Data-Driven Models for Discrete Hedging Problem

by

Ke Nian

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2020

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	undetermined Professor, School of Computer Science, University of Undetermined
Supervisor(s):	Yuying Li Professor, School of Computer Science, University of Waterloo Thomas F. Coleman Professor, Department of Combinatoric and Optimization, University of Waterloo
Internal Examiner:	Justin W.L. Wan Professor, School of Computer Science, University of Waterloo
Internal Examiner:	Yaoliang Yu Assistant Professor, School of Computer Science, University of Waterloo
Internal-External Member:	undetermined Professor, Dept. of Math of Science, University of Waterloo

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Options hedging is a critical problem in financial risk management. The prevailing approach in financial derivative pricing and hedging has been to first assume a parametric model describing the underlying price dynamics. An option model function V is then calibrated to current market option prices and various sensitivities are computed and used to hedge the option risk. It has been recognized that computing hedging position from the sensitivity of the calibrated model option value function is inadequate in minimizing variance of the option hedge risk, as it fails to capture the model parameter dependence on the underlying price. In this thesis, we demonstrate that this issue can exist generally when determining hedging position from the sensitivity of the option function, either calibrated from a parametric model from spot market option prices or estimated nonparametricaly from historical option prices. Consequently the sensitivity of the estimated model option function typically does not minimize variance of the hedge risk, even instantaneously, unless the parameters dependence is addressed. We propose several data-driven approaches to directly learn a hedging function from the market data by minimizing certain measure of the local hedging risk and total hedging risk. This thesis will focus on answering the following questions: 1) Can we efficiently build direct data-driven models for discrete hedging problem that outperform existing state-of-art parametric hedging models? 2) Can we incorporate feature selection and feature extraction into the data-driven models to further improve the performance of the discrete hedging? 3) Can we build efficient models for both the one-step hedging problem and multi-steps hedging problem based on the state-of-art learning framework such as deep learning framework and kernel learning framework?

Using the S&P 500 index daily option data for more than a decade ending in August 2015, we first propose a direct data-driven approach [118] based on kernel learning framework and we demonstrate that the proposed method outperforms the parametric minimum variance hedging method proposed in [90], as well as minimum variance hedging corrective techniques based on stochastic volatility or local volatility models. Furthermore, we show that the proposed approach achieves significant gain over the implied Black-Sholes delta hedging for weekly and monthly hedging.

Following the direct data-driven kernel learning approach [118], we propose a robust encoder-decoder Gated Recurrent Unit (GRU), GRU_δ, for optimal discrete option hedging. The proposed GRU_δ utilizes the Black-Scholes model as a pre-trained model and incorporates sequential information and feature selection. Using the S&P 500 index European option market data from January 2, 2004 to August 31, 2015, we demonstrate that the weekly and monthly hedging performance of the proposed GRU_δ significantly surpasses that of the data-driven minimum variance (MV) method in [90], the regularized kernel data-driven model [118], and the SABR-Bartlett method [77]. In addition,the daily hedging performance of the proposed GRU_δ also surpasses that of MV methods in [90] based on parametric models, the kernel method [118] and SABR-Bartlett method [77].

Lastly, we design a multi-steps data-driven models $\text{GRU}_{\text{total}}$ based on the GRU_δ to hedge the option discretely until the expiry. We utilize SABR model and Local Volatility Function

(LVF) to augment existing market data and thus alleviate the problem for lack of market option information. The augmented market data is used to train a sufficient multi-steps total hedging model $\text{GRU}_{\text{total}}$ that outperform the one-step hedging model GRU_{δ} when we evaluate the hedging performance on the expiries. We further compare the total hedging model $\text{GRU}_{\text{total}}$ built based on purely market underlying information with the total hedging model $\text{GRU}_{\text{total}}$ built based on both market underlying information and augmented option information to indicate the importance of market option information in determining the data-driven option hedging position.

Acknowledgements

I would like to thank all the little people who made this thesis possible.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Options hedging is a critical problem in financial risk management. The state-of-art approaches in financial derivative pricing and hedging rely heavily on parametric assumptions describing the dynamics of underlying asset . The common practice is to calibrate an option pricing function based on the specific parametric model and compute various sensitivities to hedge the option risk. For example, the sensitivity of the option value function to the underlying price is used in delta hedging. Ideally, the value of an option written on the underlying asset can be perfectly replicated by a hedging portfolio consisting of the underlying asset and the risk-free asset, when the market is complete [135]. In practice, such perfect scenarios does not exist and we have to rebalance the hedging portfolio discretely instead of continuously due to the existence of the transaction cost. The practice of adjusting the hedging portfolio discretely is often referred to as discrete hedging.

There are many parametric models proposed to describe the dynamics of underlying asset. The original and most celebrated parametric Black-Scholes (BS) model uses a constant volatility [18, 114], which is shown to produce inaccurate option prices for deeply out-of-the-money options and deeply in-the-money options [68]. In addition, the BS model is unable to capture the non-zero correlation between the volatility and the underlying asset price [63, 20]. The practitioner's BS delta hedging approach sets the constant volatility in the BS model to the implied volatility calibrated to the market price at the time of re-balancing. Many alternative parametric models have been proposed to improve the BS model, including the Stochastic Volatility (SV) model [78, 81, 89, 12], the Local Volatility Function (LVF) model [38, 53, 130, 55], and the jump model [79, 98]. Unfortunately, all models have been shown to have their limitations in accurately modeling option market prices.

Errors in the option value model have significant implications in hedging. Consider, for example, when the hedging position is computed from the sensitivity of the option value function calibrated at the hedging time, the computed hedging position only depends on the assumed underlying price model and the current market option prices. Unless the assumed model for the underlying price is exact and all assumptions that results in the option pricing function are all valid, the option function calibrated at the hedging time cannot predict the future option market

price.

Specifically, assume that $V(S,t,T,K,r,q;\theta^*)$ is the option value function and θ^* is the vector of model parameters of the assumed option pricing model and, at the hedging time t, option calibration ensures that

$$V(S,t,T,K,r,q;\theta^*) = V^{mkt}_{t,T,K} \tag{1.0.1}$$

where V^{mkt} denotes the actual market option price, t is the current trading time, K is strike price, T is the expiry time, and S is the underlying price input used in the option pricing function. We use $V^{mkt}_{t,T,K}$ to indicate it is the market price at time t for the expiry time T and strike K. The option value function $V(S,t,T,K,r,q;\theta^*)$, calibrated at the hedging time t, does not ensure that $\frac{dV}{dS}$ equals to $\frac{dV^{mkt}}{dS}$, which is indeed unknown. Furthermore, $\frac{dV}{dS}$ is difficult to be computed exactly due to the dependence of the calibrated model parameter on the underlying price [118, 38, 90]. The missing sensitivity $\frac{\partial\theta^*}{\partial S}$, is difficult to account for and is often ignored, though for some models, corrections have been proposed to account for the dependence [90, 77, 15].

Since machine learning algorithms usually do not impose assumptions on the model to be learned, they have recently been adopted to determine an option value function directly from the market data, with the goal of avoiding the model misspecification issues from the parametric modeling approach e.g.,[73, 65, 92]. Unfortunately, using nonparametric learning, hedging positions still need to be computed from the sensitivity of the model value function. While no assumption is explicitly made for the dynamics of underlying asset, the option value function is determined by data through cross-validation, leading to training errors. Since there is no assurance that the sensitivity of the learned option value function with regards to underlying asset matches that the sensitivity of the market option price, the parameters of the model learned directly from data can similarly exhibit dependence on the underlying price. When the hedging position is computed from the partial derivative of the data-driven option value function, e.g., [92], this dependence cannot be accounted for and again is ignored. Hence, option hedging risk remains insufficiently minimized.

Furthermore, using option delta $\frac{\partial V}{\partial S}$ as the hedging position becomes inadequate when discrete hedging is performed in practice, particularly when rebalancing becomes infrequent. Instead, optimal discrete hedging strategy can be determined by determining hedging strategy directly using an appropriate objective in the discrete hedging context, e.g., minimizing the variance of the hedging error, needs to be chosen [90, 7, 72].

In hedging, the ultimate goal is to discover a hedging strategy which minimizes the hedging error which is measured by the market option and underlying prices. With the increasing availability of market option prices, a timely question arises: is it possible to learn optimal hedging positions directly from market option price and underlying price data? Up until now, research in learning the hedging position directly from market data is scarce. Recently, a data-driven approach [90] is proposed to learn a parametric model for the minimum variance delta hedging based on the analysis of the BS option greeks and underlying market prices. However, the proposed parametric model focuses on the instantaneous hedging error analysis in a parametric model framework.

In this thesis, we study the discrete option hedging problem by explicitly focusing on the issues arising from model specification errors and model parameter dependence. We illustrate that the inability to minimize variance of the hedging error, when determining hedging position from option value function from a parametric model, is also shared by an option model estimated from a nonparametric method. Although a nonparametric modeling approach to option value can potentially lead to smaller mis-specification error, we illustrate that non-parametric model parameters can similarly depend on the underlying. Consequently the sensitivity of the optimally estimated option value function will not lead to minimization of option hedging risk. Furthermore, the estimated pricing function inevitably has errors, due to both model mis-specification, discretization, and numerical roundoff errors. The error in the value function can potentially be substantially magnified in computing partial derivatives as hedging positions.

We explore several direct market data-driven approaches to bypass challenges mentioned above to achieve effective hedging performance. We first propose a data-driven kernel learning approach [118] to learn a local risk minimization hedging model directly from the market data observed at the hedging time t. We learn a hedging function from the market data by minimizing the empirical local hedging risk with a suitable regularization. The local risk corresponds directly to the variance of the hedging error in the discrete rebalancing period. A novel encoder-decoder RNN model GRU_{δ} [119], to extract both sequential and local features at hedging time t from market prices, is later proposed to learn option hedging positions directly from the market. We include a feature weighting procedure to select the most relevant local features at hedging time t and sequential features for the sequential data-driven model GRU_{δ}. Lastly, in order to deal with multi-steps discrete total hedging scenarios where we hedge until the expiry of the option [120], we enhance our sequential local hedging model GRU_{δ} to be $\text{GRU}_{\text{TOTAL}}$. We compared our data-driven approaches with the parametric approaches and demonstrate the effectiveness of the data-driven hedging models in terms of both local hedging risk and total hedging risk.

1.1 Contribution

The contributions with respect to the data-driven kernel hedging model [118] are summarized below:

- We analyze and discuss implications from model mis-specification in the option value function for discrete option hedging. We illustrate challenges in accounting for the dependence of the calibrated model parameters on the underlying, which arises due to model mis-specification.
- We analyze a regularized kernel network for option value estimation and illustrate that the partial derivative of the estimated value function with respect to the underlying similarly does not minimize variance of the hedging risk in general, even infinitesimally.
- We propose a data-driven approach to learn a hedging position function directly by minimizing the variance of the local hedging risk. Specifically we implement a regularized

spline kernel method $\mathrm{DKL}_{\mathrm{SPL}}$ to nonparametrically estimate the hedging function from the market data.

- Using synthetic data sets, we compare daily, weekly, and monthly hedging performance using the kernel direct data-driven hedging approach with the performance of the indirect approach where hedging positions are computed from the sensitivity of the nonparametric option value function. In particular, we present computational results which demonstrate that the direct spline kernel hedging position learning outperforms the hedging position computed from the sensitivity of the spline kernel option value function.

- Using S&P 500 index option market data for more than a decade ending in August 31, 2015, we demonstrate that the daily hedging performance of the direct spline kernel hedging function learning method surpasses that of the minimum variance quadratic hedging formula proposed in [90], as well as corrective methods based on LVF and SABR implemented in [90].

- We also present weekly and monthly hedging results using the S&P 500 index option market data and demonstrate significant enhanced performance over the BS implied volatility hedging.

The contributions with respect to the data-driven sequential hedging model [119] are summarized below:

- We propose a novel encoder-decoder RNN model, to extract both sequential and current features from market prices, to learn option hedging positions directly from the market. We include a feature weighting procedure to select the most relevant local features and sequential time series features for the data-driven model.

- To ensure robust learning, we use the Huber loss function as the learning objective, adaptively setting the error resolution parameter to the Black–Scholes hedging error, allowing it to varying from data instance to data instance. Furthermore, the proposed GRU_{δ} can be updated more frequently than the data-driven model in [118] to account for the market shifts.

- Using the S&P 500 index option market data from January 2, 2004 to August 31st, 2015, we demonstrate that the weekly and monthly hedging performance of the proposed GRU_{δ} significantly surpasses that of the data-driven minimum variance (MV) method in [90], the regularized kernel data-driven model [118], and the SABR-Bartlett method [15].

- Using the S&P 500 index option market data from January 2, 2004 to August 31st, 2015, we demonstrate that the daily hedging performance of the proposed GRU_{δ} surpasses that of the minimum variance quadratic hedging method proposed in [90], the corrective methods based on LVF and SABR implemented in [90],the SABR-Bartlett method [15], as well as the data-driven model in [118].

- To motivate the roles of each major component of the proposed GRU_δ, we demonstrate performance sensitivity through computational experiments. In addition, we illustrate and analyze the relative importance of selected features.

The contributions with respect to the data-driven total hedging model [120] are summarized below:

- We enhance the sequential data-driven local hedging model GRU_δ [119] to cope with total hedging scenarios where we rebalance multiple times until the expiries of the options.
- We augment the market data using SABR model and Local Volatility Function to cope with the challenges of lacking market option data.
- Using the S&P 500 index call option market data from January 2, 1996 to August 31st, 2015, we demonstrate that the weekly, bi-weekly and monthly hedging performance of the proposed total hedging model $\text{GRU}_{\text{TOTAL}}$ surpasses that of sequential data-driven local hedging model GRU_δ [119] and the SABR-Bartlett method [15], when the hedging performance is evaluated on the expiries of the option.
- We compare $\text{GRU}_{\text{TOTAL}}$ based on market option information with $\text{GRU}_{\text{TOTAL}}$ based on purely underlying asset information and indicate the importance of market option information in determining the total hedging position.

1.2 Outline

The remainder of the thesis is organized as follows. Chapter 2 reviews basic concept of derivative pricing models, discrete hedging problems, local and total hedging risk and various existing parametric approaches to hedge options. Chapter 3 discusses the kernel learning framework and introduces the data-driven kernel local hedging model DKL_{SPL}. Chapter 4 discusses the Recurrent Neural Network(RNN) framework and introduces the data-driven sequential local hedging model GRU_δ. Chapter 5 discuss the empirical results from the data-driven sequential local hedging model GRU_δ and data-driven kernel local hedging model DKL_{SPL}.

Chapter 6.4 introduces the data-driven sequential total hedging model $\text{GRU}_{\text{TOTAL}}$ and presents the empirical comparisons between local hedging model GRU_δ and total hedging model $\text{GRU}_{\text{TOTAL}}$. Chapter 6.4 also discusses the challenges of using market data to build data-driven total hedging models and the data augmentation procedure to cope with the challenges. We conclude in Chapter 7 with summary remarks and potential extensions.

Chapter 2

Background

In this chapter, we review the basic concept of option pricing models and discuss the problem of pricing model parameters dependence on underlying asset. In addition, we specify the discrete hedging problem and define the total and local hedging risk. Most of the discussion in this chapter are drawn from [135, 81, 15, 77, 78].

2.1 Option Pricing Model

2.1.1 Black-Scholes Model

A European style call or put option gives its buyer the right to buy the underlying asset on the option expiry with a strike price. Let the strike price be K and the S_T be the underlying price at expiry T. The payoff of call options is :

$$\max(S_T - K, 0)$$

The payoff of put options is:

$$\max(K - S_T, 0)$$

Black and Scholes [18] drive the famous closed-form pricing formula for European options. They show that one can construct a riskless portfolio consisting of one option and shares of the underlying asset. The riskless portfolio needs to be continuously adjusted so that the number of shares always equal to the partial derivative of the option pricing function with regards to the underlying asset. No-arbitrage condition implies that the the return of the riskless portfolio must be equal to the risk-free interest rate. This leads to the renowned Black-Scholes (BS) partial differential equation and the closed-form pricing formula.

More specifically, under BS model, it is assumed that the underlying asset price follows a geometric Brownian motion:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

where W_t is a standard Brownian motion, μ is the constant drift rate of the asset and σ is the constant volatility of the asset. We can easily show that:

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

Let C_{BS} be the option value function for call option which depends on t and S. For notational simplicity, we have:

$$\frac{\partial C_{BS}}{\partial t} = \frac{\partial C_{BS}}{\partial t}(t, S_t) \ , \frac{\partial C_{BS}}{\partial S_t} = \frac{\partial C_{BS}}{\partial S}(t, S_t) \ , \frac{\partial^2 C_{BS}}{\partial S_t^2} = \frac{\partial^2 C_{BS}}{\partial S^2}(t, S_t) \ , C_t^{BS} = C_{BS}(t, S_t)$$

Follows Ito's Lemma [135],we have:

$$dC_t^{BS} = \left(\frac{\partial C_{BS}}{\partial t} + \mu S_t \frac{\partial C_{BS}}{\partial S_t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 C_{BS}}{\partial S_t^2} \right) dt + \sigma S_t \frac{\partial C_{BS}}{\partial S_t} dW_t$$

Now consider a specific portfolio, called the delta-hedge portfolio, consisting of being short one call option and long $\frac{\partial C_{BS}}{\partial S_t}$ shares at time t. The total value of the delta-hedge portfolio P^δ at time t is:

$$P_t^\delta = -C_t^{BS} + S_t \frac{\partial C_{BS}}{\partial S_t}$$

The instantaneous profit or loss is:

$$dP_t^\delta = -\left(\frac{\partial C_{BS}}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 C_{BS}}{\partial S_t^2} \right) dt$$

Assume there is a riskless asset with constant rate of return r, which is also known as risk-free interest rate. We can see that the delta-hedge portfolio P_δ is also riskless because the diffusion term associated with dW_t is dropped. Under no-arbitrage condition, two riskless investment must earn the same rate of return so we must have:

$$dP_t^\delta = rP_t^\delta dt$$

This leads to the Black-Scholes partial differential equation:

$$\frac{\partial C_{BS}}{\partial t} + \frac{1}{2}\sigma^2 S_t^2 \frac{\partial^2 C_{BS}}{\partial S_t^2} + rS\frac{\partial C_{BS}}{\partial S_t} - rC_t^{BS} = 0 \tag{2.1.1}$$

The solution with European call option is the well-known Black-Scholes pricing formula:

$$Call = S\mathcal{N}(d_1) - e^{-r(T-t)}K\mathcal{N}(d_2) \tag{2.1.2}$$

where $\mathcal{N}$ is the cumulative density function of the standard normal distribution

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t}$$

Similarly, we can derive the Black-Scholes pricing formula for European put option to be:

$$Put = e^{-r(T-t)}K\mathcal{N}(-d_2) - S\mathcal{N}(-d_1) \tag{2.1.3}$$

Alternatively, we can derive the Black-Scholes formula under the risk-neutral pricing framework. As the name suggests, under a risk-neural measure Q, all agents in the economy are neutral to risk, so that they are indifferent between investments with different risk as long as these investments have the same expected return. Under a risk-neutral measure, all tradable assets should have the same expected rate of return as the risk-free asset, which earns the risk-free interest rate r. The derivative price can thus derived from the expected payoff, discounted back to the current time at the risk-free rate r. It can been shown that [135], following the Black-Scholes assumption, there is a unique risk-neutral probability measure Q that is equivalent to the actual physical probability measure. Under risk-neutral pricing framework, we have:

$$Call = e^{-r(T-t)}E^Q[\max(S_T - K, 0)] \tag{2.1.4}$$

$$Put = e^{-r(T-t)}E^Q[\max(K - S_T, 0)] \tag{2.1.5}$$

where $E^Q[\cdot]$ is the expectation under the risk-neutral measure Q. More specifically, define Θ_1 to be the market price of risk:

$$\Theta_1 = \frac{\mu - r}{\sigma} \tag{2.1.6}$$

We change the original Brownian motion dW_t to $d\hat{W}_t$ with

$$d\hat{W}_t = dW_t + \Theta_1 dt \tag{2.1.7}$$

The underlying dynamic will have the drift to be the risk-free interest rate r.

$$dS_t = rS_t dt + \sigma S_t d\hat{W}_t$$

It can be shown that $d\hat{W}_t$ is the Brownian motion under the risk-neutral measure Q defined through Radon-Nikodym derivative via Girsanov's theorem [135]. Using the fact that the drift rate of underlying asset dynamic under risk-neutral measure Q is r, following (2.1.4) and (2.1.5), we can arrive at the same pricing formula as (2.1.2) and (2.1.3). Interest reader can refer to [135] for more details about risk-neutral pricing and change from physical measure to risk-neutral measure Q.

