pi(s,a) \quad \forall s \in S, \;\; a \in A.$$

Now that we have defined what the optimal policy and optimal value functions are, we are ready to express them together in the following way:

$$q_*(s,a) = E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a].$$

We have thus established many important properties of dynamic programming in this section.

3.4.3 Monte Carlo reinforcement learning

Monte Carlo methods are one of the most efficient methods to solve reinforcement learning problems, especially in non-stationary environments [26]. For ease of understanding, let's consider **episodic** reinforcement learning problems. The Monte Carlo methods can be described in a series of following steps for such problems:

1. Simulate a large number of episodes for a given policy.

2. Observe the final returns at the end of the episode.

3. For each episode, use the following discounting scheme

$$G \leftarrow \gamma G + R_{t+1}.$$

to work the returns back through time and calculate values for each state S_t. Here $0 \leq \gamma \leq 1$ is our usual discount parameter.

4. Over multiple episodes, average the values calculated over each episode for each state.

5. The average is an unbiased estimate of the value function and the standard deviation falls in $O(\frac{1}{\sqrt{n}})$.

There are some other nice properties to Monte Carlo methods. One of the essential problems with using MDPs is that in real life, you do not know the transition probability matrix *apriori*. This means that the Bellman equation can not be applied readily to a problem unless we are in a position to have a prior belief about the transition probabilities. Fortunately, Monte Carlo iterations do not have any probability terms. This offers a major advantage when it comes to applying reinforcement learning to a general class of problems with little prior expert knowledge. Monte Carlo methods also come with theoretical guarantees that our estimates will converge if we have played a large number of episodes of the task repeatedly under a given policy. Thus Monte Carlo methods offer a very simple yet very robust way of solving reinforcement learning problems.

We need to introduce two more concepts [19] that are critical to the understanding of Monte Carlo Learning problems.

1. First visit vs. every visit Monte Carlo - The idea here relates to how we want to compute the Monte Carlo average. In first visit Monte Carlo, we are essentially recording the return calculations once per episode for every state when the state is encountered for the first time. On the other hand, for every visit Monte Carlo, we record returns every time a state is encountered within an episode.

2. On policy vs. Off policy learning - Recall that in reinforcement learning, we often start with a random policy and improve it over the course of many trials. On Policy methods refer to the fact that we use our previous experiences to improve our policy. Off policy methods, on the other hand, compare a policy to a target policy that was generated from an entirely different dataset through the use of importance sampling.

In this thesis, we have employed the on policy Monte Carlo method which is presented in Algorithm 2:

Initialize, $Q(s,a) \leftarrow 0$, $G(s,a) \leftarrow 0$, $RandomPolicy\ \ \pi(s)$

while $forever$ **do**

Randomly Choose Initial State and Action S_0, A_0

Using $\pi(s)$, generate an episode of the form $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$

G $\leftarrow 0$

for *Every step in Episode backwards* **do**

G $\leftarrow \gamma G + R_{t+1}$

Append G *to returns* G(s,a)

Q(S_t, A_t) $\leftarrow average(G(s_t, a_t))$

$\pi(S_t) \leftarrow \underset{a}{\text{argmax}} Q(S_t, A_t)$

end

end

Algorithm 2: On policy Monte Carlo algorithm

3.4.4 Implementation of the Monte Carlo variant algorithm

In this thesis, we have implemented our own faster variant of the Monte Carlo learning algorithm. The variant has the following key characteristics:

1. Runs through the complete episode picking a random action at each time step. This creates a lot of synthetic episodes.

2. Calculates the reward for each episode using the reward function.

3. Picks the top 10% of episodes and stores the complete action history of the top sampled episodes.

4. Looking at the data longitudinally, the algorithm decides reward from which actions chosen at a particular time step dominates other actions chosen at the same time step for the top episodes $R(a, t)$.

5. Directly creates the best policy by selecting the best actions identified at each time step.

6. On test data sets, it is imperative to maintain at least some level of exploration to get the best results.

7. This variant works much faster than the traditional implementation of the Monte Carlo method. This is so because of two improvements; first, we are only considering those simulated episodes that produce profits that are in the top 10% of all profits generated from all episodes and second, it simplifies the calculation of returns of taking action at a given state at any point in the episode.

8. The variant seems to be robust to noise and empirically effective with even just 2000 episodes.

In algorithmic form, the variant can be expressed as follows:

Initialize,

Run several thousand episodes by choosing random actions at each time step

Choose the top 10% episodes with highest rewards

for *Every step in Time Steps* **do**

Observe $R(a, t)$ for each action across all episodes

$\pi(s_t, a) \leftarrow \underset{a}{\text{argmax}}\ R(a, t)$

end

Algorithm 3: On Policy Monte Carlo variant

3.5 Chapter Summary

In this chapter, we discussed the following topics:

1. Learnt the fundamentals of reinforcement learning, including various notations and definitions.

2. Explored the different standard variants of reinforcement learning algorithms.

3. Presented our own variant of the Monte Carlo algorithm to predict and evaluate delta hedging strategies.

CHAPTER 4
DELTA HEDGING WITH REINFORCEMENT LEARNING

4.1 Key Ideas

In this section, we use the key ideas discussed in the previous chapters to design an algorithm that will suggest a strategy to the market maker to delta hedge the option. More specifically, we want the algorithm to have the following characteristics:

1. Appropriately scale stock prices - This is a particularly challenging problem, given that the numerical scale of stock prices changes over time. This means that when we fit a model on an absolute scale, the model may not generalize well when the stock prices change. In the upcoming section we propose a normalization scheme that defines a new scale for stock prices such that a model learned during a certain time period remains applicable for other time periods.

2. Constructing a dynamic reward function - We have discussed previously that one of the key limitations of most existing models is that they do not include transactional costs in the delta hedging model. We propose a dynamic reward function that includes transaction costs.

3. Use Reinforcement Learning algorithms to combine multiple strategies over the option period. The idea is, in many ways, analogous to the core idea behind Random Forests [27] - a meta-algorithm by aggregating the power of base learners. We are leveraging a similar concept here e.g., If a market maker has multiple delta hedging strategies, A and B, we demonstrate how A and B can be combined to create a unified strategy C that performs better than A and B individually. In some ways, we show that our algorithm results in Pareto improvement, which is a net gain in economic welfare

(profits for market maker) as a result of an action with no one (either market maker or buyers) being made worse off.

4.1.1 The idea of state and state as a hyperparameter

The idea of "state" is fundamental to apply reinforcement learning to any problem. Care needs to be taken to define states in a system because we are trying to encapsulate all defining parameters of a dynamic system at the right level of granularity. E.g., if states are to be represented by an ordered tuple $(a_0, a_1, a_2, ..., a_k)$, what is the correct k? If k is too small, we end up with a definition that may not be entirely useful in the context of the problem because a certain state parameter a_i is missing. On the other hand, a state definition with too many parameters will create a lot of overhead both in terms of computation and also memory efficiency.