Since actual drift μ is irrelevant in determining the option price under Black-Scholes framework, in this thesis, we assume we are dealing with risk-neutral measure Q and the drift of underlying dynamic is always the risk-free interested r. Lastly, to include the effect of dividend,

we can rewrite the d_1 and d_2 in the Black-Scholes pricing formula as:

$$d_1 = \frac{\ln(S/K) + ((r-q) + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}, \;\; d_2 = d_1 - \sigma\sqrt{T-t}$$

where q is the given annual dividend yield. In this thesis, we use $V_{BS}(S,t,T,K,r,q;\sigma)$ to denote the European Black-Scholes pricing function regardless of the call or put nature.

Although, Black-Scholes framework provides a nice close-form formula, the real markets are never as ideal as the assumption of Black-Scholes model. Empirical evidence indicates markets often violate the assumption of Black-Scholes model. The two major aspects that has been criticized about Black-Scholes model are:

1. The constant volatility does not hold in real market. In practice, the implied volatility σ^{imp}, which equates the Black-Scholes option price $V_{BS}(S,t,T,K,r,q;\sigma)$ to market option price $V^{mkt}_{t,T,K}$, is often used to make sure that Black-Scholes price match the market observation. However, one can often find that the implied volatility σ^{imp} tends to differ across different strikes and expiries. This breaks down the assumption of a constant volatility

2. Transaction cost exists in real market. Due the existence of transaction cost, continuously adjusting the shares of underlying is prohibitively expensive. Therefore, the argument of the Black-Scholes theory falls apart.

The market deviations from the assumption of Black-Scholes model motivates people to propose various approaches for relaxing the assumptions of Black and Scholes. These attempts include, but not limited to, local volatility models [38, 53, 130, 55], stochastic volatility models [78, 81, 89, 12], jump diffusion models [79, 98] and nonparametric pricing models based on regression [154, 16, 73, 65, 106]. In the following section, we discuss two stochastic volatility models relevant to this thesis, Heston model and SABR model, which provide efficient closed-form solutions for the option price similar as Black-Scholes model.

2.1.2 Heston Model

Heston [81] proposed a version of the stochastic volatility model, which has become quite popular to model the volatility smiles. The volatility smiles refer to the phenomenon that the options in real market whose strike prices differ substantially from the underlying asset price tend to have higher implied volatilities than options whose strike prices are close to the underlying asset price under Black–Scholes models. One of the key reason for its popularity is that European call and put option under Heston model have closed-form solution which makes the calibration of the model computationally efficient and accurate. The Heston model assumes that the underlying, S_t follows a Black-Scholes type stochastic process, but with a stochastic variance υ_t that follows a

Cox, Ingersoll, Ross (CIR) process [44].

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{\upsilon} S_t dW_t \\ d\upsilon_t &= \kappa(\overline{\upsilon} - \upsilon_t)dt + \eta\sqrt{\upsilon_t} dZ_t \\ E[dZ_t dW_t] &= \rho dt \end{aligned}$$

These parameters are described as follows:

- μ is the drift coefficient of the underlying asset
- $\overline{\upsilon}$ is the long term mean of variance
- κ is the rate of mean reversion
- η is the volatility of volatility
- S_t is the underlying asset price
- υ_t is the instantaneous variance
- W_t and Z_t are correlated Wiener process with correlation coefficient ρ

Similarly, the Heston dynamics can be transformed to be under a risk-neutral measure Q. Heston [81] assumes that the market price of volatility risk is proportional to the volatility $\sqrt{\upsilon_t}$:

$$\Theta_2 = \frac{\lambda}{\eta}\sqrt{\upsilon_t} \tag{2.1.8}$$

λ is a parameter used to generate the market price of volatility risk. Recall that market price of risk is:

$$\Theta_1 = \frac{\mu - r}{\sqrt{\upsilon_t}}$$

It can be shown that a risk-neutral measure Q can be defined through Radon-Nikodym derivative via Multi-dimensional Girsanov's theorem [135] using the Θ_1 and Θ_2.

Therefore, Heston model under a risk-neutral measure Q is:

$$\begin{aligned} dS &= rSdt + \sqrt{\upsilon} S d\hat{W} \\ d\upsilon &= \kappa^*(\overline{\upsilon}^* - \upsilon)dt + \eta\sqrt{\upsilon} d\hat{Z} \\ E[d\hat{Z} d\hat{W}] &= \rho dt \end{aligned} \tag{2.1.9}$$

where

$$\kappa^* = \kappa + \lambda, \overline{\upsilon}^* = \frac{\kappa\overline{\upsilon}}{\kappa + \lambda}$$

$$d\hat{W} = dW + \Theta_1 dt$$

$$d\hat{Z} = dZ + \Theta_2 dt$$

Similar to Black-Scholes model, Heston model have closed-form solutions. The closed formed solution for European call option under the risk-neutral measure Q is

$$Call = S\,N_1 - Ke^{-r(T-t)}\,N_2 \tag{2.1.10}$$

Let us define the imaginary unit $\mathscr{I}^2 = -1$. Then the N_1 and N_2 are defined as:

$$N_j = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-\mathscr{I}\varphi \ln K} f_j(S,\upsilon,t,T;\varphi)}{\mathscr{I}\varphi}\right] d\varphi; \;\; , j = 1,2$$

with $Re[\cdot]$ denoting the real part. The characteristic function $f_j(S,\upsilon,t,T;\varphi)$ is :

$$f_j(S,\upsilon,t,T;\varphi) = e^{A_j(t,T,\varphi)+B_j(t,T,\varphi)\upsilon+\mathscr{I}\varphi \ln S}\;, j = 1,2$$

Where:

$$A_j(t,T,\varphi) = r\varphi(T-t) + \frac{\kappa^*\overline{\upsilon}^*}{\eta^2}\left\{(b_j - \rho\eta\varphi\mathscr{I} + d_j)(T-t) - 2\ln\left[\frac{1-g_j e_j^d(T-t)}{1-g_j}\right]\right\}$$

$$B_j(t,T,\varphi) = g_j\left[\frac{1-e_j^d(T-t)}{1-g_j e^{d_j(T-t)}}\right]$$

$$g_j = \frac{b_j - \rho\varphi\mathscr{I} + d_j}{b_j - \rho\varphi\mathscr{I} - d_j}$$

$$d_j = \sqrt{(\rho\eta\varphi\mathscr{I} - b_j)^2 - \eta^2(2u_j\varphi - \varphi^2)}$$

$$u_1 = 0.5, u_2 = -0.5, b_1 = \kappa^* - \rho\upsilon, b_2 = \kappa^*$$

European put option price can be derived from the call-put parity:

$$Put = Call - S + Ke^{-r(T-t)}$$

The parameters to be calibrated from the market option prices are $\{\upsilon, \kappa^*, \overline{\upsilon}^*, \eta, \rho\}$ where υ is the initial instantaneous variance, $\overline{\upsilon}^*$ is the long term mean of variance under risk-neutral measurement Q, κ^* is the rate of mean reversion under risk-neutral measurement Q, η is the volatility of volatility, and ρ is correlation.

Under Black-Scholes model, an option is dependent on tradable asset S_t. The randomness in option value is solely determined by the randomness of the asset S_t. Such uncertainly can be hedged by continuously adjusting the shares of underlying asset as we have discussed in section 2.1.1. This implies a complete market [135]. Under a stochastic volatility model such as Heston model, the uncertainty of option value comes from both the underlying asset S_t and the volatility (or variance υ_t as in Heston model). The volatility itself is not tradable which implies an incomplete market under stochastic volatility model. The incompleteness implies the

risk-neutral measure is not unique. In other words, λ is not unique. Different choices of λ will lead to different risk-neutral measurements. However, one can assume a risk-neutral measure Q exists and calibrate the Heston model to match the the market option prices directly using the dynamics in (2.1.9) without specifying the λ. In this way, λ has been implied and embedded into the calibrated model parameters κ^* and $\overline{\upsilon}^*$. Interested reader can refer to [81, 66] for more details of risk-neutral pricing under Heston model.

Again, given the annual dividend yield q, the above heston pricing formula can be rewritten as:

$$Call = Se^{-q(T-t)}N_1 - Ke^{-r(T-t)}N_2$$

$$N_j = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left[\frac{e^{-\mathscr{I}\varphi \ln K} f_j(S,\upsilon,t,T;\varphi)}{\mathscr{I}\varphi}\right] d\varphi; \quad , j = 1,2$$

with $Re[\cdot]$ denoting the real part. The characteristic function $f_j(S,\upsilon,t,T;\varphi)$ is :

$$f_j(S,\upsilon,t,T;\varphi) = e^{A_j(t,T,\varphi)+B_j(t,T,\varphi)\upsilon+\mathscr{I}\varphi \ln S}, j = 1,2$$

Where:

$$A_j(t,T,\varphi) = (r-q)\varphi(T-t) + \frac{\kappa^*\overline{\upsilon}^*}{\eta^2}\left\{(b_j - \rho\eta\varphi\mathscr{I} + d_j)(T-t) - 2\ln\left[\frac{1-g_j e_j^d(T-t)}{1-g_j}\right]\right\}$$

$$B_j(t,T,\varphi) = g_j\left[\frac{1-e_j^d(T-t)}{1-g_j e^{d_j(T-t)}}\right]$$

$$g_j = \frac{b_j - \rho\varphi\mathscr{I} + d_j}{b_j - \rho\varphi\mathscr{I} - d_j}$$

$$d_j = \sqrt{(\rho\eta\varphi\mathscr{I} - b_j)^2 - \eta^2(2u_j\varphi - \varphi^2)}$$

$$u_1 = 0.5, u_2 = -0.5, b_1 = \kappa^* - \rho\upsilon, b_2 = \kappa^*$$

European put option price can be derived from the call-put parity:

$$Put = Call - Se^{-q(T-t)} + Ke^{-r(T-t)}$$

In this thesis, we deal with Heston model under the risk-neural measurement Q. For simplicity, in this thesis, we use $V_{Heston}(S,t,T,K,r,q;\upsilon,\kappa^*,\overline{\upsilon}^*,\eta,\rho)$ to denote the European Heston pricing function regardless of the call or put nature.

2.1.3 SABR Model

The SABR model [78] is another stochastic volatility model, which attempts to capture the volatility smiles in derivatives markets. The name stands for "Stochastic Alpha, Beta, Rho",

referring to the parameters of the model. The SABR model is another popular stochastic volatility model widely used in financial risk management. Its popularity is due to the fact that it can reproduce comparatively well the market-observed volatility smile and that it provides a closed-form formula for the volatility. Given the risk-free interest rate r, the annual dividend yield q, and the forward F_t with expiry T is:

$$F_t = S_t e^{(r-q)(T-t)}$$

In the SABR stochastic volatility model, the forward F_t price follows the following stochastic differential equation:

$$\begin{aligned} dF_t &= \alpha_t (F_t)^{\beta} dW_t \\ d\alpha_t &= \nu \alpha_t dZ_t \\ E[dW_t dZ_t] &= \rho dt \end{aligned}$$

These parameters are described as follows:

- α_t is the instantaneous volatility of the forward F_t.
- ν is the volatility of instantaneous volatility α_t .
- W_t and Z_t are correlated Wiener process with correlation coefficient ρ

A variant of the Black–Scholes option pricing model, Black model [17], is often used together with SABR model. Under Black model, the forward F_t price follows the following stochastic differential equation:

$$dF_t = \sigma_B F_t dW_t$$

where σ_B is the volatility. We use $V_B(F,K,r,t,T;\sigma_B)$ to denote the Black pricing function. For European call option:

$$V_B(F,t,T,K,r;\sigma_B) = e^{-r(T-t)}[F\mathcal{N}(d3) - K\mathcal{N}(d4)]$$

For European put option:

$$V_B(F,t,T,K,r;\sigma_B) = e^{-r(T-t)}[K\mathcal{N}(-d4) - F\mathcal{N}(-d3)]$$

where $\mathcal{N}$ is the cumulative density function of standard normal distribution

$$d_3 = \frac{\ln(F/K) + \sigma^2/2(T-t)}{\sigma\sqrt{T-t}}, \;\; d_4 = d_3 - \sigma\sqrt{T-t}$$

Consider an option on the forward F with expiry T and strike K at time t. If we force the SABR model price of the option into the form of the Black model valuation formula. Then the SABR implied volatility, which is the value of the σ_B in Black's model that forces it to match the SABR price, is approximately given by:

$$\sigma_B(F,t,T,K;\alpha,\beta,\nu,\rho) \approx \frac{\alpha}{(FK)^{(1-\beta)/2}\left[1+\frac{(1-\beta)^2}{24}\log^2(F/K)+\frac{(1-\beta)^4}{1920}\log^4(F/K)+\cdots\right]}\cdot\frac{z}{x(z)}$$
$$\cdot\left\{1+\left[\frac{(1-\beta)^2}{24}\frac{\alpha^2}{(FK)^{1-\beta}}+\frac{1}{4}\frac{\rho\beta\nu\alpha}{(FK)^{(1-\beta)/2}}+\frac{2-3\rho^2}{24}\nu^2\right](T-t)+\cdots\right\}$$

where

$$z=\frac{\nu}{\alpha}(FK)^{(1-\beta)/2}\log(F/K),\ x(z)=\log\left\{\frac{\sqrt{1-2\rho z+z^2}+z-\rho}{1-\rho}\right\}$$

For the special case of at-the-money options, options struck at $K=F$, this formula reduces to

$$\sigma_{ATM}=\sigma_B(F,t,T,F;\alpha,\beta,\nu,\rho)$$
$$\approx\frac{\alpha}{F^{(1-\beta)}}\left\{1+\left[\frac{(1-\beta)^2}{24}\frac{\alpha^2}{F^{2-2\beta}}+\frac{1}{4}\frac{\rho\beta\alpha\nu}{F^{(1-\beta)}}+\frac{2-3\rho^2}{24}\nu^2\right](T-t)+\cdots\right\}$$

Therefore, the European option value under SABR model is given by:

$$V_{SABR}=V_B(F,t,T,K,r;\sigma_B(F,t,T,K;\alpha,\beta,\nu,\rho))$$

The $\sigma_B(F,t,T,K;\alpha,\beta,\nu,\rho)$ under SABR model depends on the forward F, the strike K, the time to expiry $T-t$, the initial SABR volatility α, the power of forward β, the volatility of volatility ν, and the correlation ρ. Interested readers can refer to [78] for the detailed derivation of the analytical approximation $\sigma_B(F,t,T,K;\alpha,\beta,\nu,\rho)$. We use $V_{SABR}(S,t,T,K,r,q;\alpha,\beta,\nu,\rho)$ to denote the option pricing formula under SABR model regardless of the call and put nature in the latter discussion.

2.2 Discrete Hedging Problem

In this thesis, we are dealing with real market option and underlying prices. We use $V^{mkt}(t,T,K)$ and $S(t)$ to denote the time t market price of the option with expiry T and strike K and the underlying price respectively. We propose to learn discrete data-driven hedging position for standard options, calls or puts, based on observations of the option market prices on the same underlying price at a set of trading times. The goal of the data-driven discrete option hedging in this thesis is to compute a hedging position δ in the underlying dynamically to minimize certain appropriate measurements of the hedging risk. We are interested in two different types of hedging risk:

- Discrete local hedging risk
- Discrete total hedging risk

The definition of the discrete local hedging risk and discrete total hedging risk are explained in the following subsections.

2.2.1 Discrete Local Hedging Risk

Let Δt denote the fixed time interval. Each observation of a market option price $V^{mkt}_{t,K,T}$ is uniquely associated with a triplet $\{t,T,K\}$, where t is the trading time of the option price, K is the strike, and expiry T. Denote:

$$\begin{aligned} \Delta V^{mkt}_{t,K,T} &= V_{mkt}(t+\Delta t,K,T) - V_{mkt}(t,K,T) \\ \Delta S_t &= S(t+\Delta t) - S(t) \end{aligned} \tag{2.2.1}$$

The discrete local hedging risk in the rebalancing interval Δt is:

$$\Delta S_t \delta_{t,T,K} - \Delta V^{mkt}_{t,K,T} \tag{2.2.2}$$

where $\delta_{t,T,K}$ is a hedging position.

The discrete local hedging risk can be understood as the following. We set up the following portfolio at time t:

- A short position on option $V^{mkt}(t,T,K)$
- Long $\delta_{t,T,K}$ shares of $S(t)$
- An amount in risk-free bank account $B(t)$

The bank account at time t is set to be:

$$B(t) = V^{mkt}(t,T,K) - S(t)\delta_{t,T,K}$$

We use $P_H(t,T,K)$ to denote the hedging portfolio value at t for hedging options with expiry T and strike K. The hedging portfolio value is:

$$P_H(t,T,K) = -V^{mkt}(t,T,K) + S(t)\delta_{t,T,K} + B(t) = 0$$

Let $dV^{mkt}_{t,T,K}$, dS_t, dB_t, and $dP^H_{t,T,K}$ denote the instantaneous change in the market option price, the underlying price, bank account and the hedging portfolio value respectively, the instantaneous hedging risk at time t is therefore:

$$dP^H_{t,T,K} = \ dS_t \delta_{t,T,K} - dV^{mkt}_{t,T,K} + dB_t \tag{2.2.3}$$

Note that dB_t is deterministic:

$$dB_t = rB(t)dt$$

Therefore, if we assume the risk-free interest rate is zero (or omit dB_t since it is deterministic), we have:

$$dP_t^H = dS_t\delta_{t,T,K} - dV_{t,T,K}^{mkt} \tag{2.2.4}$$

When the market is complete as what we assume in section 2.1.1 for the derivation of Black–Scholes formula, the randomness in this instantaneous hedging risk can theoretically be eliminated by continuously trading the underlying asset and set $\delta = \frac{dV_{t,T,K}^{mkt}}{dS_t}$. However in practice, option hedging needs to be implemented in an incomplete market, since hedging can only be done at discrete times. Besides additional risk, e.g., jump and volatility, cannot be eliminated by trading the underlying asset only [81, 66]. Lastly, in practice, $\frac{dV_{t,T,K}^{mkt}}{dS_t}$ is hard to be accurately computed due the pricing model parameters dependence on underlying asset. The pricing model parameters dependence will be explained in more details in section 2.3.2. Therefore, the hedging risk cannot be eliminated even instantaneously.