Our definition of the state

We define our state as a ordered triplet given by (a_1, a_2, a_3). Here a_1, a_2, a_3 are defined as follows:

a_1 - Discretizes the moneyness for current time period (ratio of current stock price to strike price) $\frac{S_t}{K}$ into three different levels H, M, L (High, Medium, Low). Accordingly, we define two thresholds J_1, J_2 that map these ratios to the discrete states using the following rule

$$\begin{cases} H & \forall \frac{S_t}{K} \geq J_2 \\ M & \forall J_1 \leq \frac{S_t}{K} < J_2 \\ L & \forall \frac{S_t}{K} < J_1. \end{cases}$$

a_2 - Similaryly, discretizes moneyness for previous time period $\frac{S_{t-1}}{K}$ into three different levels H, M, L (High, Medium, Low) using the same rules as was the case for a_1.

a_3 - This state parameter encodes how long the system has been in a given state in

discrete terms H, M, L (High, Medium, Low) with levels H, M, L (High, Medium, Low) given by threshold parameters M_1, M_2.

Note - The reason we define our state in this manner is to capture some of the insights from the Clewlow and Hodges model (parameters a_1, a_2) and the Heston and Nandi model (parameter a_3):

1. In line with the central idea of Clewlow and Hodges, we define bands (H, M, L). Only when stock prices move outside these bands, we reach a decision point where action needs to be taken.

2. Our third parameter represents an indirect way to capture volatility. If states tend to change slowly, we are implying that the underlying volatility is low and vice versa.

Therefore, we have a total of $(H, M, L) \times (H, M, L) \times (H, M, L) = 27$ possible system states.

4.1.2 The Nature of the reward function

Continuing with our goal to include market realities such as transaction, we define our reward function with the following components:

1. Transaction cost component (X_C) - Assuming that the market maker borrows money at a risk-free rate from the capital markets to hold delta hedged positions against an option, we can define:

$$X_C^t = \Delta_t \cdot S_t \cdot r.$$

Over the entire option period, our total transaction cost is given by

$$X_C = \sum_t X_C^t = \sum_t \Delta_t \cdot S_t \cdot r.$$

2. Trading cost component (X_T) - This component embodies the well known-idea that a trader should never sell low and buy high. The Black - Scholes Delta hedging approach

forces the market maker to adjust its position on the basis of the daily calculated delta. But in this process, many occasions arise where the market maker is buying high and selling low. The foundational idea of our algorithm is to address this issue by introducing a delay in reaction time to a market movement by scoring historical price movements with the reinforcement learning Algorithm. While introducing this delay does wean the market maker away from risk neutrality, yet we control it by introducing bands. That is, even if prices have moved if they are still inside the bands, it means that from a practical perspective, our risk profile hasn't changed drastically. We define X_T as:

$$X_T = \sum_t (\Delta_t - \Delta_{t-1}) \cdot S_t.$$

3. Total reward function is thus given by:

$$X = X_T - X_C.$$

Our aim is to maximize this reward function over the entire option time period using our Monte Carlo variant algorithm.

4.1.3 Define actions and the complete policy table

To show how our algorithm works, we will consider the simplest possible situation in this thesis. Only two possible actions will be chosen by the reinforcement learning algorithm at each time point:

1. Black Scholes delta - In this scenario, we simply take the action of adjusting to the delta position as suggested by the Black-Scholes formula, indicated by $A = 1$. Thus a corresponding entry in a policy table will look like $(a_1, a_2, a_3) = 1$. Here $a_1, a_2, a_3 \in H, M, L$ as defined earlier.

2. Do nothing - In this alternative scenario, we choose to do nothing at time t, i.e., hold on to the delta position that we had in the previous time period $t - 1$. It can be

observed that all we are doing is setting a few of the terms in $X_T = \sum_t (\delta_t - \delta_{t-1}) \cdot S_t$ to zero. This vastly reduces the trading losses. A different way of looking at it is to argue that even if there is high volatility in the stock price, we choose not to react to it. Thus the volatility that we experience is less than the volatility of the stock. Thus a corresponding entry in a policy table will look like $(a_1, a_2, a_3) = 0$.

4.2 Combine multiple strategies.

As we observed in the previous section, we can choose between two possible actions. The problem that we are really solving is when to choose one of these actions as we navigate along the option time period at each time t. Also, for this combined strategy to make sense, we have to show that the combination results in a strategy that has a higher reward when compared to the individual strategies available.

4.2.1 Monte Carlo variant formulation

The details of our Monte Carlo variant implementation in python are as follows:

1. For a given stock (for which we are selling an option), we choose a fixed option period.
2. Read data (daily underlying stock prices for an option) and split it into several chunks using the option period above. Consider one chunk at a time, say C_k.
3. Define threshold $J_1 = 0.5, J_2 = 1.0$ for moneyness $\frac{S_t}{K}, \frac{S_{t-1}}{K}$ as well as $M_1 = 3, M_2 = 7$ for time (days).
4. Run a forward pass through C_k and construct states as a ordered tuple of the form (a_1, a_2, a_3) using the function ***state_discretizer()*** that we have implemented.
5. Define a set of candidate actions for the algorithm to choose from. In our model, we propose to choose between just two actions at any discrete time-point i.e., between Black-Scholes ($a = 1$) and do nothing ($a = 0$).
6. Using the steps defined in Chapter - 3, we define a method that implements the Monte Carlo variant.

7. Define the reward function denoted by X in the previous section. We will try to maximize X.

8. Try $n = 2000$ Monte Carlo episodes. By the end of all episodes, our algorithm populates the policy table.

9. Once the policy is populated, we run an $\epsilon - greedy$ policy on unseen test data and monitor the performance of our algorithm.

4.3 Chapter Summary

In this chapter, we explored in detail:

1. Define the state space for the delta hedging problem.

2. Define actions in this state space.

3. Explore in detail the components of the reward structure.

4. Steps in implementation of the Monte Carlo variant in python.

CHAPTER 5
RESULTS AND DISCUSSION

5.1 Discussion of the various scenarios

One of the key goals of the proposed algorithm is generalizability to a wide variety of scenarios. There are two key characteristics to consider:

1. Inherent volatility of the underlying asset. For example, compared to the S&P 500 ETF, the ETF QQQ is more volatile, and the ETF BND is less volatile. We want to see if our algorithm generalizes well to each of these scenarios.

2. Different economic situations also impact volatility. For example, the world experienced a significant amount of volatility and market drawdown during the one month period between Feb 19, 2020, and Mar 21, 2020, due to the COVID-19 outbreak. One of our goals is to see how well does the proposed delta hedging Algorithm work during periods of high economic uncertainty.

5.2 Approach to testing the algorithm in different scenarios

1. Taking training data over multiple time periods, we learn a policy using the Monte Carlo algorithm.

2. We prepare various testing datasets that mimic different volatility conditions.

3. Run the various testing datasets through the learned policy in (1) with some exploration rate ϵ and calculate the value of the function that indicates the reward generated by taking the actions recommended by the policy. We are using the same definition of reward, denoted by X, which was defined in Chapter - 3.

4. Repeat (3) for pure Black-Scholes policy.

5. Compare the performance of both strategies defined in (3) and (4) above

5.3 A note on interpreting the policy plot and rewards table

To demonstrate our results, we are using the following plot and table:

1. Reward Table - These tables show rewards resulting from following a particular policy. Our algorithm is denoted as "Monte Carlo Variant", the Black-Scholes method is denoted as "Black-Scholes" and finally, the Longstaff - Schwartz method is denoted as "Longstaff - Schwartz".

2. Policy Plot - This plot superimposes the actions chosen by the Reinforcement Learning algorithm on top of the stock prices. Chosen Action (1) indicates that the algorithm chose to use Black - Scholes delta at a certain time point ,and Chosen Action (0) indicates that the algorithm did not change the delta for that time point i.e., retained the delta.

5.4 A note on volatility

In this chapter, we use the term volatility in two different contexts. The contexts and definitions are as follows:

1. Volatility periods - This notion refers to the more classical definition of volatility i.e., we calculate the standard deviation of daily stock prices through a quarterly time frame for our algorithms. While our Monte Carlo variant does not directly require a volatility estimate, we feed in the volatility of the previous 90 day time period as an input parameter to the Black-Scholes algorithm.

2. Volatility with respect to the market - This notion of volatility is used to compare the volatility of two different stocks, each compared against the same baseline. The baseline is usually the S&P 500 ETF (market) (VOO). A measure called "beta" is used to compare volatilities. Higher the beta(β), the higher the beta, the more volatile is a stock w.r.t the market. In the following sections, we take two ETFs and compare

them with the market ETF i.e., VOO. ETF QQQ ($\beta > 1$) and ETF BND ($\beta < 1$). We do so to show how our algorithm performs with both ETFs that are either more volatile or less volatile when compared to the reference market ETF i.e., VOO.

5.5 Testing the model on a broad market ETF. (VOO)

VOO is one of the most popular ETFs and is considered to be a market benchmark. It tracks the S&P 500 Index.

The reason behind testing our algorithm to the market ETF is to show that our algorithm works in the most general situation. The market ETFs are among the most highly traded ETFs and quite naturally a favorite for option buyers. By showing that our algorithm performs well for the market ETF, we are able to make a strong case for the adoption of our algorithm.

Our goal is also to test our approach across various known market conditions. We choose out of sample periods that correspond to periods of normal volatility, low volatility ,and high volatility (such as during early months of COVID-19). The following table shows the various training and testing time periods used in our model based on actual prevalent stock prices.

Volatility	**Train Period**	**Test Period**
Normal	7.4.2017 - 10.9.2017	2018.01.10 - 2018.04.10
High	7.4.2017 - 10.9.2017	2019.12.31 - 2020.03.30
Low	7.4.2017 - 10.9.2017	2019.01.07 - 2019.04.04

Table 5.1: [VOO] Specification of model train and test time frames.

The results are as follows:

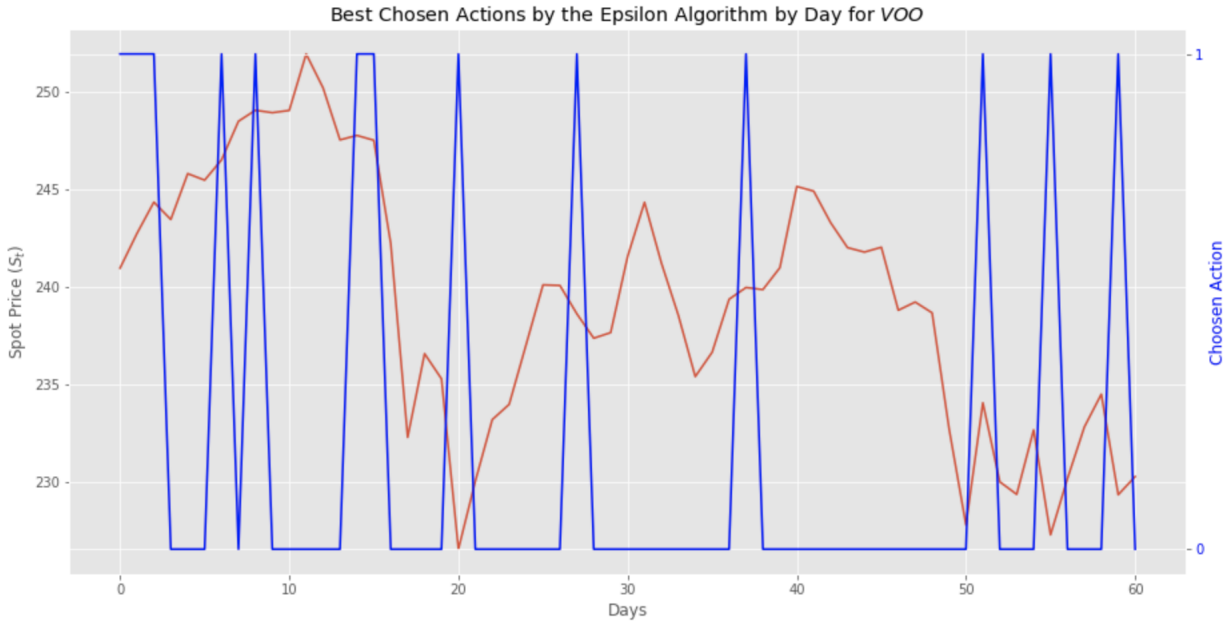

Fig. 5.1: Policy Plot : Normal Volatility [VOO] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	-21.19
Black - Scholes	-23.33
Longstaff - Schwartz	5.60

Table 5.2: Reward Table : Normal Volatility [VOO] - Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