In practice, one actually care about the portfolio changes after a discrete time interval Δt. After the fixed interval Δt and assuming the interest rate is zero, we have:

$$\Delta P_H(t,T,K) = P_H(t+\Delta t,T,K) - P_H(t,T,K) = \Delta S_t\delta_{t,T,K} - \Delta V_{t,K,T}^{mkt} \tag{2.2.5}$$

The local hedging risk is therefore defined to be the one-step hedging error when we assume the risk-free interest risk is zero. The local discrete hedging risk (2.2.2) measures the changes in the hedging portfolio after a fixed time interval Δt when the hedging position is set to be $\delta_{t,T,K}$. As $\Delta t \to 0$, we also have local hedging risk converge to instantaneous hedging risk:

$$\Delta S_t\delta_{t,T,K} - \Delta V_{t,K,T}^{mkt} \to dS_t\delta_{t,T,K} - dV_{t,T,K}^{mkt}$$

2.2.2 Discrete Total Hedging Risk

In reality, one usually want to hedge until the expiries of the options, which requires rebalancing the hedging portfolio multiple times. Again, consider a hedging portfolio $P_H(t,T,K)$ which is composed of:

- A short position on option $V^{mkt}(t,T,K)$
- Long $\delta_{t,T,K}$ (hedging position) shares of $S(t)$
- An amount in a risk-free bank account $B(t)$

For notational simplicity since the T and K are fixed in following discussion, we denote:

$$V_t^{mkt} = V^{mkt}(t,T,K)\ , S_t = S(t)\ , \delta_t = \delta_{t,T,K}, B_t = B(t)\ , P_t^H = P_H(t,T,K)$$

The hedging portfolio is rebalanced at discrete times $\{t_0, t_1, \ldots, t_{N-1}\}$. Initially at t_0, we have

$$P_{t_0}^H = -V_{t_0}^{mkt} + \delta_{t_0}S_{t_0} + B_{t_0} = 0$$

And

$$B_{t_0} = V_{t_0}^{mkt} - \delta_{t_0} S_{t_0}$$

At each rebalancing time t_j, we update our hedging position by changing the share we hold from $\delta_{t_{j-1}}$ to δ_{t_j} at t_j, where any required cash is borrowed, and any excess cash is loaned. Assume $\Delta t = t_j - t_{j-1}$ is fixed and risk-free interest rate is zero. The bank account is updated by:

$$B_{t_j} = B_{t_{j-1}} - S_{t_j}(\delta_{t_j} - \delta_{t_{j-1}})$$

Let t_j^+ and t_j^- be the time immediately after and immediately before t_j. Assume that the performance is measured at the t_N^-, where $t_N = T$ is the expiry:

$$\begin{aligned} P_{t_N^-} &= B_{t_{N-1}} - V_{t_N}^{mkt} + S_{t_N}\delta_{t_{N-1}} \\ &= \sum_{j=0}^{N-1} \left\{ \left(S_{t_{j+1}} - S_{t_j}\right) \delta_{t_j} \right\} + V_{t_0}^{mkt} - V_{t_N}^{mkt} \\ &= \sum_{j=0}^{N-1} \left\{ \left(S_{t_{j+1}} - S_{t_j}\right) \delta_{t_j} - (V_{t_{j+1}}^{mkt} - V_{t_j}^{mkt}) \right\} \end{aligned} \tag{2.2.6}$$

Equation (2.2.6) is defined as the *discrete total hedging risk*. If we always set $\delta_t = \frac{dV_t^{mkt}}{dS_t}$ and let $\Delta t \to 0$ (we continuously rebalance the portfolio), then $P_{t_N^-} = 0$. In reality, even if we can set the hedging position to be the actual partial derivative of market option price with regards to underlying price $\delta_t = \frac{dV_t^{mkt}}{dS_t}$, we can only rebalance discretely due to the transaction cost. Thus, $P_{t_N^-}$ can take positive (profit) and negative value (loss). The *discrete total hedging risk* measures the hedging portfolio profit and loss at the expiry T.

Equation (2.2.6) can be written as:

$$\begin{aligned} P_{t_N^-}^H &= \sum_{j=0}^{N-1} \left\{ \left(S_{t_{j+1}} - S_{t_j}\right) \delta_{t_j} - (V_{t_{j+1}}^{mkt} - V_{t_j}^{mkt}) \right\} \\ &= \sum_{j=0}^{N-1} \left\{ \Delta S_{t_j} \delta_{t_j} - \Delta V_{t_j}^{mkt} \right\} \end{aligned}$$

with $\Delta V_{t_j}^{mkt}$ and ΔS_{t_j} given in (2.2.1). Plug in the notation for T and K, the discrete total hedging risk is:

$$P_H(T,T,K) = \sum_{j=0}^{N-1} \left\{ \Delta S_{t_j} \delta_{t_j,T,K} - \Delta V_{t_j,T,K}^{mkt} \right\} \tag{2.2.7}$$

By comparing equation (2.2.2) and (2.2.7), we can see that the *discrete total hedging risk* is summation of the *discrete local hedging risk* evaluated at discrete rebalancing time $\{t_0, t_1, \ldots, t_{N-1}\}$.

2.2.3 Interest Rate and Dividend

In the previous discussion, we assume zero interest rate when we define the *discrete total hedging risk* and the *discrete local hedging risk*. In the years of publication of the results presented in [118] and [119], the zero interest rate was assumed when measuring the hedging performance under discrete local hedging risk framework. The results presented [118] and [119] were evaluated from 2007 to 2015. During most of the time from 2007 and 2015, the interest rate is virtually zero. So the zero interest assumption is valid during most of the time. In [120], such assumption is relaxed and we include the effect of interest rate when we evaluate total hedging performance. We assume the interest rate is fixed as a constant during the life time of the hedging portfolio and denote it as r, the discrete total hedging risk is therefore:

$$P_H(T,T,K) = \sum_{j=0}^{N-1} \left\{ \left[e^{r(N-j-1)\Delta t} S_{t_{j+1}} - e^{r(N-j)\Delta t} S_{t_j} \right] \delta_{t_j,T,K} \right\} + e^{rN\Delta t} V^{mkt}_{t_0,T,K} - V^{mkt}_{T,T,K} \quad (2.2.8)$$

In this thesis, we are dealing with S&P 500 index option. For S&P 500 index, the closing index price of each day is already adjusted to capture corporate actions that affect market capitalization such as as additional share insurance, dividends and restructuring events such as mergers or spin-offs[99]. When we calibrating the option pricing models such as Black-Scholes model, an estimate of the dividends to be paid up until the option's expiration are needed. For dividend-paying indices, we assume that the security pays dividends continuously, according to a continuously-compounded annual dividend yield q.

For the calibration of different pricing models in this thesis, we use the option market data from OptionMetric [122] database. The OptionMetric [122] database provides us market option quotes for each trading day from 1996-01 to 2015-08-31. We use the zero curves on each trading day provided by the OptionMetric [122] database to extract the risk-free interest rate r. In addition, OptionMetric [122] database provides annual dividend yield q for S&P 500 index. The q is recorded daily for the S&P 500 index and supplied as the input for calibrating the option pricing models. Details of how OptionMetric compute the zero curves and the annual dividend yield q using market data can be found in [122].

2.3 Option Hedging Using the Sensitivity from Pricing Model

In this section, we discuss how we can compute the hedging position $\delta_{t,T,K}$ from the option pricing model, and the drawbacks of this approach. Specifically, we discuss the pricing model parameters dependence issue, which motivates our data-driven model, and provide several existing ways for correcting the model parameter dependence.

2.3.1 Classical Hedging Position From Pricing Model

In section 2.1, we have introduced several option pricing models with analytical pricing formula. Let $V(S,t,T,K,r,q;\theta)$ denote the arbitrary option pricing model. For Black-Scholes Model, θ is the volatility σ. For Heston model, $\theta = \{\upsilon, \kappa^*, \overline{\upsilon}^*, \eta, \rho\}$. For SABR model, $\theta = \{\alpha, \beta, \nu, \rho\}$. [1] We want to calibrate the pricing model to match the market option quotes.

On each trading time t, we have observed market option prices for different strikes $\{K_1, \ldots, K_{N_k}\}$ and different expiries $\{T_1, \ldots, T_{N_T}\}$, where N_k is the number of strikes and N_T is the number of expiries. Further assume the risk-free interest rate and annual dividend yield are given as r and q. The underlying asset price is S_t. The calibration can be done by:

$$\min_{\theta} \sum_{i=1}^{N_T} \sum_{j=1}^{N_K} \Big(V(S_t, t, T_i, K_j, r, q; \theta) - V^{mkt}(t, T_i, K_i) \Big)^2$$

For Black-Scholes model, there is only one parameter to be calibrated, so one can usually find that a constant volatility cannot match all market option quotes very well. Alternatively, one can match each market quotes exactly by:

$$V_{BS}(S_t, t, T_i, K_j, r, q; \sigma^{imp}_{t,T_i,K_j}) = V^{mkt}(t, T_i, K_j)$$

The volatility σ^{imp}_{t,T_i,K_j} that equals the Black–Scholes price with the market option price is called **Black-Scholes implied volatility**. (In section 2.1.3, we have introduced a term called SABR implied volatility which equals the SABR model price with the Black model price. The SABR implied volatility and Black-Scholes implied volatility are two different but related terminologies.) Note that in this way, for different options $V^{mkt}(t, T_i, K_j)$, different models are calibrated. One can often find that the implied volatility σ^{imp} tends to differ across different strikes and expiries.

Alternatively, for stochastic volatility models such as Heston and SABR, the calibration is usually done by matching the volatility smile instead of the entire volatility surface [81, 78, 90]. In other words, for a fixed expiry T_i, we solve

$$\min_{\theta} \sum_{j=1}^{N_K} \Big(V(S_t, t, T_i, K_i, r, q; \theta) - V^{mkt}(t, T_i, K_i) \Big)^2$$

Note that in this way, pricing models are calibrated separately for different expiries T_i on the same trading date t.

Given the option pricing model $V(S,t,T,K,r,q;\theta^*)$ with the calibrated θ^*, the hedging positive is commonly determined by:

$$\delta_{t,T,K} = \frac{\partial V(S,t,T,K,r,q;\theta^*)}{\partial S}$$

[1] Note that, Studies [90, 78, 77] shows that β can often be fixed a priori. In this case, $\theta = \{\alpha, \nu, \rho\}$.

Although a more complex option pricing model such as stochastic volatility models can match the market quotes better and account for the volatility smiles better, it does not necessarily lead to better hedging strategy. Several studies [53, 12, 160, 99] detail the hedging performances of option pricing models and show that, while explicitly modelling volatility smile substantially improves the pricing accuracy, it frequently decreases the hedging performance compared to the simpler Black-Scholes models. A recent empirical study [99] demonstrates that, for hedging S&P 500 index options, the simple Black-Scholes model with a constant volatility calibrated on each trading date can even outperform complex stochastic volatility models such as Heston model [81] and Heston-Nandi model [82]. The phenomenon where better pricing model leads to worse hedging performance is refer to as pricing/hedging conundrum in [99]. In the following section, we provide one of the potential explanation why such phenomenon exists.

2.3.2 Parameter Dependence on Underlying Asset

We notice that determining the hedging position from the partial derivative:

$$\delta_{t,T,K} = \frac{\partial V(S,t,T,K,r,q;\theta^*)}{\partial S}$$

of the calibrated pricing function inevitably leads to incomplete elimination of the underlying sensitivity, even in the context of instantaneous hedging. To see this, let us assume that a pricing model allows matching of the market price V^{mkt} at t exactly, i.e., θ^* is determined so that for arbitrary observed T and K in market.

$$V(S,t,T,K,r,q;\theta^*) = V^{mkt}(t,T,K) \tag{2.3.1}$$

In general, $\{t,T,K\}$ do not depend on S. Let us further assume $\{r,q\}$ do not depend on S. Unfortunately, the calibration process inevitably relates S and θ^*. Therefore, we have:

$$\frac{dV}{dS} = \frac{\partial V}{\partial S} + \frac{\partial V}{\partial \theta^*}\frac{\partial \theta^*}{\partial S}$$

Thus, unless one can ensure $\frac{\partial \theta^*}{\partial S} = 0$, we will have $dV - \frac{\partial V}{\partial S}\cdot dS \neq 0$. Several studies have demonstrate that pricing model parameters can be correlated with the underling asset price, so whenever S changes, the pricing model parameters should change too. As an example, negative correlation between Black-Scholes implied volatility and underlying asset price has been observed in [63, 20]. Besides, the calibration equation (2.3.1) says little about how the sensitivity $\frac{dV}{dS}$ matches $\frac{dV^{mkt}}{dS}$. Thus, even if we can compute the sensitivity $\frac{dV}{dS}$ exactly, we generally have:

$$dV^{mkt} - \frac{dV}{dS}\cdot dS \neq 0$$

Even with the perfect case where we can have a pricing model matching all observed market prices, which is often not achievable due to the calibration error, and the unrealistic assumption:

$$\frac{dV^{mkt}}{dS} = \frac{dV}{dS} = \frac{\partial V}{\partial S} + \frac{\partial V}{\partial \theta^*}\frac{\partial \theta^*}{\partial S} \tag{2.3.2}$$

We still have the pricing model parameters dependence issue to be addressed. Since it is difficult to account for parameter dependence $\frac{\partial \theta^*}{\partial S}$, $\frac{dV^{mkt}}{dS}$ can be hard to be captured accurately. The classical approach of using $\frac{\partial V}{\partial S}$ as the hedging position does not eliminate the instantaneous hedging risk.

A more complex model usually has more pricing model parameters to be calibrated. The more parameters there are, the more severe the parameters dependence issue possibly is. That may be one of the reason why many studies observe poorer hedging performance from more complex pricing model than simple Black-Scholes model in real market. Even with the simple Black-Scholes model, the Black–Scholes implied volatility still depends on the underlying asset. In the following section, we introduce several corrections under Black-Scholes framework for the parameter dependence. In addition, complex models like Heston model usually can not match the observed market option prices exactly. The calibration error can also potentially increase the hedging errors when the partial derivative is used as the hedging position.

2.3.3 Corrections Under Black-Scholes Framework

Although the issue of pricing model parameters dependence on underlying asset has been noted, e.g., in [38, 90], correcting for this risk exposure is not straightforward. Methods have been proposed to compensate for this missing exposure. Typically methods are based on analysis of some parametric models, e.g., LVF and SV; these methods minimize the variance of the hedging error [90, 3, 6, 7, 72]. In this section, we present several correction methods under Black-Scholes Framework. The discussion below are majorly drawn from [90, 77, 15].

A Data-Driven Correction For The Black–Scholes Delta Hedging

Equation (2.3.2) implies that the sensitivity of the option value function $\frac{\partial V}{\partial S}$ calibrated at the re-balancing time to satisfy the calibration equation (2.3.1) cannot completely hedge the sensitivity of the market option price to the underlying price since $\frac{\partial \theta^*}{\partial S}$ is not accounted for. If the hedging risk in each rebalancing period is the performance criteria, the information in the option market price change needs to be explicitly incorporated to determine a better hedging method which minimizes the hedge risk as measured by market option price change instead of the model option price change.

If sufficient market option and underlying price data exist, a potentially more effective approach is to learn the hedging strategy based on historical observations of both changes in the

market option price and the underlying price. This approach is completely different from the existing parametric underlying price model based hedging methods.

To the best of our knowledge, the study [90] is the first to determine the hedging strategy based on historical observations of the market option prices and underlying prices. This particularly form (2.3.4) is derived based on the following analysis using the Black–Scholes model. More specifically, assume we have calibrate the implied volatilities σ^{imp} to match all the market option quotes as what we have discussed in section 2.3.1. Let us again assume we are computing the hedging position with a fixed T and K at a specific trading time t, so for notational simplicity in the following discussion, we drop t, T, K in the subscript. Following (2.3.2), to determine the sensitivity $\frac{dV_{BS}}{dS}$ accurately, one needs to compute

$$\frac{\partial V_{\mathrm{BS}}}{\partial S} + \frac{\partial V_{\mathrm{BS}}}{\partial \sigma^{imp}} \frac{\partial \sigma^{imp}}{\partial S} \tag{2.3.3}$$

$\frac{\partial V_{\mathrm{BS}}}{\partial S}$ is the Black–Scholes delta with implied volatility and $\frac{\partial V_{\mathrm{BS}}}{\partial \sigma^{imp}}$ Black–Scholes vega with implied volatility. Both can be computed analytical. The undetermined part is $\frac{\partial \sigma^{imp}}{\partial S}$. In [90], the authors choose

$$\frac{\partial \sigma^{imp}}{\partial S} = \frac{a + b \cdot \frac{\partial V_{\mathrm{BS}}}{\partial S} + c \cdot (\frac{\partial V_{\mathrm{BS}}}{\partial S})^2}{S\sqrt{T-t}}$$

where a, b and c are the parameters to be fitted using market option data. Therefore, plug in the notation for t, T, K, the minimum variance delta is therefore:

$$\delta^{MV}_{t,T,K} = \delta^{BS}_{t,T,K} + \mathrm{vega}^{BS}_{t,T,K} \frac{a + b \cdot \delta^{BS}_{t,T,K} + c \cdot (\delta^{BS}_{t,T,K})^2}{S_t\sqrt{T-t}}, \tag{2.3.4}$$

where $\delta^{BS}_{t,T,K}$ and $\mathrm{vega}^{BS}_{t,T,K}$ are respectively the Black–Scholes delta and vega, using the implied volatility at time t for option with strike K and expiry T.

The model parameters a, b, c are determined using historical observations. Let M denote the number of historical data instances. Each data instance corresponds to a unique combination of $\{t, T, K\}$. For data instance i, with time t_i, strike K_i, and expiry T_i, we approximate the instantaneous changes in the market option price and underlying price by the **daily** changes of the market option prices and underlying prices:

$$dV^{mkt}_{t_i,T_i,K_i} \approx \Delta V^{mkt}_{t_i,T_i,K_i}, \;\; dS_{t_i} \approx \Delta S_{t_i}$$

Thus, we have the following minimization problem:

$$\min_{a,b,c} \sum_{i=1}^{M} \left(\Delta S_{t_i} \delta^{MV}_{t_i,T_i,K_i} - \Delta V^{mkt}_{t_i,T_i,K_i} \right)^2$$

A linear regression is performed to determine the model parameters a, b, c.

Consequently, the minimum variance hedging function δ^{MV} is computed based on regression

estimation, assuming the quadratic parametric model (2.3.4). It has been shown [90] that the corrective formula (2.3.4) can significantly improve the daily hedging performance.

Correction With LVF Model For The Black–Scholes Delta Hedging

As an improvement over the BS model, the local volatility function (LVF) model,

$$\frac{dS_t}{S_t} = (r-q)dt + \sigma(S_t,t)dW_t \tag{2.3.5}$$

has been considered, see e.g., [54, 47, 48], and LVF and its extensions remain widely popular in practice. Many methods have been proposed to calibrate a local volatility function $\sigma(S,t)$ from the traded market option prices, see e.g., [93, 4, 37]. In addition, Coleman et al. [38] discuss a relationship between the partial derivatives of calls and puts in the context of the LVF model, under which a call and put symmetry relation holds, see, e.g., [26, 27]. This relationship is found useful in correcting the dependence of the implied volatility on the underlying for Black–Scholes delta hedging.

In Theorem 2.3.1 below, we formalize this relation [38] to any call and put functions satisfying the call-put-symmetry. We note that, for this relationship to be useful in correcting for MV hedging in practice, hence the relevant price functions correspond to the market option prices, not model option values.

Theorem 2.3.1. *Let $Call(S,t,T,K,r,q)$ and $Put(S,t,T,K,r,q)$ be the call option price and put option price with underlying price S, strike K, expiry T, time t, interest rate r and dividend yield q. Assume further that there exists a unique implied volatility calibrating to $Call(S,t,T,K,r,q)$ and $Put(S,t,T,K,r,q)$ respectively and the call-put-symmetry below holds:*

$$Call(S,t,T,K,r,q) = Put(K,t,T,S,q,r). \tag{2.3.6}$$

Then

$$\frac{\partial \breve{\sigma}_c(S,t,T,K,r,q)}{\partial S} = \frac{\partial \breve{\sigma}_p(K,t,T,S,q,r)}{\partial S}. \tag{2.3.7}$$

where $\breve{\sigma}_c(\cdot)$ is the BS implied volatility calibrated to $C(\cdot)$ and $\breve{\sigma}_p(\cdot)$ is the BS implied volatility calibrated to $P(\cdot)$ respectively.