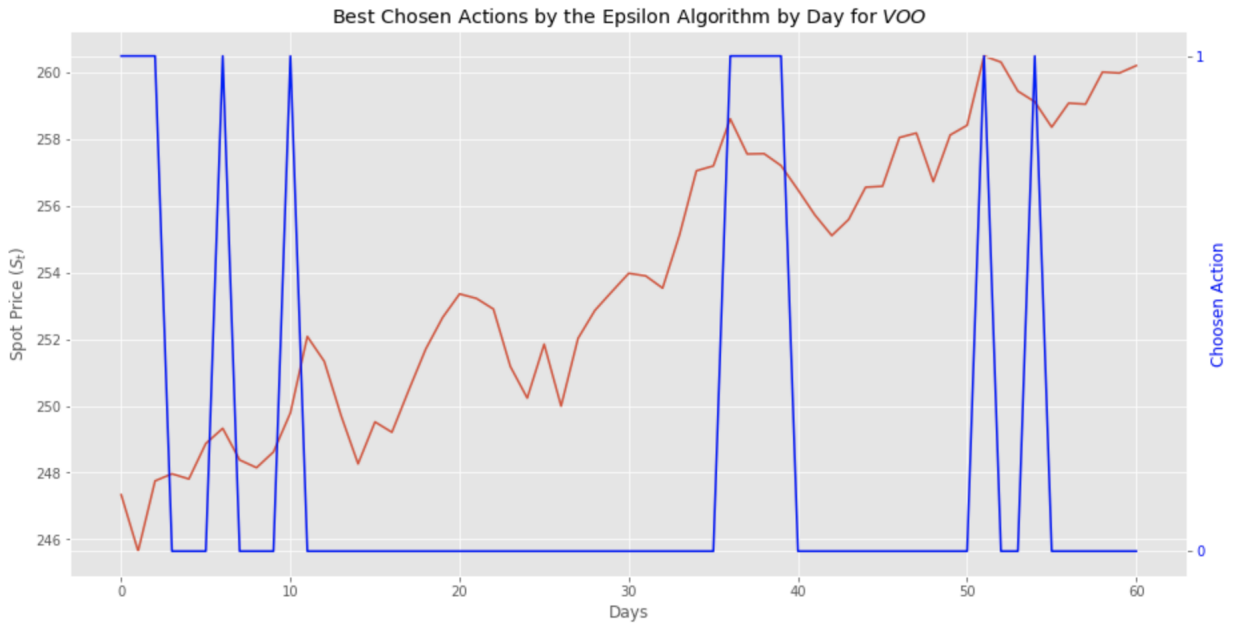

Fig. 5.2: Policy Plot : Low Volatility [VOO] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	162.68
Black - Scholes	180.75
Longstaff - Schwartz	180.64

Table 5.3: Reward Table : Low Volatility [VOO] Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

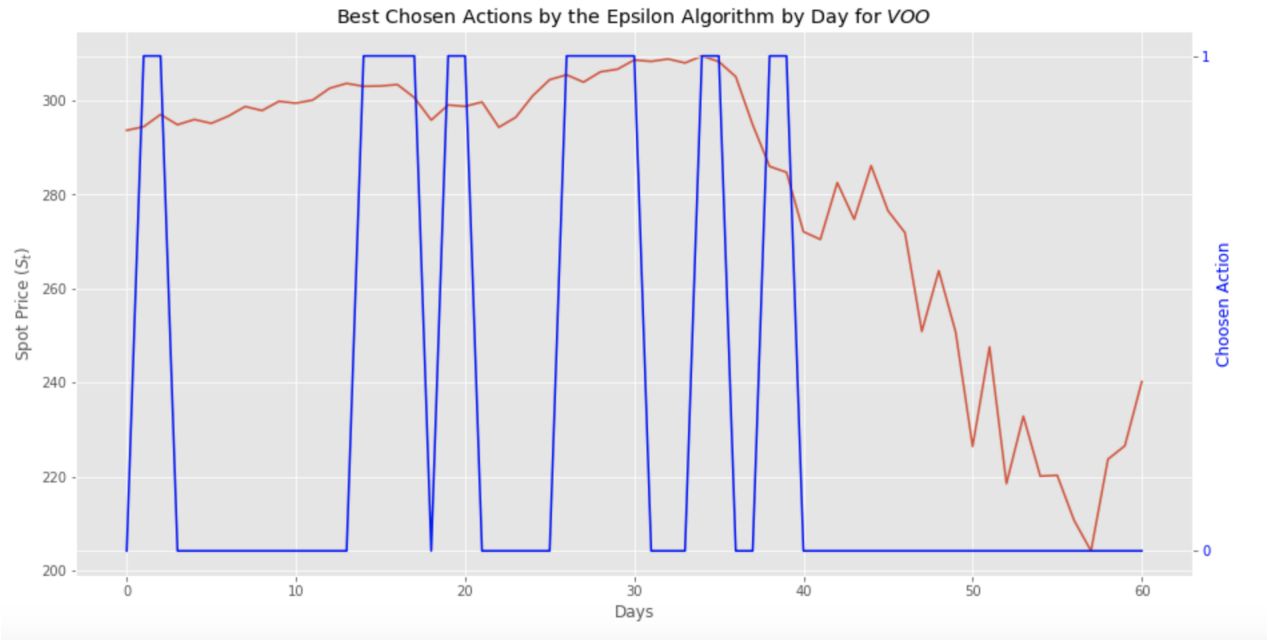

Fig. 5.3: Policy Plot : High Volatility [VOO] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	13.02
Black - Scholes	-80.38
Longstaff - Schwartz	-80.04

Table 5.4: High Volatility [VOO] - Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

5.5.1 Observations

We observe the following:

1. Figures 5.1, 5.2, 5.3 represent the policy plots that show the actions taken by our algorithm during a 90 day option time frame.

2. Tables 5.2, 5.3, 5.4 represent the rewards generated by following recommendations of all three policies i.e., Monte Carlo variant - our algorithm, the Black-Scholes formulation ,and the Longstaff-Schwartz algorithm.

3. Table 5.4 shows that the Monte Carlo variant is performing remarkably better than the Black-Scholes and the Longstaff-Schwartz algorithms in high volatility situations. It is important to note that we have used the COVID-19 crash time frame to demonstrate the usefulness of our algorithm.

4. Our algorithm is performing significantly better in normal volatility situations ,but the Longstaff-Schwartz algorithm outperforms ours in this case.

5. The policy plot in figure 5.2 (high volatility time frame) shows that our ability to outperform both the Black-Scholes and the Longstaff-Schwartz models stems from the fact that our algorithm doesn't react suddenly to market movements thereby cutting out a lot of experienced volatility ultimately reducing trading losses.

6. The policy plot in figure 5.1 (low volatility time frame) shows that using our algorithm actually proves to be counterproductive. This is perhaps because the cost of holding a position far outweighs the advantages of reducing trading losses.

7. The Longstaff-Schwartz model seems to be performing about the same as the Black-Scholes model in most cases, except for the normal volatility situation where it significantly outperforms both our model as well as the Black-Scholes model.

5.6 Testing the model on a high volatility technology ETF (QQQ)

QQQ tracks the NASDAQ 100 Index. QQQ has been historically more volatile than the broad market ETF S&P 500. QQQ is composed mainly of large-cap technology companies and is representative of the health of the technology sector.

The reason behind testing our algorithm to the QQQ ETF is to show that our algorithm works for ETFs with $\beta > 1$. By showing that our algorithm performs well for the QQQ ETF, we are able to make a strong case algorithm generalizes just as well for ETFs that are inherently more volatile than market ETFs ,such as VOO.

Our goal is also to test our approach across various known market conditions. We choose out of sample time periods that correspond to periods of normal volatility, low volatility ,and high volatility (such as during early months of COVID-19). The following table shows the various training and testing time periods used in our model based on actual prevalent stock prices.

Volatility	**Train Period**	**Test Period**
Normal	7.4.2017 - 10.9.2017	2019.07.05 - 2019.10.01
High	7.4.2017 - 10.9.2017	2018.01.10 - 2018.04.10
Low	7.4.2017 - 10.9.2017	2019.10.02 - 2019.12.30

Table 5.5: [QQQ] Specification of model train and test time frames.