Proof. Let $C_{\mathrm{BS}}(\cdot)$ and $P_{\mathrm{BS}}(\cdot)$ denote the BS model option value functions. Put and call symmetry under the Black-Scholes model implies that

$$C_{\mathrm{BS}}(S,t,T,K,r,q;\sigma) = P_{\mathrm{BS}}(K,t,T,S,q,r;\sigma), \tag{2.3.8}$$

where σ is any constant volatility. Let $\breve{\sigma}_c(S,t,T,K,r,q)$ and $\breve{\sigma}_p(K,t,T,S,q,r)$ be the Black–Scholes implied volatilities calibrated to $Call(S,t,T,K,r,q)$ and $Put(K,t,T,S,q,r)$ Then

$$\begin{aligned} Call(S,t,T,K,r,q) &= C_{\mathrm{BS}}(S,t,T,K,r,q;\breve{\sigma}_c(S,t,T,K,r,q)) \\ Put(K,t,T,S,q,r) &= P_{\mathrm{BS}}(K,t,T,S,q,r;\breve{\sigma}_p(K,t,T,S,q,r)). \end{aligned} \tag{2.3.9}$$

From (2.3.6) and above, it follows

$$C_{\mathrm{BS}}(S,t,T,K,r,q;\breve{\sigma}_c(S,t,T,K,r,q)) = P_{\mathrm{BS}}(K,t,T,S,q,r;\breve{\sigma}_p(K,t,T,S,q,r)) \tag{2.3.10}$$

Using (2.3.8) with $\breve{\sigma}_c(\cdot)$,

$$C_{\mathrm{BS}}(S,t,T,K,r,q;\breve{\sigma}_c(S,t,T,K,r,q)) = P_{\mathrm{BS}}(K,S,T,q,r;\breve{\sigma}_c(S,t,T,K,r,q))$$

From above and (2.3.10), it follows

$$P_{\mathrm{BS}}(K,S,T,q,r,\breve{\sigma}_c(S,t,T,K,r,q)) = P_{\mathrm{BS}}(K,t,T,S,q,r;\breve{\sigma}_p(K,t,T,S,q,r))$$

Assuming that there is a unique implied volatility from the BS formula, we have

$$\breve{\sigma}_c(S,t,T,K,r,q) = \breve{\sigma}_p(K,t,T,S,q,r) \tag{2.3.11}$$

Taking derivative with respect to S, we have

$$\frac{\partial \breve{\sigma}_c(S,t,T,K,r,q)}{\partial S} = \frac{\partial \breve{\sigma}_p(K,t,T,S,q,r)}{\partial S},$$

i.e., (2.3.7) holds. This completes the proof. □

Assume that the market option prices satisfy put-call symmetry. Then the relevance of Theorem 2.3.1 in accounting for dependence of the implied volatility on the underlying can be appreciated as follows. On the left hand side of (2.3.7), the derivative is with respect to the underlying price (the first argument). On the right hand side, the derivative is with respect to the strike price (the second argument). When $q \approx r$ (as in the futures options market), $\frac{\partial \breve{\sigma}_p(K,t,T,S,q,r)}{\partial S}$ can be estimated from the observed implied volatility surface. In other words, for at-the-money with $K = S$ option, the rate of change in the implied volatility with respect to changes in the underlying price is equal to the slope of the volatility smile. Consequently the sensitivity of the implied volatility to the underlying can be estimated.

Hull and White [90] implement a corrective formula for minimum variance hedging based on (2.3.7), referred to as the LVF minimum variance hedging. Hull and White [90] further assume that the rate of change in the implied volatility with respect to changes in the underlying price is equal to the slope of the volatility smile for options which are not at-the-money. Therefore, the LVF correction is:

$$\delta^{LVF}_{t,T,K} = \delta^{BS}_{t,T,K} + \mathrm{vega}^{BS}_{t,T,K}\frac{\partial \sigma^{imp}}{\partial K} \tag{2.3.12}$$

where $\frac{\partial \sigma^{imp}}{\partial K}$ can be estimated from the a quadratic function fitting the volatility smile for each expiry. Other minimum variance delta hedge methods have also been proposed to correct practitioner Black–Scholes delta explicitly, see, e.g., [12, 45, 14, 125, 10]. Interested reader can refer to them for more details.

Correction With SABR model for the the Black–Scholes Delta Hedging

Recall that the SABR implied volatility, which is the value of the σ_B in Black's model that forces it to match the SABR price, is given in section 2.1.3. Based on SARR model, a correction formula for Black delta hedging is thus given by:

$$\frac{\partial V_B}{\partial F} + \frac{\partial V_B}{\partial \sigma_B}\frac{\partial \sigma_B}{\partial F} \tag{2.3.13}$$

Note that, Black delta can be converted to Black–Scholes delta. One can rewrite the formula to compute the sensitivity with regards to underlying asset price S instead of forward F by:

$$\left(\frac{\partial V_B}{\partial F} + \frac{\partial V_B}{\partial \sigma_B}\frac{\partial \sigma_B}{\partial F}\right)\frac{dF}{dS} \tag{2.3.14}$$

Given the risk-free interest rate r, the annual dividend yield q, and the forward F_t at time t with expiry T is:

$$F_t = S_t e^{(r-q)(T-t)}$$

Plug in the notation for t, T, K as what we did in previous sections, we have:

$$\delta^{SABR}_{t,T,K} = \left(\delta^{B}_{t,T,K} + \text{vega}^{B}_{t,T,K}\frac{\partial \sigma_B}{\partial F}\right)e^{(r-q)(T-t)} \tag{2.3.15}$$

where $\delta^{B}_{t,T,K}$ is the Black delta $\frac{\partial V_B}{\partial F}$ at time t for option with strike K and expiry T. $\text{vega}^{B}_{t,T,K}$ is the Black vega $\frac{\partial V_B}{\partial \sigma_B}$ at time t for option with strike K and expiry T. $\frac{\partial \sigma_B}{\partial F}$ is the shortened form of $\frac{\partial \sigma_B(F_t,t,T,K;\alpha,\beta,\nu,\rho)}{\partial F}$.

Although, SABR model can be used to correct the dependence of implied volatility on the forward price, the formula (2.3.15) omits the fact the initial SABR volatility α is correlated with the forward F. So whenever the forward F changes, the initial SABR volatility α changes. There is still unaccounted parameter dependence in equation (2.3.15).

Therefore, Bartlett correction [77, 15] was proposed to account for the dependence of α on F. The Bartlett correction is based on the following analysis. Recall that, in SABR model we assume:

$$\begin{aligned} dF_t &= \alpha_t (F_t)^{\beta} dW_t \\ d\alpha_t &= \nu \alpha_t dZ_t \\ E[dW_t dZ_t] &= \rho dt \end{aligned}$$

which can be rewritten as:

$$dF_t = \alpha_t (F_t)^\beta dW_t$$
$$d\alpha_t = \nu \alpha_t \left(\rho dW_t + \sqrt{1-\rho^2} dZ_t \right) = \frac{\rho \nu}{(F_t)^\beta} dF_t + \nu \alpha_t \sqrt{1-\rho^2} dZ_t$$

Therefore, one can readily find that:

$$E\left[\frac{d\alpha_t}{dF_t}\right] = \frac{\rho \nu}{(F_t)^\beta}$$

The formula for Bartlett correction is thus:

$$\delta^{Bartlett}_{t,T,K} = \left[\delta^{B}_{t,T,K} + \text{vega}^{B}_{t,T,K} \left(\frac{\partial \sigma_B}{\partial F} + \frac{\partial \sigma_B}{\partial \alpha} \frac{\rho \nu}{(F_t)^\beta} \right) \right] e^{(r-q)(T-t)} \tag{2.3.16}$$

with $\frac{\partial \sigma_B}{\partial \alpha}$ being the shortened form for $\frac{\partial \sigma_B(F_t,t,T,K;\alpha,\beta,\nu,\rho)}{\partial \alpha}$. The Bartlett correction improves over the SABR delta in two different aspects:

- Empirically, the Bartlett correction is less sensitive to the choice of β parameter. In practice, the β is often fixed instead of being calibrated together with α, ν, ρ in SABR model. Different choices of β can often all fit the market prices but different choices of β can often lead to different SABR delta $\delta^{SABR}_{t,T,K}$ position. On the other hand, the $\delta^{Bartlett}_{t,T,K}$ is consistent with different choices of β. This phenomenon is shown in Figure 2.1. We calibrate a SABR model for S&P 500 index option with trading time t being 2012-01-04 and expiry being 2012-12-31. We compute and compare delta position and we can see that for SABR delta, different choice of β can lead to significantly different hedging position while the Bartlett delta position is consistent.

- Empirically, the Bartlett correction provides better hedging strategy. As an example, one can observe significant hedging performance difference between $\delta^{Bartlett}_{t,T,K}$ and $\delta^{SABR}_{t,T,K}$ on S&P 500 index options. More specifically, we calibrate SABR model on each trading date from Jan 1st 2007 to August 31st 2015 as what we have discussed in section 2.1 with $\beta = 1$. And we compare the local hedging performance for all the traded options from Jan 1st 2007 to August 31st 2015. Following [90], we use the Gain ratio below as the evaluating criteria of the hedging performance measure:

$$\text{GAIN} = 1 - \frac{\sum_{i=1}^{m} \left(\Delta V^{mkt}_{t_i,T_i,K_i} - \delta_{t_i,T_i,K_i} \Delta S_{t_i} \right)^2}{\sum_{i=1}^{m} \left(\Delta V^{mkt}_{t_i,T_i,K_i} - \delta^{BS}_{t_i,T_i,K_i} \Delta S_{t_i} \right)^2}. \tag{2.3.17}$$

  where m is the total number of data instances to be evaluated. The $\delta^{BS}_{t_i,T_i,K_i}$ is the implied Black–Scholes delta, δ_{t_i,T_i,K_i} is the hedging position from the method under consideration,

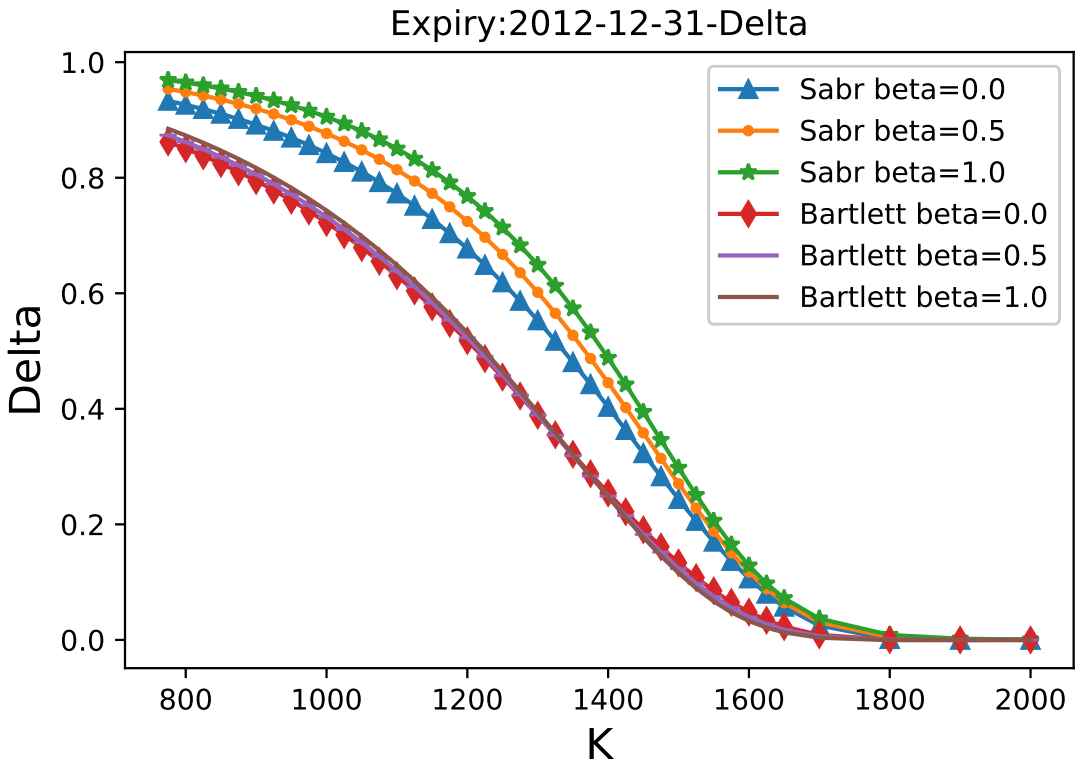


Figure 2.1: Comparison based on 2012-01-04 S&P500 index option

e.g., SABR delta $\delta^{SABR}_{t_i,T_i,K_i}$, MV delta $\delta^{MV}_{t_i,T_i,K_i}$, and Bartlett delta $\delta^{Bartlett}_{t_i,T_i,K_i}$. $\Delta V^{mkt}_{t_i,T_i,K_i}$ and ΔS_{t_i} are the changes in market option and underlying prices over a fixed time interval. Here we compare the **daily** local hedging risk with the daily changes of prices. The results are shown in Table 2.1: As we can see in Table 2.1, $\delta^{Bartlett}_{t,T,K}$ performs much better than

Method	SABR $\delta^{SABR}_{t,T,K}$	Bartlett $\delta^{Bartlett}_{t,T,K}$
Gain (%)	-4.2	27.1

Table 2.1: Daily Local Hedging Risk Comparison Between $\delta^{SABR}_{t,T,K}$ and $\delta^{Bartlett}_{t,T,K}$

$\delta^{SABR}_{t,T,K}$. What is more, $\delta^{SABR}_{t,T,K}$ actually performs worse than Black–Scholes delta $\delta^{BS}_{t,T,K}$ with implied volatility. The similar phenomenon has been observed in [99] where Heston and Heston-Nandi model performance worse than Black–Scholes delta $\delta^{BS}_{t,T,K}$ with implied volatility. **The difference of the hedging performance between $\delta^{Bartlett}_{t,T,K}$ and $\delta^{SABR}_{t,T,K}$ is an example to illustrate how the unaccounted pricing model parameter dependence on underlying asset affects the hedging performance.** Unfortunately, the pricing model parameter dependence often is not that easy to be accounted for as in SABR model with Bartlett correction.

Lastly, we introduce the corrective formula for minimum variance hedging based on SABR model implemented by Hull and White [90] . We use δ^{SV} to denote the minimum variance hedging based on SABR model from [90]. Let ΔF be a small changes in forward F, the δ^{SV} is given by:

$$\delta^{SV}_{t,T,K} = \frac{V_B(F_t+\Delta F,t,T,K,r;\sigma_B(F_t+\Delta F,t,T,K;\alpha,\beta,\nu,\rho)) - V_B(F_t,t,T,K,r;\sigma_B(,t,T,K;\alpha,\beta,\nu,\rho))}{\Delta F} \tag{2.3.18}$$

with V_B be the Black pricing formula and σ_B be the SABR implied volatility formula discussed in section 2.1.3.

2.4 Overview of the Data-Driven Hedging Models

In the previous sections, we discuss the issue of parameter dependence and present several correction methods under Black–Scholes framework. Although the corrective formula (2.3.4), (2.3.12), (2.3.15), and (2.3.16) adopt similar forms, the data-driven MV delta (2.3.4) is significantly different from other corrective formula for the fact that it is based on historical data while (2.3.12) ,(2.3.15), and (2.3.16) are based on pricing model calibrated to the spot options and underlying prices. Hull and White [90] indicate that MV delta $\delta^{MV}_{t,T,K}$ estimated based on historical data performs better than $\delta^{LVF}_{t,T,K}$ and $\delta^{SV}_{t,T,K}$ estimated based on spot data.

Our exploration on the data-driven models for hedging start roughly the same years as the work of [90] and we adopt the similar methodology as the MV delta where we learn a hedging position function from historical data. However, our explorations are motivated differently:

- The hedging position $\delta^{MV}_{t,T,K}$, $\delta^{LVF}_{t,T,K}$ and $\delta^{SV}_{t,T,K}$ proposed in [90] is to correct for the Black–Scholes implied volatility dependence on underlying asset. Specifically, Hull and White [90] demonstrate how one can estimate the $\frac{\partial \sigma^{imp}}{\partial S}$ empirically.
- Noticing the pricing parameters dependence, we try to learn a hedging position directly from the market data. We are not specifically trying to correct parameters dependence for certain pricing models. We are trying to avoid it by obtaining hedging position without a pricing model at all.

In addition, the data-driven models proposed in this thesis and MV delta are also different in the following ways:

- Hull and White [90] assume a quadratic form to account for the implied volatility dependence on the underlying asset within Black–Scholes framework. The hedging position from our proposed data-driven models is purely determined by market data with machine learning algorithms with no specific parametric form.
- The MV hedging model (2.3.4) focuses on instantaneous hedging analysis (2.2.4). For discrete hedging, particularly when re-balancing is done infrequently, e.g., weekly or monthly, (2.3.4) may no longer be suitable. Our proposed models, which are not based on computing sensitivity of option pricing functions with regards to the underlying asset, can potentially improve the hedging results when the hedging is done less frequently.
- Our proposed data-driven models GRU_{δ} and $\text{GRU}_{\text{TOTAL}}$ can naturally include feature selection and feature extraction which can potentially improve the hedging performance.
- Our proposed data-driven model $\text{GRU}_{\text{TOTAL}}$ is enhanced to deal with total hedging scenarios instead of focusing on reducing the instantaneous hedging risk as the MV hedging model (2.3.4).

In the following chapters, we will start to formally describe the proposed hedging models and present numerical hedging performance comparison on both synthetic data and real market data.

Chapter 3

Data-Driven Kernel Learning Framework for Local Hedging Risk

In this chapter, we start the discussion on our proposed data-driven local hedging model from kernel learning framework. The data-driven kernel hedging model can be summarized as the following: Assume we have M data instances. Each observation of a market option price $V^{mkt}_{t_i,K_i,T_i}$ is uniquely associated with a triplet $\{t_i, T_i, K_i\}$, where t_i is the trading time of the option price, K_i is the strike, and expiry T_i, $i = 1, \ldots, M$. The hedging position function is determined by the quadratic data-driven local risk minimization problem:

$$\min \frac{1}{2M} \sum_{i=1}^{M} \left(\Delta S_{t_i} \delta(\mathbf{x}_{t_i}^{T_i,K_i}) - \Delta V^{mkt}_{t_i,K_i,T_i} \right)^2$$

With ΔS_{t_i} and $\Delta V^{mkt}_{t_i,K_i,T_i}$ given in (2.2.1) for a fixed time interval Δt. The hedging position $\delta(\mathbf{x}_{t_i}^{T_i,K_i})$ is given by a data-driven hedging function $\mathrm{DKL}_{\mathrm{SPL}}(\mathbf{x}_{\mathbf{t}}^{\mathbf{T},\mathbf{K}}; \widehat{\boldsymbol{\alpha}})$:

$$\delta(\mathbf{x}_{t_i}^{T_i,K_i}) = \mathrm{DKL}_{\mathrm{SPL}}(\mathbf{x}_{\mathbf{t}}^{\mathbf{T},\mathbf{K}}; \widehat{\boldsymbol{\alpha}})$$

with $\widehat{\boldsymbol{\alpha}}$ being the parameters for the hedging functions and $\mathbf{x}_t^{T,K}$ being the input features for the hedging function. Please note that $\widehat{\boldsymbol{\alpha}}$ in this chapter is the parameter for the regularized kernel network to be learnt from data. The α in section 2.1.3 is the volatility for SABR. They are two unrelated terminologies.

From chapter 2, we have seen various challenges for hedging when the position is computed from the calibrated option value function. Since nonparametric option function estimation does not make specific assumptions and can potentially match option prices more accurately, it is not unreasonable to expect that this hedging challenge can potentially be addressed by reducing mis-specification using a data-driven approach to learn an option value function. Here we in this chapter we also discuss this approach and analyze its challenging for option hedging. Similar to discrete hedging under the parametric model, the hedging position can be computed by

learning a nonparametric option value model and then determining hedging position from the partial derivatives. Indeed this is the approach adopted in [92]. In this chapters, we indicates that the issue of pricing model parameters dependence still exists even if one estimate the pricing model using a machine learning model with no assumption on the dynamic of the underlying asset movement.