The results are as follows:

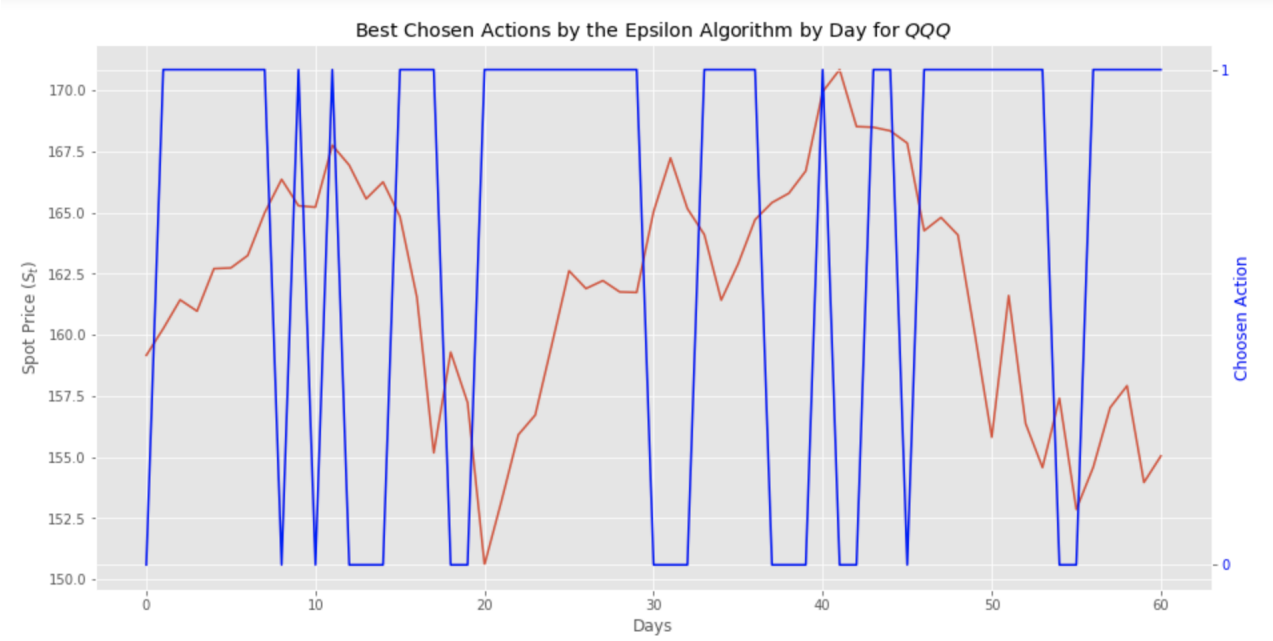

Fig. 5.4: Policy Plot : High Volatility [QQQ] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	7.66
Black - Scholes	-34.50
Longstaff - Schwartz	-32.96

Table 5.6: Reward Table : High Volatility [QQQ] - Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

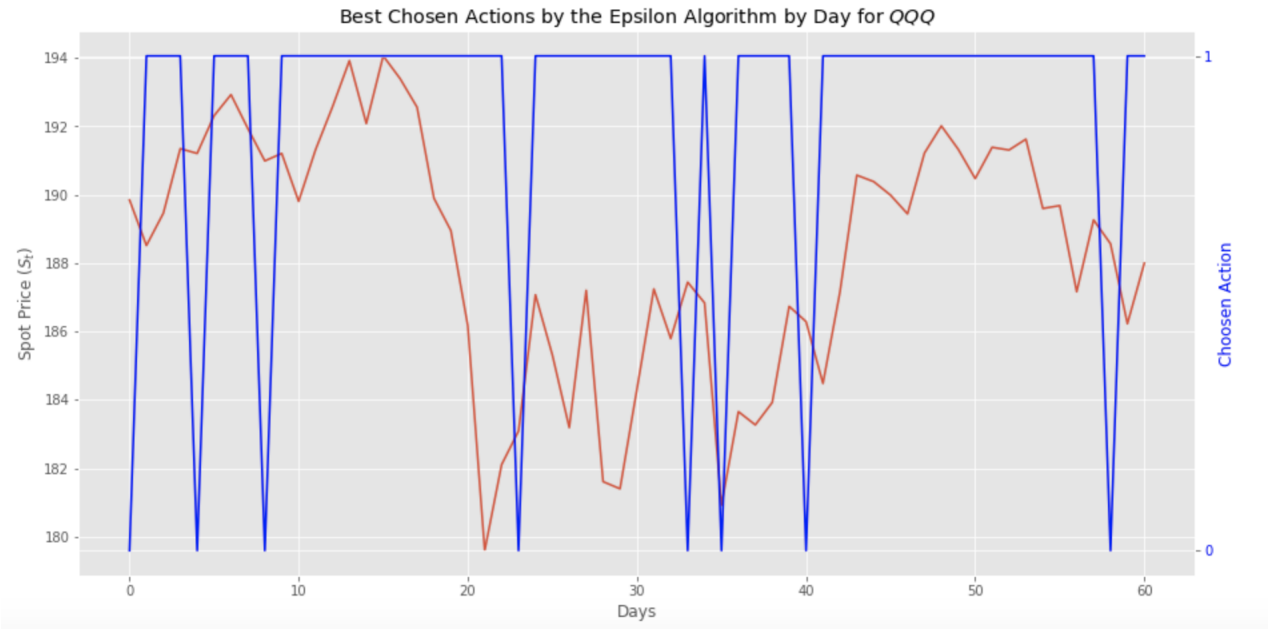

Fig. 5.5: Policy Plot : Normal Volatility [QQQ] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	3.58
Black - Scholes	-65.96
Longstaff - Schwartz	-65.13

Table 5.7: Reward Table : Normal Volatility [QQQ] - Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

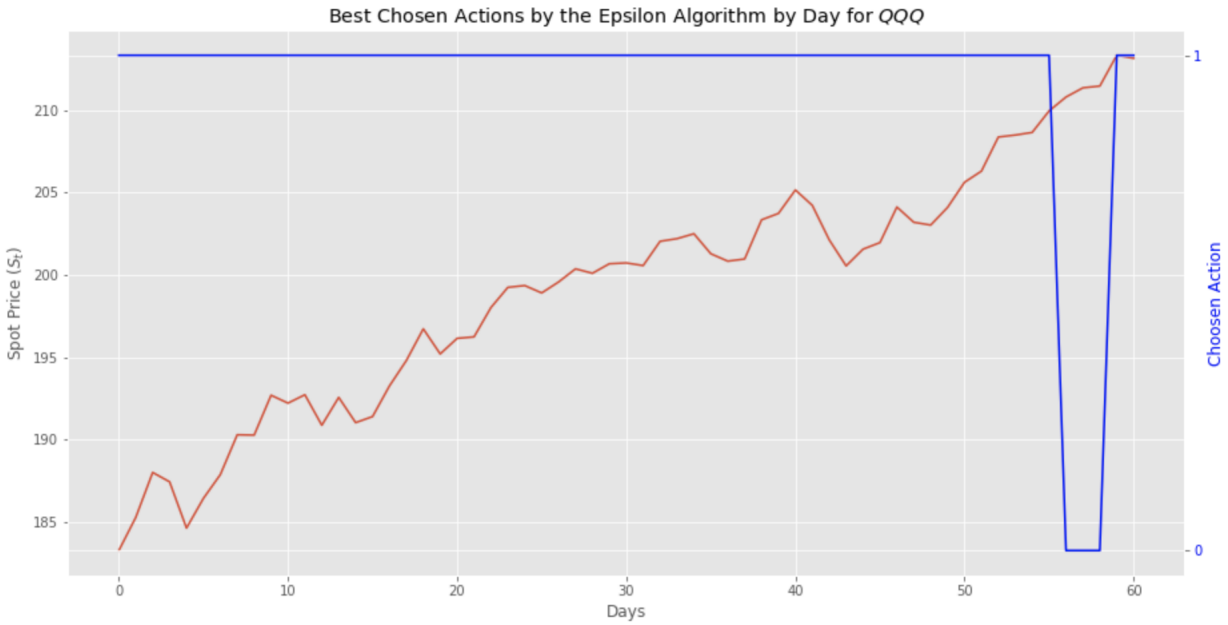

Fig. 5.6: Policy Plot : Low Volatility [QQQ] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	147.05
Black - Scholes	146.99
Longstaff - Schwartz	14.83

Table 5.8: Reward Table : Low Volatility [QQQ] Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model for ETF QQQ.