The organization of this chapter is as the following: We discuss how one can compute a hedging positions from partial derivatives of the optimal regularized kernel functions in section 3.1.1. The proposed data-driven kernel hedging learning method is presented in section 3.2. In section 3.3, we discuss how cross-validation can be done efficiently for both kernel pricing and kernel hedging model. In section 3.4, we present experiments using synthetic data to illustrate the drawbacks of determining hedging position from partial derivative of pricing function estimated using machine learning algorithms and the effectiveness of data-driven direct kernel hedging functions.

3.1 Regularized Kernel Pricing Model

Recognizing various challenges in the parametric financial modeling approach, nonparametric option pricing has also been studied. The nonparametric option value modeling approach has the distinctive advantage of not relying on specific assumptions about the underlying asset price dynamics. Hutchinson et al. [92] first propose a nonparametric data-driven approach to price and hedge European options using neural networks, radial basis functions, and projection pursuit regressions. Many other neural network methods for European option pricing have also been proposed, see, e.g., [154, 16, 73, 65, 106].

Although there are quite a few studies on nonparametric option pricing models, to our knowledge, there has been little research specifically focusing on discrete hedging using a nonparametric method. Even when the hedging problem is considered, e.g., [92], it is treated as a byproduct of obtaining a nonparametric pricing function: The hedging position is obtained from the partial derivative of the option value function. Hutchinson et al. [92] show that, based on hedging errors on some simulated paths, this indirect data-driven hedging approach can potentially be an effective alternative to the traditional parametric delta hedging methods.

In this section, we follow the methodology of [92] and learn a data-driven option pricing model with the regularized kernel network. The hedging position is then given by the partial derivative of the option pricing model with regards to the underlying asset. Our goal is to learn a nonlinear option pricing function $V(\mathbf{x};\widehat{\boldsymbol{\alpha}})$ using a regularized kernel method [59].

Assume that we are given a positive definite kernel similarity

$$\mathscr{K}(\mathbf{x},\mathbf{x}') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R},$$

which captures similarity between $\mathbf{x}$ and $\mathbf{x}'$ implicitly in a high dimensional feature space. Assume that $\mathscr{H}_K$ is the *Reproducing Kernel Hilbert Space* (RKHS) induced by the symmetric pos-

itive definite kernel function $\mathscr{K}(\mathbf{x},\mathbf{y})$ and $\|f\|_{\mathscr{K}}$ is the norm in RKHS. We have the following Representer Theorem [146]:

Theorem 3.1.1. *(The Representer Theorem) Let $\mathscr{X}$ be a nonempty set and k a positive definite real-valued kernel on $\mathscr{X}\times\mathscr{X}$ with the RKHS $\mathscr{H}_K$. Given the a training sample $\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_M\}$, a strictly monotonically increasing real valued function $g:[0,\infty)\to\mathbb{R}$, and an arbitrary empirical risk function $E:\mathscr{X}^M\to\mathbb{R}$, then for any $f^*\in\mathscr{H}_K$ satisfying*

$$f^* = \underset{f\in\mathscr{H}_K}{\operatorname{argmin}}\ E\left(f(\mathbf{x}_1),f(\mathbf{x}_2),\cdots,f(\mathbf{x}_M)\right)+g(\|f\|_{\mathscr{K}})$$

f^* admits a representation of the form:

$$f^*(\cdot)=\sum_{i=1}^{M}\widehat{\alpha}_i\mathscr{K}(\cdot,\mathbf{x}_i)$$

with $\widehat{\alpha}_i\in\mathbb{R}$ for all $1\le i\le M$

A regularized kernel regression problem can be formulated as

$$\min_{f\in\mathscr{H}_K}\left(\sum_{i=1}^{M}L(f(\mathbf{x}_i))+\lambda_P\|f\|_{\mathscr{K}}^2\right) \tag{3.1.1}$$

where $L(\cdot)$ is a loss function. The regularization parameter $\lambda_P>0$ can be determined based on cross validation.

Following the Representer Theorem, e.g., [146], a solution of (3.1.1) has the form

$$f(\mathbf{x})=\sum_{i=1}^{M}\widehat{\alpha}_i\mathscr{K}(\mathbf{x},\mathbf{x}_i) \tag{3.1.2}$$

and the regularization term is given by

$$\|f\|_{\mathscr{K}}^2=\sum_{i=1}^{M}\sum_{j=1}^{M}\widehat{\alpha}_i\widehat{\alpha}_j\mathscr{K}(\mathbf{x}_i,\mathbf{x}_j). \tag{3.1.3}$$

Assume that a set of M training points $\{(\mathbf{x}_{t_1}^{T_1,K_1},V_{t_1,K_1,T_1}^{mkt}),\cdots,(\mathbf{x}_{t_M}^{T_M,K_M},V_{t_M,K_M,T_M}^{mkt})\}$ are given, where $\mathbf{x}_t^{T,K}\in\mathbb{R}^{d_l}$ is the input feature for pricing the option with strike K and expiry T at time t and $V_{t,K,T}^{mkt}=V^{mkt}(t,T,K)$ is the market option price at time t with strike K and expiry T. Each data instance corresponds to a unique triplet $\{t,T,K\}$. We can estimate an option value function $V(\mathbf{x};\widehat{\boldsymbol{\alpha}})$ based on the regularized kernel estimation (3.1.1) with $\widehat{\boldsymbol{\alpha}}=\{\widehat{\alpha}_1,\ldots,\widehat{\alpha}_M\}$.

Using (3.1.2) and (3.1.3), assuming quadratic loss, the option pricing function:

$$V(\mathbf{x}_t^{T,K};\widehat{\alpha}) = \sum_{i=1}^{N} \widehat{\alpha}_i \mathscr{K}(\mathbf{x}_t^{T,K}, \mathbf{x}_{t_i}^{T_i,K_i})$$

can be computed by solving

$$\min_{\widehat{\alpha}} \left(\sum_{i=1}^{M} \left(V_{t_i,T_i,K_i}^{mkt} - \sum_{j=1}^{M} \widehat{\alpha}_j \mathscr{K}(\mathbf{x}_{t_j}^{T_j,K_j}, \mathbf{x}_{t_i}^{T_i,K_i}) \right)^2 + \lambda_P \sum_{i=1}^{M} \sum_{j=1}^{M} \widehat{\alpha}_i \widehat{\alpha}_j \mathscr{K}(\mathbf{x}_{t_j}^{T_j,K_j}, \mathbf{x}_{t_i}^{T_i,K_i}) \right) \tag{3.1.4}$$

For standard options, the universal RBF kernel

$$\mathscr{K}(\mathbf{x}, \tilde{\mathbf{x}}) = e^{-\frac{\|\mathbf{x}-\tilde{\mathbf{x}}\|_2^2}{2\rho^2}} \tag{3.1.5}$$

is a reasonable kernel choice, since the option value function is very smooth, and a suitable bandwidth ρ is typically problem dependent and can be determined using cross validation.

3.1.1 Indirect Hedging Positions From Kernel Pricing Functions

Assume that the one dimension of the attribute vector $\mathbf{x}_t^{T,K} \in \mathbb{R}^{d_l}$ corresponds to the moneyness S_t/K at time t and other dimensions are not related to underlying prices. Then the delta hedging function option with strike K and expiry T at trading time t is typically determined as:

$$\delta_{t,T,K}^{IKL} = \frac{\partial V(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})}{\partial S} = \sum_{i=1}^{M} \widehat{\alpha}_i \frac{\partial \mathscr{K}(\mathbf{x}_t^{T,K}, \mathbf{x}_{t_i}^{T_i,K_i})}{\partial S} = \sum_{i=1}^{M} \frac{\widehat{\alpha}_i}{K} \frac{\partial \mathscr{K}(\mathbf{x}_t^{T,K}, \mathbf{x}_{t_i}^{T_i,K_i})}{\partial S/K} \tag{3.1.6}$$

Hutchinson et al. [92] demonstrate that this nonparametric hedging approach, using the partial derivative of a nonparametric pricing function learned from historical market data, can be a useful alternative for option hedging.

However, using (3.1.6) as the hedging position similarly does not minimize variance of hedge risk in general and the challenge in accounting for parameter dependence on the underlying remains. We can see that from the following arguments. We made an unrealistic assumption that the estimated kernel function $V(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})$ matches the target market option price exactly, i.e.,

$$V(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}}) = V_{t,T,K}^{mkt}$$

and

$$\frac{dV}{dS}(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}}) = \frac{dV^{mkt}}{dS}$$

Then we have:

$$\frac{dV}{dS}(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}}) = \sum_{i=1}^{M} \widehat{\alpha}_i \frac{\partial \mathscr{K}(\mathbf{x}_t^{T,K}, \mathbf{x}_{t_i}^{T_i,K_i})}{\partial S} + \sum_{i=1}^{M} \frac{\partial \widehat{\alpha}_i}{\partial S} \mathscr{K}(\mathbf{x}, \mathbf{x}_i) = \frac{dV^{mkt}}{dS}$$

Hence in general

$$\frac{\partial \widehat{\alpha}_i}{\partial S} \neq 0$$

unless

$$\frac{\partial V}{\partial S}(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}}) = \frac{dV^{mkt}}{dS}. \tag{3.1.7}$$

However there is no reason that a solution of the regression problem (3.1.4) should satisfy (3.1.7). Consequently it is similarly difficult to account for all dependence from $\widehat{\boldsymbol{\alpha}}$ on the underlying asset, even infinitesimally, in the estimated kernel model.

Furthermore, error magnification can happen by deriving the hedging position from a estimated kernel function. In general, the estimated kernel pricing function $V(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})$ can not match the all the target market option prices exactly and the estimated kernel pricing function inevitably will have prediction error when used to predict the price of unobserved testing data instances. The calibration error and prediction can potentially increase the hedging error when using the $\frac{\partial V}{\partial S}(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})$ as the hedging position.

3.2 Regularized Kernel Hedging Model

Let us again assume that a set of M training points

$$\left\{(\mathbf{x}_{t_1}^{T_1,K_1}, \Delta V_{t_1,K_1,T_1}^{mkt}, \Delta S_{t_1}), \cdots, (\mathbf{x}_{t_M}^{T_M,K_M}, \Delta V_{t_M,K_M,T_M}^{mkt}, \Delta S_{t_M})\right\}$$

are given, where $\mathbf{x}_t^{T,K} \in \mathbb{R}^{d_l}$ is the input feature for **hedging** the option with strike K and expiry T at time t. The $\Delta V_{t,K,T}^{mkt}$ is the change of market option price at time t with strike K and expiry T over a fixed time intervel Δt, e.g., daily, weekly, or monthly. The ΔS_t is the change of underling price at time t over a fixed time intervel Δt over the same Δt. $\Delta V_{t,K,T}^{mkt}$ and ΔS_t are given by equation (2.2.1). Again, each data instance corresponds to a unique triplet $\{t,T,K\}$.

We can estimate an option hedging function $\delta(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})$ based on the regularized kernel network with $\widehat{\boldsymbol{\alpha}} = \{\widehat{\alpha}_1, \ldots, \widehat{\alpha}_M\}$. The empirical loss function is chosen to correspond to the square of discrete local hedging risk in section 2.2, i.e.,

$$L\left(\delta(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})\right) = \left(\Delta V_{t,K,T}^{mkt} - \Delta S_t \delta(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})\right)^2. \tag{3.2.1}$$

The hedging position function $\delta(\mathbf{x}_t^{T,K};\widehat{\boldsymbol{\alpha}})$ can be estimated from the regularized optimization

below:

$$\min_{\delta \in \mathscr{H}_K} \left\{ \sum_{i=1}^{M} L\left(\delta(\mathbf{x}_{t_i}^{T_i,K_i}; \widehat{\boldsymbol{\alpha}})\right)^2 + \lambda_P \|\delta\|_{\mathscr{K}}^2 \right\} \tag{3.2.2}$$

Note that the Representer Theorem still holds for (3.2.2). Using the Representer Theorem (3.1.2) and (3.1.3), (3.2.3) can be computed by solving the following convex quadratic minimization,

$$\min_{\widehat{\boldsymbol{\alpha}}} \left(\sum_{i=1}^{M} \left(\Delta V_{t_i,T_i,K_i}^{mkt} - \Delta S_{t_i} \sum_{j=1}^{M} \widehat{\alpha}_j \mathscr{K}(\mathbf{x}_{t_j}^{T_j,K_j}, \mathbf{x}_{t_i}^{T_i,K_i}) \right)^2 + \lambda_P \sum_{i=1}^{M} \sum_{j=1}^{M} \widehat{\alpha}_i \widehat{\alpha}_j \mathscr{K}(\mathbf{x}_{t_j}^{T_j,K_j}, \mathbf{x}_{t_i}^{T_i,K_i}) \right) \tag{3.2.3}$$

The loss function for the hedging function in the proposed formulation (3.2.2) directly corresponds to the sum of square of local hedging risk for a discrete Δt-time period. In addition the hedge function is the solution of the optimization problem and there is no potential error magnification through partial derivative computation. We avoid the pricing model parameters dependence on underlying asset by not computing a prcing model at all.

Since at the expiry the delta of the payoff function is a discontinuous step function, the delta hedging function of an option changes quickly as the underlying changes near the expiry. Consequently we choose to use a spline kernel function [144] for the hedging function estimation. The explicit expression for the spline kernel with order O_d for the one-dimensional case is :

$$\mathscr{K}(x,y) = \sum_{r=0}^{O_d} \frac{\binom{O_d}{r}}{2O_d - r + 1} min(x,y)^{2O_d - r + 1} |x-y|^r + \sum_{r=0}^{O_d} x^r y^r$$

For multidimensional data, the spline kernel is the product of one-dimensional spline kernel functions with regard to each dimension. Interested readers can referred to [144] for more details. The major benefit of using the spline kernel over the gaussian kernel is that spline kernel does not have a hyperparameter to be tuned while gaussian kernel has a bandwidth parameter to be tuned by cross-validation which is often costly. In this thesis, we majorly use spline kernel and, in section 3.4, we will show that spline kernel performs better or equally well when compared with gaussian kernel. In the latter discussion, we denote $\delta_{t,T,K}^{DKL} = \delta(\mathbf{x}_t^{T,K}; \widehat{\boldsymbol{\alpha}})$ as the hedging position form the directly kernel hedging model.

3.3 Cross Validation

For our data-driven approach, we need to select an appropriate penalty λ_P to control the model complexity. Cross-validation (CV) is a commonly used method for the performance estimation and model selection for the learning algorithms. For example, the Leave-One-Out Cross-Validation (LOOCV) computes the output for each data instance using parameters trained on the remaining data instances. For the regularized kernel methods, we can compute the CV error efficiently without retraining the model in each CV round.

Recall that, for the regularized network or pricing model, the minimization problem is

$$\min_{\widehat{\boldsymbol{\alpha}}} \left(\sum_{i=1}^{M} \left(V^{mkt}_{t_i,T_i,K_i} - \sum_{j=1}^{M} \widehat{\alpha}_j \mathscr{K}(\mathbf{x}_{t_j}^{T_j,K_j}, \mathbf{x}_{t_i}^{T_i,K_i}) \right)^2 + \lambda_P \sum_{i=1}^{M} \sum_{j=1}^{M} \widehat{\alpha}_i \widehat{\alpha}_j \mathscr{K}(\mathbf{x}_{t_j}^{T_j,K_j}, \mathbf{x}_{t_i}^{T_i,K_i}) \right)$$

Let $\mathbb{K}$ be the kernel matrix with $\mathbb{K}_{ij} = \mathscr{K}(\mathbf{x}_i, \mathbf{x}_j)$. (Note $\mathbb{K}$ here denotes the kernel matrix which should not be confused with the strike K.) Problem (3.1.1) can be rewritten in matrix form:

$$\min_{\widehat{\boldsymbol{\alpha}} \in \mathbb{R}^M} (\mathbb{K}\widehat{\boldsymbol{\alpha}} - \mathbb{V})^T (\mathbb{K}\widehat{\boldsymbol{\alpha}} - \mathbb{V}) + \lambda_P \widehat{\boldsymbol{\alpha}}^T \mathbb{K} \widehat{\boldsymbol{\alpha}} \tag{3.3.1}$$

with

$$\mathbb{V} = \{V^{mkt}_{t_1,T_1,K_1}, \ldots, V^{mkt}_{t_M,T_M,K_M}\} \quad , \quad \widehat{\boldsymbol{\alpha}} = \{\widehat{\alpha}_1, \ldots, \widehat{\alpha}_M\}$$

The solution to (3.3.1) is:

$$\widehat{\boldsymbol{\alpha}} = (\mathbb{K} + \lambda_P \boldsymbol{I})^{-1} \mathbb{V}$$

Given the eigen-decomposition $\mathbb{K} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, we can easily see that:

$$(\mathbb{K} + \lambda_P \boldsymbol{I})^{-1} = \boldsymbol{Q}(\boldsymbol{\Lambda} + \lambda_P \boldsymbol{I})^{-1} \boldsymbol{Q}^T \tag{3.3.2}$$

Because $(\boldsymbol{\Lambda} + \lambda_P I)$ is a diagonal matrix, given the eigen-decomposition $\mathbb{K} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, we can get different solutions to (3.3.1) as λ_P varies in $O(M^2)$.

Let $V(\mathbf{x}_{t_j}^{T_j,K_j}; \widehat{\boldsymbol{\alpha}})$ be the output for data instance j when regularized kernel methods (3.1.4) is trained on all training data instances. Let $V^l(\mathbf{x}_{t_j}^{T_j,K_j}; \widehat{\boldsymbol{\alpha}}^l)$ be the output for data instance j when regularized kernel methods (3.1.4) is trained on all training data instances except $\mathbf{x}_{t_l}^{T_l,K_l}$. Let $\mathbb{V}^l = \{\mathbb{V}^l_1, \mathbb{V}^l_2, \cdots, \mathbb{V}^l_M\}$ be the vector where $\mathbb{V}^l_j = V^{mkt}_{t_j,T_j,K_j}$ for $j \neq l$ and $\mathbb{V}^l_l = V^l(\mathbf{x}_{t_l}^{T_l,K_l}; \widehat{\boldsymbol{\alpha}}^l)$. Since $V^l(\cdot; \widehat{\boldsymbol{\alpha}}^l)$ is the model trained on all example except $\mathbf{x}_{t_l}^{T_l,K_l}$, it is easy to see that $V^l(\cdot; \widehat{\boldsymbol{\alpha}}^l)$ minimizes

$$\min_{\widehat{\boldsymbol{\alpha}} \in \mathbb{R}^M} (\mathbb{K}\widehat{\boldsymbol{\alpha}} - \mathbb{V}^l)^T (\mathbb{K}\widehat{\boldsymbol{\alpha}} - \mathbb{V}^l) + \lambda_P \widehat{\boldsymbol{\alpha}}^T \mathbb{K} \widehat{\boldsymbol{\alpha}} \tag{3.3.3}$$

The solution to (3.3.3) is:

$$\widehat{\boldsymbol{\alpha}}^l = (\mathbb{K} + \lambda_P \boldsymbol{I})^{-1} \mathbb{V}^l$$

Let $\boldsymbol{B} = \mathbb{K}(\mathbb{K} + \lambda_P \boldsymbol{I})^{-1}$, therefore, we have

$$\boldsymbol{B}\mathbb{V} = \{V(\mathbf{x}_{t_1}^{T_1,K_1}; \widehat{\boldsymbol{\alpha}}), \ldots, V(\mathbf{x}_{t_M}^{T_M,K_M}; \widehat{\boldsymbol{\alpha}})\}$$

We can easily show that:

$$V^l(\mathbf{x}_t^{T,K}; \widehat{\boldsymbol{\alpha}}^l) - V(\mathbf{x}_t^{T,K}; \widehat{\boldsymbol{\alpha}}) = \sum_{i=1}^{M} B_{li}(\mathbb{V}^l_i - \mathbb{V}_i) = B_{ll}(V^l(\mathbf{x}_{t_l}^{T_l,K_l}; \widehat{\boldsymbol{\alpha}}^l) - \mathbb{V}_l) = B_{ll}(V^l(\mathbf{x}_{t_l}^{T_l,K_l}; \widehat{\boldsymbol{\alpha}}^l) - V^{mkt}_{t_l,T_l,K_l})$$

Thus,

$$V^l(\mathbf{x}_{t_l}^{T_l,K_l};\widehat{\boldsymbol{\alpha}}^l) = \frac{V(\mathbf{x}_{t_l}^{T_l,K_l};\widehat{\boldsymbol{\alpha}}^l) - B_{ll}V_{t_l,T_l,K_l}^{mkt}}{1-B_{ll}} \tag{3.3.4}$$

Therefore, we can get the $V^l(\mathbf{x}_{t_l}^{T_l,K_l};\widehat{\boldsymbol{\alpha}}^l)$ without actually retraining the model. The leave-one-out estimations for all data instance are:

$$\mathbb{V}_{loo} = (I - B_{LL})^{-1}(B\mathbb{V} - B_{LL}\mathbb{V}) \tag{3.3.5}$$

and the leave-one-out errors for all data instance are:

$$\mathbb{V} - \mathbb{V}_{loo} = \mathbb{V} - (\boldsymbol{I} - B_{LL})^{-1}(\boldsymbol{B}\mathbb{V} - \boldsymbol{B}_{LL}\mathbb{V}) = (I - \boldsymbol{B}_{LL})^{-1}(\mathbb{V} - \boldsymbol{B}\mathbb{V}) \tag{3.3.6}$$

where B_{LL} is a diagonal matrix with $B_{ii},\ \ i = 1,\ldots,M$ on its diagonal.

We can further simplify the expression by noting the fact that:

$$\begin{aligned}
\boldsymbol{B} &= \mathbb{K}(\mathbb{K} + \lambda_P\boldsymbol{I})^{-1} \\
&= \boldsymbol{Q}\boldsymbol{\Lambda}(\boldsymbol{\Lambda} + \lambda_P\boldsymbol{I})^{-1}\boldsymbol{Q}^T \\
&= \boldsymbol{Q}(\boldsymbol{\Lambda} + \lambda_P\boldsymbol{I} - \lambda_P\boldsymbol{I})(\boldsymbol{\Lambda} + \lambda_P\boldsymbol{I})^{-1}\boldsymbol{Q}^T \\
&= \boldsymbol{I} - \lambda_P Q(\boldsymbol{\Lambda} + \lambda_P\boldsymbol{I})^{-1}\boldsymbol{Q}^T = \boldsymbol{I} - \lambda_P(\mathbb{K} + \lambda_P\boldsymbol{I})^{-1}
\end{aligned}$$

Therefore, we can get:

$$(\boldsymbol{I} - \boldsymbol{B}_{LL}) = diag(\boldsymbol{I} - \boldsymbol{B}) = \lambda_P\, diag((\mathbb{K} + \lambda_P\boldsymbol{I})^{-1})$$

$$\mathbb{V} - B\mathbb{V} = \lambda_P(\mathbb{K} + \lambda_P\boldsymbol{I})^{-1}\mathbb{V} = \lambda_P\widehat{\boldsymbol{\alpha}}$$

Finally, we can get:

$$\mathbb{V} - \mathbb{V}_{loo} = diag((\mathbb{K} + \lambda_P\boldsymbol{I})^{-1})^{-1}\widehat{\boldsymbol{\alpha}} \tag{3.3.7}$$

Given the eigen-decomposition $\mathbb{K} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, we can compute $\widehat{\boldsymbol{\alpha}}$ as λ_P varies in $O(M^2)$. Similarly, given the eigen-decomposition $\mathbb{K} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, we can compute the diagonal of $(\mathbb{K} + \lambda_P\boldsymbol{I})^{-1}$ in $O(M^2)$. Then using (3.3.7), the LOOCV errors can be computed in $O(M^2)$. It can been further shown [123] that, given the eigen-decomposition $\mathbb{K} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^T$, the computational complexity for the n-fold cross-validation (nFCV) is $O(M^3/n)$. Interested readers can referred to [123, 146] for more details.

Let $\Delta\mathbb{V} = \{\Delta V_{t_1,T_1,K_1}^{mkt},\cdots,\Delta V_{t_M,T_M,K_M}^{mkt}\}$ and let $\boldsymbol{D}$ be a diagonal matrix with $D_{ii} = \Delta S_{t_i}$. $i = 1,\cdots,M$. on its diagonal. Similarly, we can rewrite the minimization problem (3.2.3) in matrix form:

$$\min_{\widehat{\boldsymbol{\alpha}}\in\mathbb{R}^m}\ (\boldsymbol{D}\mathbb{K}\widehat{\boldsymbol{\alpha}} - \Delta\mathbb{V})^T(\boldsymbol{D}\mathbb{K}\widehat{\boldsymbol{\alpha}} - \Delta\mathbb{V}) + \lambda_P\widehat{\boldsymbol{\alpha}}^T\mathbb{K}\widehat{\boldsymbol{\alpha}} \tag{3.3.8}$$

Let $\widetilde{\mathbb{K}} = \boldsymbol{D}\mathbb{K}$, the solution to (3.3.8) can be obtained by:

$$\widehat{\boldsymbol{\alpha}} = (\widetilde{\mathbb{K}}^T\widetilde{\mathbb{K}} + \lambda_P\mathbb{K})^{-1}\widetilde{\mathbb{K}}^T\Delta\mathbb{V}$$

Unfortunately, we can not directly use the fast cross validation algorithms from [123, 146] for the minimization problem (3.3.8). Because $(\widetilde{\boldsymbol{K}}^T\widetilde{\boldsymbol{K}}+\lambda_P\mathbb{K})^{-1}$ can not be easily inverted in $O(M^2)$ as λ_P varies.

However if we change the penalty term for (3.3.8) from $\lambda_P\widehat{\boldsymbol{\alpha}}^T\mathbb{K}\widehat{\boldsymbol{\alpha}}$ to $\lambda_P\widehat{\boldsymbol{\alpha}}^T\widehat{\boldsymbol{\alpha}}$, the minimization problem becomes:

$$\min_{\widehat{\boldsymbol{\alpha}}\in\mathbb{R}^m}(\boldsymbol{D}\mathbb{K}\widehat{\boldsymbol{\alpha}}-\Delta\check{\boldsymbol{V}})^T(\boldsymbol{D}\mathbb{K}\widehat{\boldsymbol{\alpha}}-\Delta\check{\boldsymbol{V}})+\lambda_P\widehat{\boldsymbol{\alpha}}^T\widehat{\boldsymbol{\alpha}} \tag{3.3.9}$$

The solution to (3.3.9) can be obtained by:

$$\widehat{\boldsymbol{\alpha}}=(\widetilde{\mathbb{K}}^T\widetilde{\mathbb{K}}+\lambda_P\boldsymbol{I})^{-1}\widetilde{\mathbb{K}}^T\Delta\mathbb{V}$$

Utilizing the ideas from [123, 146], we can similarly show that, given the singular value decomposition $\widetilde{\boldsymbol{K}}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, for the problem (3.3.9), the computational complexity for the LOOCV is still $O(M^2)$ and the computational complexity for the nFCV is still $O(M^3/n)$. In practice, changing the penalty term from $\lambda_P\widehat{\boldsymbol{\alpha}}^T\mathbb{K}\widehat{\boldsymbol{\alpha}}$ to $\lambda_P\widehat{\boldsymbol{\alpha}}^T\widehat{\boldsymbol{\alpha}}$ will not affect the actual performance too much. Therefore, in order to improve the computation efficiency for the direct data driven approach, we are solving the problem (3.3.9).

3.4 Comparison using synthetic data

Following the S&P 500 market option specifications in [91], we first synthetically generate option data assuming that the underlying price follows a Heston model [81]. We compare hedging effectiveness of direct hedging function learning (3.2.2) and indirect hedging function estimation (3.1.4). In addition, we compare their performance to that of using analytic delta under the Heston model, which can be regarded as a best case benchmark, at least for daily hedging. Note that there is no model mis-specification for the underlying price in this synthetic case.

The experiments in the this section is to demonstrate the following aspects:

- Computing the hedging position as the partial derivative of an nonparametric option pricing model with regards to underlying asset still suffers from the issue of parameter dependence on underlying asset.
- The hedging position directly learnt from data as in (3.2.3) can be very close to the best case benchmark in daily hedging.
- The hedging position directly learnt from data using spline kernel performs better when compared with gaussian kernel.

Since the loss function is quadratic, solutions to (3.2.2) and (3.1.4) can be easily computed from linear equation solvers. In addition we also compare hedging performance using a RBF

kernel (3.1.5) versus a spline kernel. Specifically, using the generated synthetic data, we compare here hedging performance of the following hedging computation methods:

- $\delta^{BS}_{t,T,K}$: implied volatility BS delta
- $\delta^{Heston}_{t,T,K}$: analytical Heston delta
- DKL_{SPL} : direct learning a spline kernel hedging function based on (3.2.2)
- DKL_{RBF} : directly learning a RBF kernel hedging function directly based on (3.2.2)
- IKL_{SPL}: determining hedging position indirectly as the partial derivative (3.1.6) of the option value function estimated from (3.1.4) using a spline kernel
- IKL_{RBF}: determining hedging position indirectly as the partial derivative (3.1.6) of the option value function estimated from (3.1.4) using a RBF kernel

Training data consists of simulated daily underlying price S_t for two years[1], $t = 1, \cdots, 2 \times 252$, assuming a risk-neutral Heston model below:

$$\begin{aligned} dS &= (r-q)Sdt + \sqrt{\upsilon} S d\hat{W} \\ d\upsilon &= \kappa^*(\overline{\upsilon}^* - \upsilon)dt + \eta\sqrt{\upsilon} d\hat{Z} \\ E[d\hat{Z}d\hat{W}] &= \rho dt \end{aligned}$$

In addition, a vector of simulated (traded) market call option prices are generated, on each day t, with different strikes and time to expiry, following the CBOE specifications of stock options described in [91]. The option prices V^{mkt} are computed using the analytical option formula (2.1.10) under the Heston model [81] in section 2.1.2 , using the the parameters from [12], which are given in Table 3.1. In other words, we assume in this section:

$$V^{mkt}(t,T,K) = V_{Heston}(S_t, t, T, K, r, q; \upsilon, \kappa^*, \overline{\upsilon}^*, \eta, \rho)$$

Please note that the heston delta position under this synthetic scenarios does not have the parameter dependence issue, since for the synthetic case all heston parameter is fixed, and can be used as the benchmark.

$$\delta^{Heston}_{t,T,K} = \frac{\partial V_{Heston}(S_t, t, T, K, r, q; \upsilon, \kappa^*, \overline{\upsilon}^*, \eta, \rho)}{\partial S} = \frac{dV^{mkt}}{dS}$$

The Black-Scholes delta position from implied volatility still has the the issue of implied volatility depending on underlying asset:

$$\delta^{BS}_{t,T,K} = \frac{\partial V_{BS}(S_t, t, T, K, r, q; \sigma^{imp}_{t,T,K})}{\partial S} \neq \frac{dV^{mkt}}{dS}$$

[1] We assume that there are 252 trading days in a year.

where $\sigma^{imp}_{t,T,K}$ is the volatility that equals Black–Scholes price and Heston price

$$V_{BS}(S_t,t,T,K,r,q;\sigma^{imp}_{t,T,K}) = V_{Heston}(S_t,t,T,K,r,q;\upsilon,\kappa^*,\overline{\upsilon}^*,\eta,\rho)$$

r	q	$\overline{\upsilon}^*$	κ^*	η	ρ	S_0	υ_0
0.02	0.0	0.04	1.15	0.39	-0.64	100	0.04

Table 3.1: Parameters for the Heston Model

Testing data consists of 100 daily underlying price paths and corresponding option prices, spanning a six month period. The Heston parameter is still specified as in Table 3.1. We report average performance measures over 10 random training-test-data sets generated as described. [2]

We train a regularized option price kernel model for the indirect hedging IKL_{SPL} and IKL_{RBF}. For IKL_{SPL} and IKL_{RBF}, the hedging position is the partial derivative of the estimated pricing function (3.1.6), as in [92]. We note that the simulation hedging analysis in [92] considers only the Black–Scholes model and the results in [92] indicate that a lower hedging error can be achieved on a subset of simulated paths.

Following a common practice, we use the standard deviation of the pairwise Euclidean distance of the training data as the bandwidth ρ for the RBF kernel. The regularization parameter λ_P is however selected using a 5-fold cross-validation. We also consider weekly hedging and monthly hedging, which correspond to hedging over a 5-business-days period and 20-business-days period respectively.

To investigate impact of the feature choice, we evaluate hedging performance using

- Feature Set #1 $= \{\text{MONEYNESS } (S_t/K),\ \text{TIME-TO-EXPIRY } (T-t)\}$.
- Feature Set #2 $= \{\text{MONEYNESS } (S_t/K),\ \text{TIME-TO-EXPIRY } (T-t),\ \ \delta^{BS}_{t,T,K}\}$.

In the second feature set, the Black–Scholes delta $\delta^{BS}_{t,T,K}$ using the implied volatility is used as an additional feature in determining the hedging position.

Let the number of data instances to be evaluated be m. We have the profit and loss (2.2.5) of the local hedging portfolio in section 2.2.1 over the fixed time interval Δt for each data instance, which corresponds to a unique triplet $\{t,T,K\}$:

$$\Delta P_H(t,T,K) = \Delta S_t \delta_{t,T,K} - \Delta V^{mkt}_{t,K,T}$$

where $\delta_{t,T,K}$ is the hedging position from different approaches, e.g., Black–Scholes delta $\delta^{BS}_{t,T,K}$, Heston delta $\delta^{Heston}_{t,T,K}$, direct data-driven hedging position $\delta^{DKL}_{t,T,K}$, and indirect data-driven hedging position$\delta^{IKL}_{t,T,K}$. We evaluate the performance in four different ways:

[2]Following CBOE option specification rules, the size of a training or testing data set can vary slightly for each simulation run.

1. Gain (2.3.17) over Black-Scholes $\delta^{BS}_{t,T,K}$ as in [90].
2. The mean absolute value of $\Delta P_H(t,T,K)$

$$\mathbf{E}(|\Delta V^{mkt} - \Delta S\delta|) = \frac{1}{m}\sum_{i=1}^{m} |\Delta P_H(t_i,T_i,K_i)|$$

3. The 95% Value-at-Risk (VaR) of $\{\Delta P_H(t_i,T_i,K_i)|i=1,\ldots,m\}$
4. The 95% Conditional-Value-at-Risk (CVaR) of $\{\Delta P_H(t_i,T_i,K_i)|i=1,\ldots,m\}$

Feature Set #1:$\{$**MONEYNESS** (S_t/K), **TIME-TO-EXPIRY** $(T-t)\}$

For the synthetic data, it is known that the option price is a function of the moneyness $\frac{S}{K}$ and time to expiry $T-t$, which are the attributes $\mathbf{x}_t^{T,K}$ in Feature Set #1. Table 3.2, 3.3 and 3.4 report results for daily, weekly, and monthly hedging respectively.

Method	Gain (%)	$\mathbf{E}(\lvert\Delta V^{mkt} - \Delta S\delta\rvert)$	Std	VaR	CVaR
δ_{BS}	0.0	0.185	0.286	0.380	0.574
IKL_{RBF}	-3.3	0.171	0.291	0.356	0.566
IKL_{SPL}	-183.3	0.291	0.482	0.669	1.105
DKL_{RBF}	63.1	0.120	0.174	0.251	0.352
DKL_{SPL}	**64.9**	**0.121**	**0.170**	**0.255**	**0.345**
HESTON	63.6	0.121	0.173	0.266	0.360

Table 3.2: Daily Hedging Comparison

[1] FS #1: $\mathbf{x} = \{$MONEYNESS, TIME-TO-EXPIRY$\}$
[2] Bold entry indicating best Gain

Table 3.2 and 3.3 demonstrate that the direct hedging function learning DKL_{SPL} & DKL_{RBF} significantly outperform the indirect hedging learning IKL_{SPL} & IKL_{RBF} in Gain and different risk measures considered. Indeed, DKL_{SPL} & DKL_{RBF} slightly outperform the benchmark of using the analytic Heston delta. The indirect hedging function learning performs more poorly than the implied BS delta hedging. In addition, the spline kernel performs better than the RBF kernel (with the standard deviation as the bandwidth parameter). The RBF kernel yields larger risk measures and smaller Gain for both the direct and indirect hedging learning methods.

Table 3.4 reports hedging comparison for monthly hedging. We observe that DKL_{SPL} & DKL_{RBF} significantly outperform the indirect hedging learning IKL_{SPL} & IKL_{RBF} in Gain and various risk measures. In addition, DKL_{SPL} & DKL_{RBF} continue to achieve enhanced performance over δ_{BS}, with the spline kernel DKL_{SPL} yielding better results than DKL_{RBF}. Not

Method	Gain (%)	$\mathbf{E}(\lvert\Delta V^{mkt} - \Delta S\delta\rvert)$	Std	VaR	CVaR
δ_{BS}	0.0	0.414	0.620	0.776	1.009
IKL_{RBF}	-197.6	0.406	1.070	0.741	1.254
IKL_{SPL}	-94.7	0.548	0.866	1.114	1.738
DKL_{RBF}	47.0	0.312	0.451	0.620	0.825
DKL_{SPL}	**50.8**	**0.312**	**0.435**	**0.622**	**0.797**
HESTON	45.7	0.319	0.456	0.651	0.840

Table 3.3: Weekly Hedging Comparison

[1] FS #1: $\mathbf{x} = \{\text{MONEYNESS, TIME-TO-EXPIRY}\}$
[2] Bold entry indicating best Gain

Method	Gain (%)	$\mathbf{E}(\lvert\Delta V^{mkt} - \Delta S\delta\rvert)$	Std	VaR	CVaR
δ_{BS}	0.0	0.941	1.484	1.516	1.808
IKL_{RBF}	1.0	0.888	1.470	1.516	1.829
IKL_{SPL}	-36.9	1.135	1.729	2.033	2.894
DKL_{RBF}	33.6	0.860	1.181	1.612	1.949
DKL_{SPL}	35.4	0.858	1.165	1.610	1.922
HESTON	**38.7**	**0.814**	**1.136**	**1.544**	**1.829**

Table 3.4: Monthly Hedging Comparison

[1] FS #1: $\mathbf{x} = \{\text{MONEYNESS, TIME-TO-EXPIRY}\}$
[2] Bold entry indicating best Gain

surprisingly, hedging performance of each method also deteriorates as the length of the hedging period increases, with larger mean absolute hedging error and larger standard deviation for monthly hedging than for daily and weekly hedging.

Table 3.4 also illustrates that, unlike daily and weekly hedging, DKL_{SPL} & DKL_{RBF} slightly underperform the analytic Heston delta benchmark for monthly hedging. Given that the analytic delta is for instantaneous hedging while the direct hedging learning DKL_{SPL} minimizes quadratic hedging error, one would expect better performance from the direct hedging learning DKL_{SPL}. We suspect that this is due to the effect of the specific combination of choices of features and kernel. Next we show that, with a different feature set, performance of direct hedging is improved, which suggests the possibility of surpassing analytic Heston delta benchmark, using a more suitable feature set, for a longer period hedging.

Feature Set #2: $\{\textbf{MONEYNESS}\ (S_t/K),\ \ \textbf{TIME-TO-EXPIRY}\ (T-t),\ \ \delta^{BS}_{t,T,K}\}$

We add the Black–Scholes delta $\delta^{BS}_{t,T,K}$ using the implied volatility as an additional feature in the direct hedging learning, since Hull and White [90] indicate that a better minimum variance hedge can be calculated based on the implied volatility delta. Table 3.5, 3.6, and 3.7 present hedging results for DKL_{SPL} & DKL_{RBF} for $\{\text{MONEYNESS}\ (S_t/K),\ \text{TIME-TO-EXPIRY}\ (T-t),\ \ \delta^{BS}_{t,T,K}\}$. For clarity, we also include the results for FS #1 $\{\text{MONEYNESS}\ (S_t/K),\ \text{TIME-TO-EXPIRY}\ (T-t)\}$ and Heston delta for ease of comparison.