5.6.1 Observations

We observe the following:

1. Figures 5.4, 5.5, 5.6 represent the policy plots that show the actions taken by our algorithm during a 90 day option time frame.

2. Tables 5.6, 5.7, 5.8 represent the rewards generated by following recommendations of all three policies i.e., Monte Carlo variant - our algorithm, the Black-Scholes formulation and the Longstaff-Schwartz algorithm.

3. Table 5.6 shows that the Monte Carlo variant is performing far better than both the Black-Scholes ,and Longstaff-Schwartz algorithms in all three volatility situations.

4. Just as was the case with the Market ETF, figures 5.4 and 5.5 suggest that the ability of our algorithm to delay its response to volatility results in avoidance of large trading losses. This feature of our algorithm seems to be particularly useful for a high volatility ETF ,such as QQQ.

5. Particularly interesting is the case where constantly increasing stock price of QQQ, we notice from figure 5.6 that our algorithm is msking constant adjustments as per Black - Scholes ,which is the desired behavior.

6. For a high volatility ETF such as QQQ, the Longstaff - Schwartz algorithm seems to be performing about the same or worse than the Black-Scholes algorithm.

5.7 Testing the model on a low volatility Bond ETF (BND)

BND tracks the Bloomberg Barclays U.S. Aggregate Float Adjusted Index. BND has been historically less volatile than the S&P 500. BND consists of investment-grade, taxable, fixed-income securities in the United States-including government, corporate, and international dollar-denominated bonds, as well as mortgage-backed and asset-backed securities-all with maturities of more than one year.

The reason behind testing our algorithm to the BND ETF is to show that our algorithm works for ETFs with $\beta < 1$. By showing that our algorithm performs well for the BND ETF, we are able to make a strong case that our algorithm generalizes just as well for ETFs that are inherently less volatile than market ETFs such as VOO.

Our goal is also to test our approach across various known market conditions. We choose out of sample time periods that correspond to periods of normal volatility, low volatility ,and high volatility (such as during early months of COVID-19). The following table shows the various training and testing time periods used in our model based on actual prevalent stock prices.

Volatility	**Train Period**	**Test Period**
Normal	7.4.2017 - 10.9.2017	2018.01.10 - 2018.04.10
High	7.4.2017 - 10.9.2017	2019.12.31 - 2020.03.30
Low	7.4.2017 - 10.9.2017	2018.07.10 - 2018.10.04

Table 5.9: [BND] Specification of model train and test time frames.

The results are as follows:

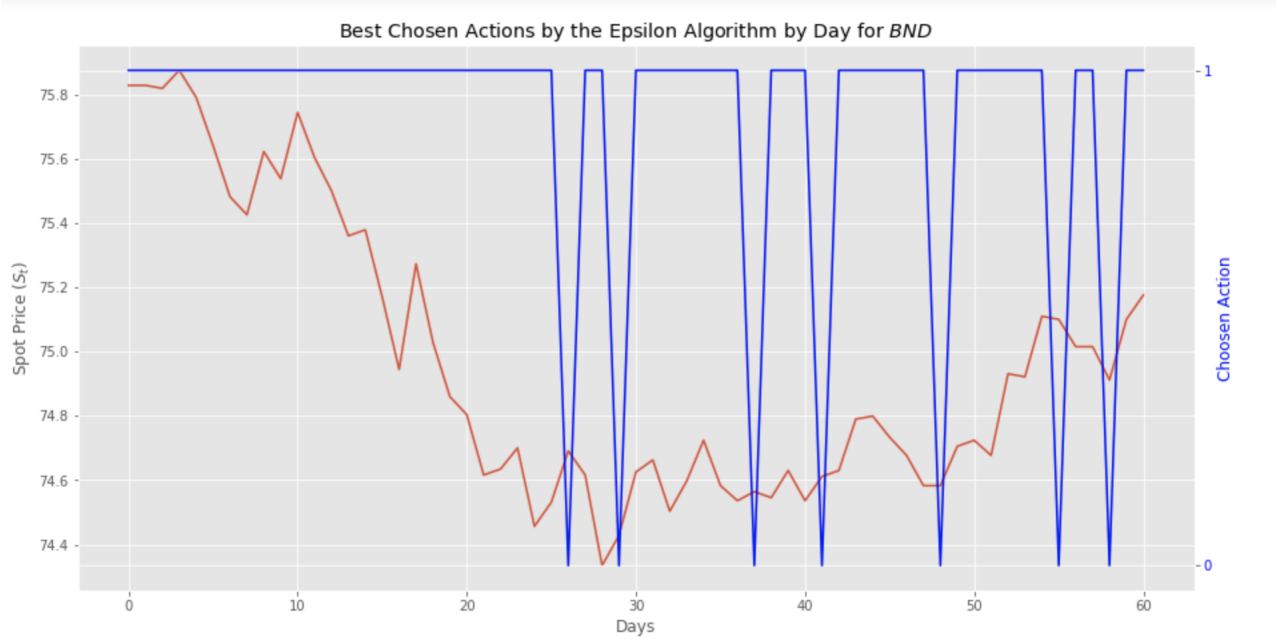

Fig. 5.7: Policy Plot : Normal Volatility [BND] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	-31.49
Black - Scholes	-31.53
Longstaff - Schwartz	-32.96

Table 5.10: Reward Table : Normal Volatility [BND] - Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

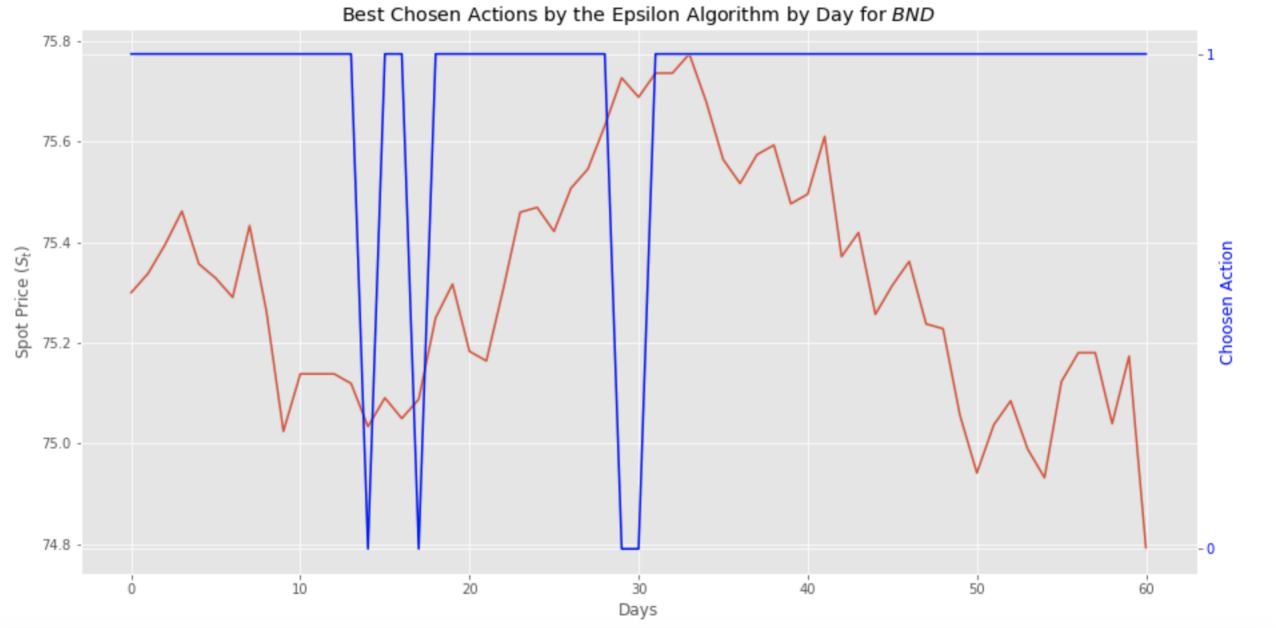

Fig. 5.8: Policy Plot : Low Volatility [BND] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	-20.25
Black - Scholes	-20.24
Longstaff - Schwartz	1.07

Table 5.11: Reward Table : Low Volatility [BND] - Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

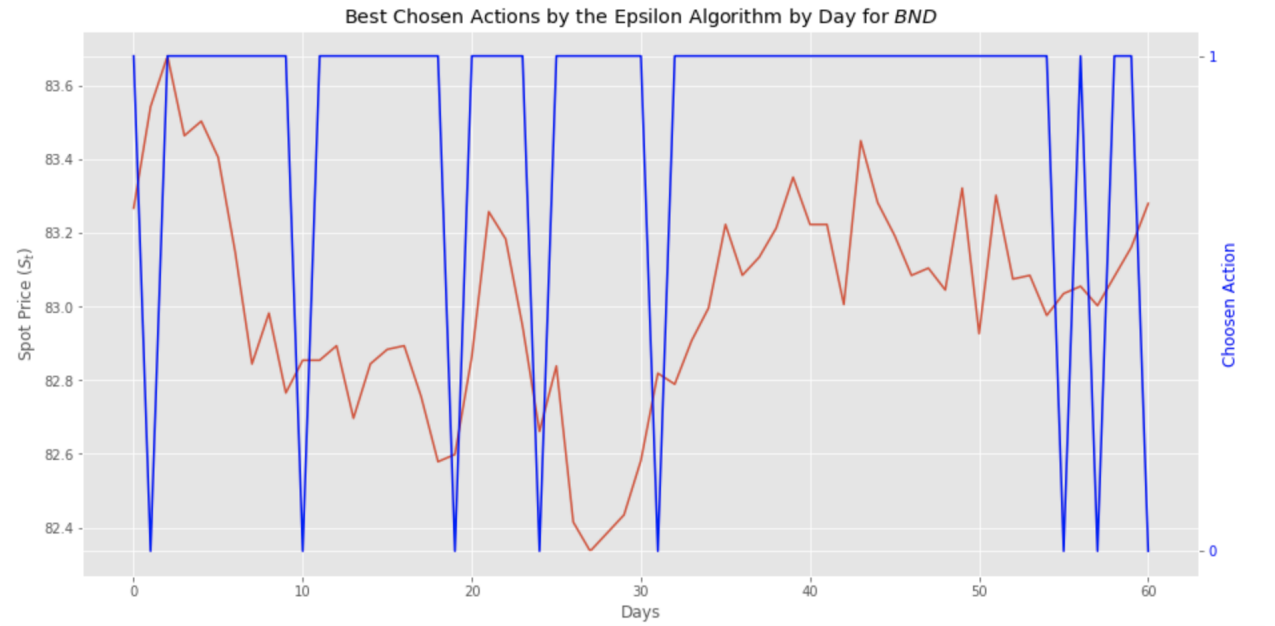

Fig. 5.9: Policy Plot : High Volatility [BND] - Normal Volatility [BND] - On policy choice of best hedging action [1 = set to BS delta, 0 = no change] by our Monte Carlo variant model.

Method	**Reward Value**
Monte Carlo Variant	-26.31
Black - Scholes	-26.32
Longstaff - Schwartz	14.82

Table 5.12: Reward Table : High Volatility [BND] - Comparing rewards from our Monte Carlo variant model with the Black - Scholes model & the Longstaff - Schwartz model.

5.7.1 Observations

We observe the following:

1. Figures 5.7, 5.8, 5.9 represent the policy plots that show the actions taken by our algorithm during a 90 day option time frame.

2. Tables 5.10, 5.11, 5.12 represent the rewards generated by following recommendations of all three policies i.e., Monte Carlo variant - our algorithm, the Black-Scholes formulation ,and the Longstaff-Schwartz algorithm.

3. Tables 5.11 and 5.12 show that the Longstaff-Schwarz algorithm is performing far better than both the Black-Scholes and Monte Carlo variant in low and high volatility situations.

4. The Monte Carlo variant is performing at about the same as the Black - Scholes algorithm probably because of the low overall volatility of the ETF because of which there isn't enough opportunity to benefit from savings in trading and transactions costs.

5. We also notice from the policy diagrams that the Monte Carlo variant is making only fewer adjustments because states are not changing that much frequently.

6. The Longstaff-Schwartz model seems to be outperforming other algorithms probably because of the predictability of the risk frontier that it generates in a typically low volatility environment.

5.8 Overall summary of the results

Following are the key set of observations across all three ETFs that represent the market (VOO), are more volatile than the market (QQQ), or less volatile than the market (BND):

1. Whenever the sum of asset volatility and market volatility is the highest, the proposed Monte Carlo variant algorithm outperforms both the Black - Scholes and the Longstaff-Schwartz.

2. In market situations where prices are constantly increasing, the proposed Monte Carlo variant algorithm does not tend to do so well. Increasing the number of states that we define in our algorithm could help in this situation.

3. In cases where market volatility or asset volatility is low and prices are swinging around a mean; the algorithm performs about the same as Black - Scholes & Longstaff-Schwartz. This seems to be because of the fact that in low volatility situations, the trading cost component does not vary as much. Hence the reinforcement learning algorithm has little incentive to take actions.

4. It is important to maintain a large amount of exploration rate in the proposed Monte Carlo variant algorithm, $\epsilon \approx 0.20$ for optimal performance.