Method		Gain (%)	$\mathbf{E}(\lvert\Delta V^{mkt} - \Delta S\delta\rvert)$	Std	VaR	CVaR
DKL_{RBF}	FS #1	63.1	0.120	0.174	0.251	0.352
	FS #2	62.3	0.114	0.176	0.238	0.349
DKL_{SPL}	FS #1	64.9	0.121	0.170	0.255	0.345
	FS #2	**70.9**	**0.110**	**0.154**	**0.234**	**0.322**
HESTON		63.6	0.121	0.173	0.266	0.360

Table 3.5: Daily Hedging Comparison

[1] FS #2: $\mathbf{x} = \{\text{MONEYNESS, TIME-TO-EXPIRY, } \delta_{\text{BS}}\}$

[2] Bold entry indicating best Gain

From Table 3.5, 3.6 and 3.7, we observe that, for daily and weekly hedging, including the BS delta further improves the performance of the direct hedging learning methods, which outperforms analytical delta hedging. For monthly hedging, however, the performance is similar to what we obtain with that of the feature set #1. Overall, including the BS delta as an attribute is beneficial for the direct hedging function learning.

Method		Gain (%)	$\mathbf{E}(\lvert\Delta V^{mkt} - \Delta S\delta\rvert)$	Std	VaR	CVaR
DKL_{RBF}	FS #1	47.0	0.312	0.451	0.620	0.825
	FS #2	51.4	0.301	0.432	0.611	0.816
DKL_{SPL}	FS #1	50.8	0.312	0.435	0.622	0.797
	FS #2	**53.5**	**0.299**	**0.422**	**0.606**	**0.794**
HESTON		45.7	0.319	0.456	0.651	0.840

Table 3.6: Weekly Hedging Comparison

[1] FS #2: $\mathbf{x} = \{\text{MONEYNESS, TIME-TO-EXPIRY, } \delta_{\text{BS}}\}$

[2] Bold entry indicating best Gain

Method		Gain (%)	$\mathbf{E}(\lvert\Delta V^{mkt} - \Delta S\delta\rvert)$	Std	VaR	CVaR
DKL_{RBF}	FS #1	33.6	0.860	1.181	1.612	1.949
	FS #2	30.1	0.863	1.217	1.652	2.104
DKL_{SPL}	FS #1	35.4	0.858	1.165	1.610	1.922
	FS #2	36.6	0.836	1.156	1.609	1.953
HESTON		**38.7**	**0.814**	**1.136**	**1.544**	**1.829**

Table 3.7: Monthly Hedging Comparison

[1] FS #2: $\mathbf{x} = \{\text{MONEYNESS, TIME-TO-EXPIRY, } \delta_{\text{BS}}\}$

[2] Bold entry indicating best Gain

3.5 Enhancement Over the Kernel Local Hedging Model

In this chapter, we illustrate that, even in a nonparametric kernel approach to model the option value function, dependence on the underlying can exist for the estimated kernel parameters. Consequently, using the partial derivatives of the model option pricing function, parametrically or nonparametrically estimated, will fail to minimize hedging error, even instantaneously.

Thus, we propose to directly learn nonparametric kernel hedging functions by minimizing the sum of square of the discrete local hedging risk, bypassing the intermediate step of the option value function estimation. Using synthetic data, we first demonstrate that the proposed direct hedging function learning significantly outperforms hedging based on the sensitivity of the model option function learned nonparametrically. In addition, we demonstrate that spline kernel yields better hedging performance in comparison to that of the RBF kernel. The exploratory research in this chapter clearly demonstrates the potential role of a market data-driven approach for financial derivative modeling and risk management.

However, we also notice the positive gain ratios (2.3.17) are smaller for monthly hedging in comparison to daily and weekly hedging for the synthetic data. In addition, when testing on real market S&P 500 index option with the DKL_{SPL} model and the feature set #2, we observed the similar phenomenon as it is for the synthetic scenario: the gain ratio decreases when we move from daily hedging to weekly and monthly hedging. The details of the real data comparison can be found in [118] and will be discussed in details in chapter 5.

We suspect that neither the feature set #1 nor the feature set #2 is enough to learn a sufficient data-driven local hedging model for longer period such as weekly and monthly. We are thus motivated to enhance the data-driven local hedging model using the RNN framework by adding the following components to include more features in computing the hedging position:

- Feature selection process.
- Sequential feature extraction process.

We demonstrate using real S&P 500 index option data that such enhancement greatly improve the local hedging performance. Besides, the GRU_δ proposed in Chapter 4 is more computational efficient enabling us to update the model more frequently to incorporate market changes. Details of the enhanced data-driven local hedging model GRU_δ will be discussed in chapter 4. The experimental results of DKL_{SPL} and GRU_δ on real $S\&P500$ index option data will be discussed in chapter 5.

Chapter 4

Data-Driven Sequential Learning Framework for Local Hedging Risk

In chapter 3, we have discussed the exploratory data-driven kernel hedging model $\mathrm{DKL}_{\mathrm{SPL}}$ [118]. However, we believe that the data-driven hedging learning approach in [118] can potentially benefit from further improvements in a few directions. Firstly, Nian et al. [118] only use moneyness, time-to-expiry, and Black–Scholes delta as features to predict the hedging position. In practice, since both the underlying and options markets have complex price dynamics, feature extraction and feature selection can potentially further enhance hedging performance. Secondly, when predicting the hedging position at the rebalancing time t, only attributes observed at t are used as features in [118]. However, a financial market exhibits volatility clustering, describing a positive, significant, and slowly decaying volatility autocorrelation [107]. For example, a generalized autoregressive conditionally heteroskedastic (GARCH) model has been proposed for option pricing because of its capability in better characterizing asset returns [82] . It has been shown that the option pricing function under a GARCH model depends not only on the current underlying price but also on the observed underlying price history [49, 82]. This suggests that the information immediately prior to the rebalancing time should be relevant in determining the hedging position at the rebalancing time. Thirdly, the mean squared error is used as the loss function in [118] but a more appropriate robust objective function and framework may lead to a more stable optimal learning as the market shifts between crisis and normal regimes. Lastly, when a kernel data-driven model needs to be updated frequently, e.g., daily, it is computationally prohibitive to conduct backtesting over a long time period (e.g., a decade). Consequently the regularized spline kernel network is only updated monthly in [118]. To accurately assess the potential of a data-driven hedging approach, a model which is sufficiently computationally efficient needs to be considered to allow more frequent updating in a back-testing study of a long time horizon.

To incorporate sequential information in the hedging model, we use RNN, a feed-forward neural networks augmented with edges that connect adjacent steps. In the early 1980s, Hopfield [86] introduced RNN for sequential pattern recognitions. Jordan [96] and Elman [56] developed basic architectures for RNN with a group of neural networks that have a ”memory” to capture past information. This can be beneficial in time series applications.

Training RNN is similar to training the traditional neural network, with the Stochastic Gradient Descent (SGD) as the primary optimization tool. The Back Propagation Through Time (BPTT) algorithm is used to calculate gradients. However, vanishing and exploding gradients [85] can occur when back-propagating errors across many steps, which pose challenges for standard RNN architectures to learn long-term dependence between steps. The Long Short-Term Memory (LSTM) [84] model and the Gated Recurrent Unit (GRU) model [33] are subsequently proposed to address issues of vanishing and exploding gradients. LSTM and GRU utilize a gate structure with element-wise operations, which can retain information in memory for a longer period. This alleviates the problem of vanishing and exploding gradients [84]. GRU and LSTM models are shown to perform better than the standard RNN model [33], although performances of GRU and LSTM are comparable. Since GRU has fewer parameters than LSTM and usually require less training data [157], we use the GRU in the proposed encode-decoder hedging model GRU_δ.

4.1 The Proposed GRU_δ for market data-driven hedging

4.1.1 Local Discrete Hedging GRU Model

Figure 6.4 depicts the proposed local discrete GRU hedging model GRU_δ, which uses an encoder-decoder structure to combine the local and sequential features, similar to the sequence to sequence (seq2seq) models [33]. This model uses both the local features at the hedging time and the sequential features, which encode information immediately before the hedging time.

Consider a data instance at the hedging time t, strike K, and expiry T. Let the associated change of the market option price be $\Delta V^{mkt}_{t,T,K}$ and underlying price change be ΔS_t, as in (2.2.1). Assume that at the hedging time t, we have local features vector $\mathbf{x}_t^{T,K} \in \mathbb{R}^{d_l}$ which records local information at the hedging time t for hedging the option with expiry T and strike K.

Let Δt_d denote the time interval for sequential information recording. In the subsequent empirical study, the interval Δt_d equals one-day. We denote the sequential features recording the daily history for hedging the option with expiry T and strike K as

$$\mathbf{Y}_t^{T,K} = \left[\mathbf{y}_{t-N\Delta t_d}^{T,K}, \ldots, \mathbf{y}_t^{T,K}\right]$$

For notational simplicity, we denote $\tau_i = t-(N+1-i)\Delta t_d$ with $i = 1,\ldots,N+1$, we thus have:

$$\mathbf{Y}_t^{T,K} = \left[\mathbf{y}_{\tau_1}^{T,K}, \ldots, \mathbf{y}_{\tau_{N+1}}^{T,K}\right]$$

The vector $\mathbf{y}_{\tau_i}^{T,K} \in \mathbb{R}^{d_s}$ has d_s features at time τ_i in the input sequential feature. Thus d_l is the dimension for the local feature $\mathbf{x}_t^{T,K}$, d_s is the dimension for the sequential feature $\mathbf{Y}_t^{T,K}$, and $N+1$ is the length of the sequential feature sequence.

The encoder transforms information from the sequential feature $\mathbf{Y}_t^{T,K}$ to a fixed-sized vector $\widehat{\mathbf{h}}_E$ and the decoder makes the final prediction based on both $\widehat{\mathbf{h}}_E$ and the local feature $\mathbf{x}_t^{T,K}$. The overall structure of the proposed model is illustrated in Figure 6.4. Next we discuss each component of the proposed encoder-decoder model in details.

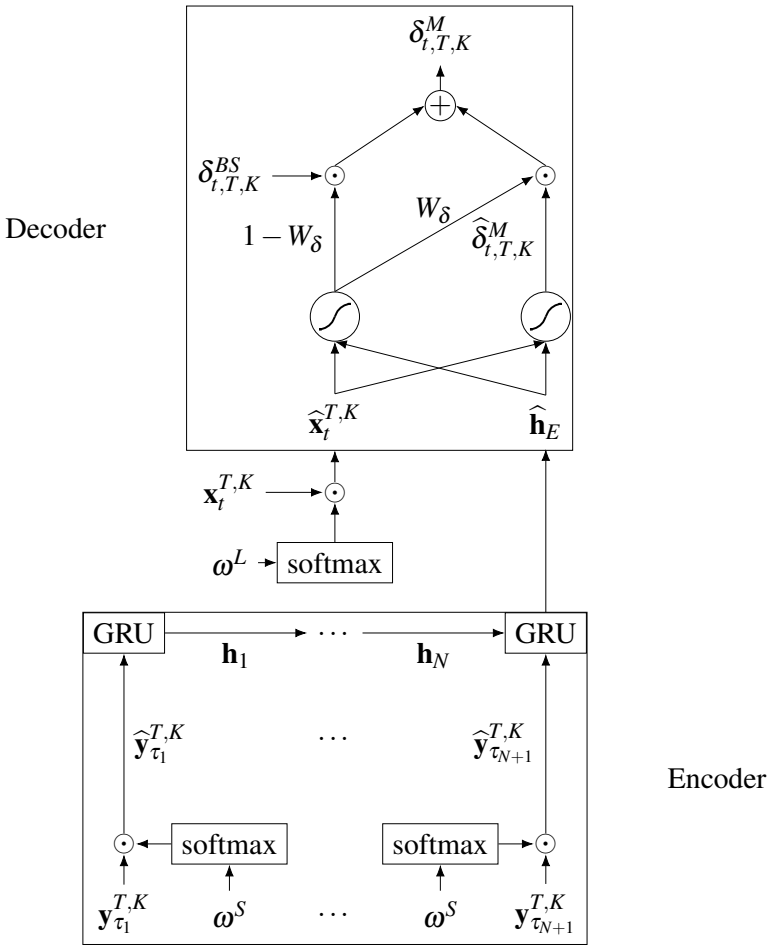


Figure 4.1: GRU_δ: GRU Encode-Decoder Hedging Model

4.1.2 Feature Selection via Embedded Feature Weighting

Feature selection improves machine learning performance by eliminating noise and providing better interpretability. While various feature selection frameworks have been proposed, we consider the feature weighting method [142], which embeds feature selection in the SVM training. This embedded feature weighting method is shown to outperform other state-of-the-art embedded feature selection methods [142]. We adopt a similar feature weighting embedding technique in the discrete GRU model to conduct feature selections on both the local feature $\mathbf{x}_t^{T,K}$ and the

sequential feature $\mathbf{Y}_t^{T,K}$. Features are first weighted and then fed into the encoder and decoder as inputs. Feature weights are optimized during model training.

We use a softmax function to generate a normalized feature weighting vector. For the local feature $\mathbf{x}_t^{T,K} \in \mathbb{R}^{d_l}$, the j^{th} component of the normalized weight vector is given by

$$\frac{exp(\omega_j^L)}{\sum_{i=1}^{d_l} exp(\omega_i^L)}$$

The weighted local feature vector is defined as

$$\widehat{\mathbf{x}}_t^{T,K} = \frac{exp(\omega^L)}{\sum_{i=1}^{d_l} exp(\omega_i^L)} \odot \mathbf{x}_t^{T,K}$$

where $\odot$ denotes the element-wise multiplication.

Similarly, for the sequential feature $\mathbf{y}_{\tau_i}^{T,K}$, the j^{th} component of the normalized weight vector is given by

$$\frac{exp(\omega_j^S)}{\sum_{i=1}^{d_s} exp(\omega_i^S)}$$

The weighted feature vector at time τ_i is defined as

$$\widehat{\mathbf{y}}_{\tau_i}^{T,K} = \frac{exp(\omega^S)}{\sum_{j=1}^{d_s} exp(\omega_j^S)} \odot \mathbf{y}_{\tau_i}^{T,K}$$

The weighting procedure acts as a feature selection. If a particular feature is not relevant in predicting the hedging position, the associated weight after learning is expected to be negligible.

4.1.3 GRU Encoder

The proposed GRU_δ in Figure 6.4 has an one-layer GRU, which encodes the sequential feature $\mathbf{Y}_t^{T,K}$ to a fixed-sized vector $\widehat{\mathbf{h}}_E$. At the step i, the encoder computes the value of the hidden state $\mathbf{h}_i$ using a GRU cell. The input at the step i of the encoder is $\widehat{\mathbf{y}}_{\tau_i}^{T,K}$, $i = 1, \ldots, N+1$. The internal structure of the GRU cell is shown in Figure 4.2.

Let $\boldsymbol{W}_z, \boldsymbol{U}_z, \mathbf{b}_z, \boldsymbol{W}_r, \boldsymbol{U}_r, \mathbf{b}_r, \boldsymbol{W}_h, \boldsymbol{U}_h, \mathbf{b}_h$ denote parameters shared by all GRU cells.

1. The **update gate** decides how much the cell updates its activation:

$$\mathbf{z}_i = sigmoid(\boldsymbol{W}_z \widehat{\mathbf{y}}_{\tau_i}^{T,K} + \boldsymbol{U}_z \mathbf{h}_{i-1} + \mathbf{b}_z)$$

2. The **reset gate** decides how much information to retain from the previous hidden state $\mathbf{h}_{i-1}$:

$$\mathbf{r}_i = sigmoid(\boldsymbol{W}_r \widehat{\mathbf{y}}_{\tau_i}^{T,K} + \boldsymbol{U}_r \mathbf{h}_{i-1} + \mathbf{b}_r)$$

3. The **candidate** hidden state value $\widehat{\mathbf{h}}_i$ is computed from the current input $\widehat{\mathbf{y}}_{\tau_i}^{T,K}$, previous hidden state $\mathbf{h}_{i-1}$, and the reset value $\mathbf{r_i}$:

$$\widehat{\mathbf{h}}_i = tanh(\boldsymbol{W}_h \widehat{\mathbf{y}}_{\tau_i}^{T,K} + \boldsymbol{U}_h(\mathbf{r}_i \odot \mathbf{h}_{i-1}) + \mathbf{b}_h)$$

4. The **output** hidden state value $\mathbf{h}_i$ is computed based on a weighted combination of the previous activation $\mathbf{h}_{i-1}$ and the candidate activation $\widehat{\mathbf{h}}_i$:

$$\mathbf{h}_i = (1 - \mathbf{z}_i) \odot \mathbf{h}_{i-1} + \mathbf{z}_i \odot \widehat{\mathbf{h}}_i$$

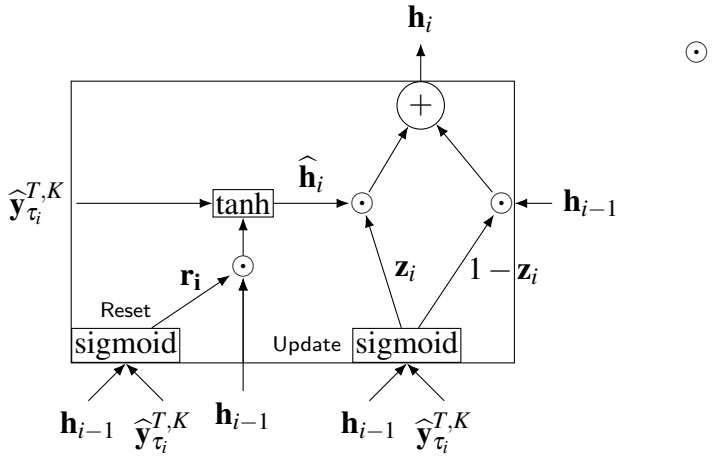


Figure 4.2: Illustration of the GRU Cell

The hidden state at the last step $\mathbf{h}_{N+1}$, corresponding to time $\tau_{N+1} = t$, is supplied to the decoder as the fixed size vector $\widehat{\mathbf{h}}_E$, which extracts relevant information in $\mathbf{Y}_t^{T,K}$.

4.1.4 Decoder

The decoder combines the output of the encoder from the sequential feature $\mathbf{Y}_t^{T,K}$ at the hedging time t with the current local feature $\mathbf{x}_t^{T,K}$. It uses one neural network to first compute a candidate output $\widehat{\delta}_{t,T,K}^M$, based on both the weighted local input $\widehat{\mathbf{x}}_t^{T,K}$ and the fixed size vector $\widehat{\mathbf{h}}_E$:

$$\widehat{\delta}_{t,T,K}^M = sigmoid(\mathbf{v}_{out}^T \, tanh(\boldsymbol{U}_{out} \widehat{\mathbf{h}}_E + \boldsymbol{W}_{out} \widehat{\mathbf{x}}_t^{T,K} + \mathbf{b}_{out})).$$

The the output gate value W_δ is given by another neural network, based on both the weighted local input $\widehat{\mathbf{x}}_t^{T,K}$ and the fixed size vector $\widehat{\mathbf{h}}_E$:

$$W_\delta = sigmoid(\mathbf{v}_{Gate}^T \, tanh(\boldsymbol{U}_{Gate} \widehat{\mathbf{h}}_E + \boldsymbol{W}_{Gate} \widehat{\mathbf{x}}_t^{T,K} + \mathbf{b}_{Gate})). \tag{4.1.1}$$

The practitioner's BS delta encapsulates a significant portion of the sensitivity of the option price to the underlying and has been adopted in option hedging practice. Therefore we use BS implied volatility delta as a pre-trained model in the designed GRU_δ to utilize both local and sequential features to minimize hedge risk. Specifically, the output of the proposed GRU_δ is a linear combination of the candidate output $\widehat{\delta}^M_{t,T,K}$ and the BS delta $\delta^{BS}_{t,T,K}$ computed from the implied volatility at the hedging time t for the option with expiry T and strike K. In addition, different combination formulae are used when training for different types of options. For hedging a call option, the final output from GRU_δ is :

$$\delta^M_{t,T,K} = \widehat{\delta}^M_{t,T,K} \times W_\delta + \delta^{BS}_{t,T,K} \times (1 - W_\delta) \tag{4.1.2}$$

For hedging a put option, the final output from the model is:

$$\delta^M_{t,T,K} = -\widehat{\delta}^M_{t,T,K} \times W_\delta + \delta^{BS}_{t,T,K} \times (1 - W_\delta) \tag{4.1.3}$$

where $\widehat{\delta}^M_{t,T,K}$ is the candidate output. When $W_\delta = 0$, $\delta^{BS}_{t,T,K}$ is the output and, when $W_\delta = 1$, the output is the candidate $\widehat{\delta}^M_{t,T,K}$. The combination weight W_δ is defined in (4.1.1). The sigmoid function is used as the final activation function because, under the Black-Scholes-Merton framework [18], the range of the hedging position for call options is $[0,1]$ and the range of the hedging position for put options is $[-1,0]$. It can be easily shown that $\delta^M_{t,T,K}$, given by (4.1.2), is within $[0,1]$ and $\delta^M_{t,T,K}$, given by (4.1.3), is within $[-1,0]$.