5. The proposed Monte Carlo variant algorithm can be used out of the box with just one tuning parameter to configure i.e., exploration rate ϵ.

CHAPTER 6
THE IMPOSSIBILITY HYPOTHESIS & FUTURE DIRECTIONS

6.1 The Impossibility Hypothesis

The following observations could be made while testing the various scenarios using our algorithm:

1. Combining two different strategies potentially produces better rewards.
2. We are able to design a meta-algorithm that combines the two different strategies.
3. We can extend our algorithm to combine any number of strategies.

Against, this background, we present the following hypothesis.

Statement of the Impossibility Hypothesis

It is impossible to create a delta hedging strategy that uniformly outperforms all other available strategies.

Intuition behind the hypothesis

The key intuition here is akin to the argument that Leo Breiman presents in his paper on "Stacked Regressions". In a world where we have multiple market makers each with their own set of strategies, it is fair to argue that there is some out of sample predictive ability in each of these strategies drawn either from prior observation or analysis. As long as our meta-algorithm is able to pick any one of these strategies for every market situation, we can be pretty sure, but without proof, that adding an $(n+1)^{th}$ strategy to n existing strategies will improve the predictive ability of our meta-algorithm on out of sample data. At least we expect to perform no worse by adding the $(n+1)^{th}$ strategy to the mix.

6.2 Future directions

We see the following research questions emerging out of this work:

1. Our policy construction is a form of Bernoulli shift. Can we use Ornstein's Isomorphism theorem to argue that if two options have nearly (nearness yet to be defined) identical policy signatures over time, does it mean that the underlying stocks (assets) of these options are cointegrated?

2. How can we extend this idea of delta hedging of a single option to hedge a complete portfolio of options?

REFERENCES

[1] T. L. Meng and M. Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, p. 110, Jul 2019. [Online]. Available: http://dx.doi.org/10.3390/data4030110

[2] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, vol. 81, no. 3, pp. 637–654, 1973. [Online]. Available: http://www.jstor.org/stable/1831029

[3] M. R. John C. Cox, Stephen A. Ross, "Option pricing: A simplified approach," *Journal of Financial Economics*, vol. 7, no. 3, pp. 229–263, 1979. [Online]. Available: https://doi.org/10.1016/0304-405X(79)90015-1

[4] S. H. Les Clewlow, "Option pricing: A simplified approach," *Journal of Economic Dynamics and Control*, vol. 21, no. 8–9, pp. 1353–1376, 1997. [Online]. Available: https://doi.org/10.1016/S0165-1889(97)00030-4

[5] S. L. Heston and S. Nandi, "A closed-form garch option valuation model," *The Review of Financial Studies*, vol. 13, no. 3, pp. 585–625, 2000. [Online]. Available: http://www.jstor.org/stable/2645997

[6] E. S. S. Francis A. Longstaff, "Valuing american options by simulation: A simple least-squares approach," *The Review of Financial Studies*, pp. 113–147, 2001. [Online]. Available: https://people.math.ethz.ch/~hjfurrer/teaching/LongstaffSchwartzAmericanOptionsLeastSquareMonteCarlo.pdf

[7] D. B. M. P Carr, "Option valuation using the fast fourier transform," *Journal of Computational Finance*, vol. 2, no. 4, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.348.4044&rep=rep1&type=pdf

[8] S. Liu, C. Oosterlee, and S. Bohte, "Pricing options and computing implied volatilities using neural networks," *Risks*, vol. 7, no. 1, p. 16, Feb 2019. [Online]. Available: http://dx.doi.org/10.3390/risks7010016

[9] D. B. Madan and W. Schoutens, "Conic option pricing," *The Journal of Derivatives*, vol. 25, no. 1, pp. 10–36, 2017. [Online]. Available: https://jod.pm-research.com/content/25/1/10

[10] W. F. Sharpe, "Capital asset prices: A theory of market equilibrium under conditions of risk*," *The Journal of Finance*, vol. 19, no. 3, pp. 425–442, 1964. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1964.tb02865.x

[11] D. Duffie, "Black, merton and scholes: Their central contributions to economics," *The Scandinavian Journal of Economics*, vol. 100, no. 2, pp. 411–423, 1998. [Online]. Available: http://www.jstor.org/stable/3440890

[12] M. Bellalah, *Derivatives, Risk Management and Value*. World Scientific, 2010.

[13] Y. YOO, "Stochastic calculus and black-scholes model." [Online]. Available: http://math.uchicago.edu/~may/REU2017/REUPapers/Yoo.pdf

[14] R. H. Chan, "Black-scholes equations." [Online]. Available: https://www.math.cuhk.edu.hk/~rchan/teaching/math4210/chap08.pdf

[15] S. R. Dunbar, "Solution of the black-scholes equation." [Online]. Available: http://www.math.unl.edu/~sdunbar1/MathematicalFinance/Lessons/BlackScholes/Solution/solution.pdf

[16] ——, "Sensitivity, hedging and the greeks." [Online]. Available: http://www.math.unl.edu/~sdunbar1/MathematicalFinance/Lessons/BlackScholes/Greeks/greeks.pdf

[17] I. Stewart, "The mathematical equation that caused the banks to crash." [Online]. Available: https://www.theguardian.com/science/2012/feb/12/black-scholes-equation-credit-crunch

[18] V. P. Hau L. Lee and S. Whang, "The bullwhip effect in supply chains." [Online]. Available: https://sloanreview.mit.edu/article/the-bullwhip-effect-in-supply-chains/

[19] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2018. [Online]. Available: https://books.google.com/books?id=sWV0DwAAQBAJ

[20] R. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 8, pp. 716–719, 1952. [Online]. Available: https://doi.org/10.1073/pnas.38.8.716

[21] R. A. Howard, *Dynamic programming and Markov processes.* MIT Press, 1960.

[22] A. Klopf, "Brain function and adaptive systems - a heterostatic theory." [Online]. Available: https://pdfs.semanticscholar.org/0c1a/ccd2ef7218534a1726a8de7d6e7c14271a75.pdf

[23] R. Koppula, "Exploration vs. exploitation in reinforcement learning." [Online]. Available: https://www.manifold.ai/exploration-vs-exploitation-in-reinforcement-learning

[24] D. e. a. Silver, "Mastering the game of go without human knowledge," *Nature*, vol. 550, p. 354–359, 2017. [Online]. Available: https://doi.org/10.1038/nature24270

[25] J. P. Jens Kober, J. Andrew Bagnell, "Reinforcement learning in robotics: A survey," *International Journal of Robotics Research*, 2013. [Online]. Available: https://www.ias.informatik.tu-darmstadt.de/uploads/Publications/Kober_IJRR_2013.pdf

[26] S. Padakandla, P. K. J., and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Applied Intelligence*, Jun 2020. [Online]. Available: http://dx.doi.org/10.1007/s10489-020-01758-5

[27] L. B. Statistics and L. Breiman, "Random forests," in *Machine Learning*, 2001, pp. 5–32.

APPENDICES

APPENDIX A

Reference to the python implementation of the model

All code files are available publicly at https://github.com/ronaktali/MSThesis