4.1.5 Robust Loss Function

At the hedging time t_i, for the data instance i with strike K_i, and expiry T_i , the local discrete hedging loss is:

$$loss_i = \Delta V^{mkt}_{t_i,T_i,K_i} - \Delta S_{t_i} \delta^M_{t_i,T_i,K_i}$$

where $\delta^M_{t_i,T_i,K_i}$ is the corresponding final output from GRU_δ, ΔS_{t_i} denotes the change in the market underlying price, and $\Delta V^{mkt}_{t_i,T_i,K_i}$ denotes the change in the market option price, see (2.2.1). Let M be the number of data instances. The objective function used in the data-driven regularized kernel hedging model in [118] is the mean squared loss,

$$MSE = \frac{1}{2M} \sum_{i=1}^{M} loss_i^2. \tag{4.1.4}$$

Since the quadratic error function is sensitive to outliers, in this work, we additionally incorporate the Huber loss function [88] below in the proposed GRU_δ:

$$HE = \frac{1}{M} \sum_{i=1}^{M} Huber(loss_i),$$

where $Huber(\cdot)$ is as defined below:

$$Huber(loss, \mathscr{T}) = \begin{cases} \frac{1}{2}loss^2, & \text{if } |loss| \leq \mathscr{T} \\ \mathscr{T}(|loss| - \frac{1}{2}\mathscr{T}), & \text{otherwise} \end{cases}$$

The Huber loss [88] has been shown to be more robust than the squared loss with respect to toutliers, when the threshold parameter $\mathscr{T}$ is carefully chosen *a priori*. In the context of machine learning, the parameter $\mathscr{T}$ can be tuned but this can be computationally expensive. In the proposed GRU$_\delta$, since delta from the BS model with the implied volatility is used as a pre-trained model, we adaptively set the thresholding parameter $\mathscr{T}$ to be absolute value of the hedging error from the Black–Scholes delta of the data instance i, i.e.,

$$\mathscr{T}_i = |\Delta V^{mkt}_{t_i,T_i,K_i} - \Delta S_{t_i} \delta^{BS}_{t_i,T_i,K_i}| \tag{4.1.5}$$

where $\delta^{BS}_{t_i,T_i,K_i}$ is the Black–Scholes delta using implied volatility for data instance i. The modified Huber loss becomes

$$MHuber(loss_i, \mathscr{T}_i) = \begin{cases} \frac{1}{2}loss_i^2, & \text{if } |loss_i| \leq \mathscr{T}_i \\ \mathscr{T}_i(|loss_i| - \frac{1}{2}\mathscr{T}_i), & \text{otherwise} \end{cases}$$

Using the Huber loss with the adaptive threshholding parameter in (4.1.5), the objective for the proposed GRU$_\delta$ is:

$$MHE = \frac{1}{M}\sum_{i=1}^{M} MHuber(loss_i, \mathscr{T}_i) \tag{4.1.6}$$

4.2 Training GRU$_\delta$

Next we discuss training GRU$_\delta$, including initialization, pre-training, optimization, and regularization.

4.2.1 Initialization

When training a RNN model, an initial weight matrix is typically chosen as a random orthogonal matrix. Since the orthogonal initialization can often speed up training [100], all the weight matrices are initialized as orthogonal random matrices in training GRU$_\delta$. The random orthogonal matrices are Householder transformations of random matrices [100].

4.2.2 Optimization

First-order optimization methods, such as stochastic gradient descent (SGD) and its extensions, are the most widely used optimization methods in machine learning, due to their low computational costs. Despite their wide usage, well-known deficiencies when training highly non-convex

objective functions, including relatively-slow convergence, sensitivity to hyper-parameter values (e.g., learning rate), stagnation at high training errors, and difficulty in escaping flat regions and saddle points [155].

More recently, improved SGD methods such as ADAM [97], have been proposed. These methods seem to achieve significantly better solutions compared with earlier SGD. However, it has recently been empirically observed in [128] that these algorithms sometimes fail to converge to a first-order critical point due to the exponential moving average used in the algorithms.

Since the data size and the number of the parameters in GRU_δ are relatively small (which is a distinguishing characteristics of many financial data mining problems), it is computationally feasible to apply a second order optimization method. Furthermore, the recently proposed trust-region sub-problem solver [101] computes the trust region sub-problem solution iteratively requiring only Hessian-vector products. Consequently we use a trust region method, which is shown in Algorithm 1 when training GRU_δ. The meta-parameters for trust-region algorithm are shown in Table 4.1.

$\mathscr{R}_0$	ε_θ	ε_r	η_{r_1}	η_{r_2}	γ_u	γ_d
1.0	10^{-5}	10^{-8}	0.25	0.75	2	0.5

Table 4.1: Meta-Parameters for the Trust-Region Algorithm 1

4.2.3 Pre-training and Regularization

We use early stopping as the regularization [126]. At each training date, we reserve a fraction of the observed data to be the validation set. The rest of the observed data is used as the training set. The performance on the validation set is used to determine when over-fitting begins. Let θ denote the set of parameters to be learned for the proposed GRU_δ. After each training step k, the model performance is evaluated on the validation set and the associated parameters θ_k is recorded. Model training optimization is performed on the training set until the trust-region optimizer terminates. The parameters that achieve the best performance on the validation set is used to predict the hedging position for the testing data instances.

For GRU_δ, the training optimization problem is nonconvex, which can benefit from a good initial model. We choose the initial model for the training problem by matching the output of GRU_δ to the pre-trained BS model $\delta^{BS}_{t,T,K}$. Specifically, we determine the initial model for training optimization by solving the nonlinear least square problem below,

$$\min \frac{1}{2M} \sum_{i=1}^{M} (\delta^{M}_{t_i,T_i,K_i} - \delta^{BS}_{t_i,T_i,K_i})^2, \tag{4.2.1}$$

where the starting point for (4.2.1) is a set of randomly initialized weight matrices, as described in section 4.2.1.

Algorithm 1: Trust-region Algorithm

Input: $\theta_0 \in \mathbb{R}^n$: initial vector of parameters.
$Obj(\theta)$: the objective function (For GRU_δ, we use either (4.1.4) or (4.1.6))
$\mathscr{R}_0$: initial trust region radius
ε_θ: tolerance for the norm of the gradient
ε_r: tolerance for the trust region radius
η_{r_1}: first threshold for update the trust region radius
η_{r_2}: second threshold for update the trust region radius
$\gamma_u > 1$: ratio to increase the trust-region radius
$\gamma_d < 1$: ratio to decrease the trust-region radius

Output: θ^*: the vector of parameters that minimize the objective function $Obj(\theta)$

begin

$k = 0$;

while $\|\nabla Obj(\theta_k)\|_2 \geq \varepsilon_\theta$ *and* $\mathscr{R}_k \geq \varepsilon_r$ **do**

solve the trust-region subproblem [101]:

$$s_k^* = \underset{s}{\text{argmin}}\, m_k(s) = Obj(\theta_k) + s^T \nabla Obj(\theta_k) + s^T \nabla^2 Obj(\theta_k) s$$
$$\text{sub to} \|s\|_2 \leq \mathscr{R}_k$$

define

$$\mathscr{P}_k = \frac{Obj(\theta_k) - Obj(\theta_k + s_k^*)}{Obj(\theta_k) - m_k(s_k^*)}$$

Update the radius: **if** $\eta_{r_2} > \mathscr{P}_k < \eta_{r_1}$ **then**

$\theta_{k+1} = \theta k + s_k^*$

else

$\theta_{k+1} = \theta k$

end

$$\mathscr{R}_{k+1} = \begin{cases} \gamma_u \|s_k^*\|_2, & \text{if } \mathscr{P}_k < \eta_{r_1} \\ \mathscr{R}_k, & \text{if } \eta_{r_1} \leq \mathscr{P}_k \leq \eta_{r_2} \\ \max(\gamma_u \|s_k^*\|_2, \mathscr{R}_k), & \text{if } \mathscr{P}_k > \eta_{r_2} \end{cases}$$

k=k+1

end

$$\theta^* = \theta_k$$

end

Using a solution to (4.2.1) as the initial model, we train GRU_δ with the robust objective function (4.1.6). Recall that the output of GRU_δ, either (4.1.2) or (4.1.3), is a linear combination of the candidate output $\widehat{\delta}^M_{t,T,K}$ and the BS delta $\delta^{BS}_{t,T,K}$ computed from the implied volatility. When W_δ approaches 0, the output of GRU_δ converges to $\delta^{BS}_{t,T,K}$ as the output. When W_δ approaches 1, GRU_δ outputs candidate $\widehat{\delta}^M_{t,T,K}$. This simple combination structure allows the pre-training phase to be finished in just a few training steps.[1] The pre-training stage guarantees that the initial performance of the model on the validation set is close to $\delta^{BS}_{t,T,K}$. If the output from the model after training performs worse than $\delta^{BS}_{t,T,K}$ on the validation set, we use the initial model after pre-training.

Additionally the validation set is also used to select whether to use the mean square objective function (4.1.4) or the modified Huber loss objective (4.1.6) . The model trained with the objective function that performs better on the validation set is used when predicting the hedging position of the testing data instances, which are unobserved at the time of training.

4.2.4 Model Re-use and Re-initialization

In contrast to the regularized kernel method in [118], one advantage of the proposed GRU_δ is that the parameters from one trained model can be readily reused as the starting point for the model training on a subsequent rebalancing day. This allows a model to be updated more frequently, adapting to market changes without completely rebuilding a model. The kernel method in [118], on the other hand, recomputes the kernel matrix at each rebalancing time, which requires $O(m^3)$ computation assuming a completely new kernel matrix. Thus, updating the model frequently using the kernel method in [118] is computationally prohibitive.

An artificial neural network is known to potentially suffer catastrophic interference or catastrophic forgetting [111], i.e., a model forgets completely and abruptly previously learned information upon observing new information. When this happens, the resulting broken model usually has poor generalization ability. If a broken model with poor generalization ability is allowed to be continually updated, the performance of all the future models may be negatively affected. Thus, whenever a worse performance, in comparison to $\delta^{BS}_{t,T,K}$, is observed on a validation set, we re-initialize the parameters of GRU_δ and pre-train the model again. This avoids continually updating the broken model. If after model re-initialization and training, the performance of the proposed GRU_δ remains worse than that of $\delta^{BS}_{t,T,K}$ on the validation set, we simply output $\delta^{BS}_{t,T,K}$.

In summary, we initialize the model as described in section 4.2.1 and 4.2.3 on the first testing date. When we train the model, we reuse the parameters from the previously trained model as the starting point unless the performance of the proposed GRU_δ is worse than that of $\delta^{BS}_{t,T,K}$, in which case we re-initialize the parameters of the proposed GRU_δ and train the model again.

[1] Note that we will also stop the pre-training when $W_\delta > 0.97.5$ or $W_\delta < 0.025$. This is because, when we have $W_\delta > 0.97.5$ or $W_\delta < 0.025$, due to saturating gradient problem of the sigmoid function, the initial gradient, when training with actual local hedging objective, will be very small. This will slow down the first few learning steps when training with actual local hedging objective.

4.3 Alternative Data-Driven Hedging Models Under Neural Network Framework

Lastly, before jumping into the real data experimental results for local discrete hedging in Chapter 5, we introduce the following supplementary models:

1. To gain insights on the roles of the decoder and encoder in the proposed GRU_δ, we also consider a decoder only model NN_δ, which corresponds to removing the encoder from GRU_δ in Figure 4.3. We train NN_δ in the same way as described in section 4.2.

2. To gain insights on the role of output gate and robust Huber loss, we remove the output gate part of the decoder model. The resulting model is shown in Figure 4.4. In other words, the output δ^M is purely based $\widehat{\mathbf{h}}_E$ and $\widehat{\mathbf{x}}_t^{T,K}$:

$$\delta_{t,T,K}^M = sigmoid(\mathbf{v}_{out}^T \, tanh(\boldsymbol{U}_{out}\widehat{\mathbf{h}}_E + \boldsymbol{W}_{out}\widehat{\mathbf{x}}_t^{T,K} + \mathbf{b}_{out})).$$

   We denote the model shown in Figure 4.4 as GRU_c. The training procedure of GRU_c is slightly different from GRU_δ:

   - Early stopping, pre-training, and model re-initialization mentioned in section 4.2 are not used.
   - The validation set is not needed since early stopping and model re-initialization are not used.
   - The objective function is fixed to be the mean squared loss.

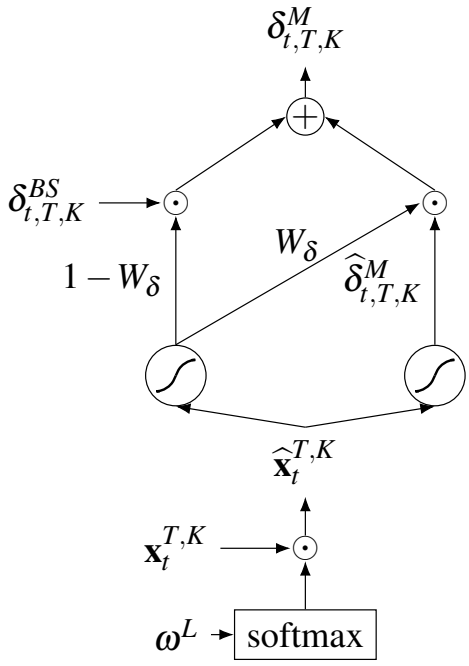


Figure 4.3: NN_δ: decoder only model

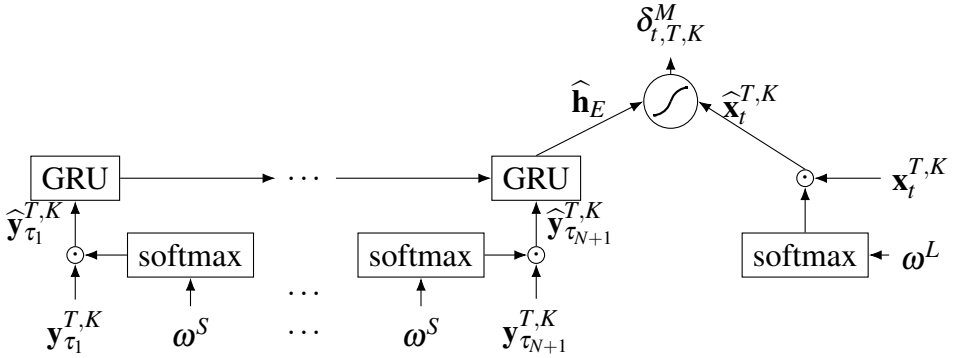


Figure 4.4: GRU_c: Removing the Output Gate

Chapter 5

Local Discrete Hedging Performance Comparison Using S&P 500 index Options

Using the S&P 500 (European) index option market data from January 2, 2001 to August 31, 2015, we compare hedging performance of different hedging strategies.

Following [90], we use the Gain ratio below as the evaluating measure for the hedging performance:

$$\text{GAIN} = 1 - \frac{\sum_{i=1}^{m} \left((\Delta V^{mkt}_{t_i,T_i,K_i} - \delta_{t_i,T_i,K_i} \, \Delta S_{t_i} \right)^2}{\sum_{i=1}^{m} \left((\Delta V^{mkt}_{t_i,T_i,K_i} - \delta^{BS}_{t_i,T_i,K_i} \, \Delta S_{t_i} \right)^2},$$

where m is the total number of data instances to be evaluated. In addition, for the data instance i, corresponding to hedging option with expiry T_i and strike K_i at hedging time t_i, we recall that $\delta^{BS}_{t_i,T_i,K_i}$ is the implied Black–Scholes delta, δ_{t_i,T_i,K_i} is the hedging position from the method under consideration, e.g., SABR delta, MV delta, and output from DKL_{SPL} or GRU_{δ}. In addition $\Delta V^{mkt}_{t_i,T_i,K_i}$ is the change in option price, and ΔS_{t_i} is the change in the underlying price which are given by (2.2.1).

5.1 Data and Experimental Setting

The option data used in this study comes from the OptionMetric database and the data is processed following the same procedure described in [90]. On each day, the mid-price from the bid and ask is used as the price of the option on that day. The closing price for the underlying is regarded as the daily underlying price. Options with time-to-expiry less than 14 days are removed from the data set. The call options with the BS delta $\delta^{BS}_{t,T,K}$ from the implied volatility outside of [0.05, 0.95] and the put options with the BS delta $\delta^{BS}_{t,T,K}$ from implied volatility outside of [-0.95, -0.05] are also removed from the data set. These removed option prices are noisy and unreliable

since they correspond to either deeply out-of-money options, deeply in-money option, or very short term options. As in [90], the option data instances are divided into nine buckets according to the Black–Scholes delta δ^{BS} from the implied volatility (rounded to the nearest tenth) for detailed hedging performance comparisons.

Inputs to GRU_δ, $\mathbf{Y}_t^{T,K} \in \mathbb{R}^{d_s \times (N+1)}$, are 6 sequential features and the length of each sequence is 6, i.e., $d_s = 6$ and $N = 5$. At each hedging time t, the six features are:

1. option middle price for the previous 5 business days and the hedging day t,
2. option implied volatility for the previous 5 business days and the hedging day t,
3. option Black–Scholes delta for the previous 5 business days and the hedging day t,
4. option Black–Scholes gamma for the previous 5 business days and the hedging day t,
5. option Black–Scholes vega for the previous 5 business days and the hedging day t,
6. moneyness for the previous 5 business days and the hedging day t.

At each hedging time t, there are 10 local features $\mathbf{x}_t^{T,K}$:

1. moneyness,
2. time to expiry,
3. index close price,
4. option bid price,
5. option offer price;
6. option Black–Scholes delta,
7. minimum variance delta δ_{MV} (2.3.4) estimated using the training data set,
8. option implied volatility;
9. option Black–Scholes gamma;
10. option Black–Scholes vega.

The number of hidden states, for the single-layer GRU encoder, the neural network outputting $\widehat{\delta}^M_{t,T,K}$, and the neural network outputting W_δ in Figure 6.4, are all set to be 5. In comparison with most RNN applications, we note we train a relatively small model and the number of data instances available for the hedging problem is also relatively small. We note that our training and testing data instances are generated from daily observations with a moving window. On each day, data instances observed in the previous 36 months are reserved as the training set and

the validation set. For daily and weekly hedging, the validation consists of the data instances observed during the past 30 days. For monthly hedging, the validation set is the data instances observed during the previous 60 days. This is because, for monthly hedging, the ΔS_t and $\Delta V^{mkt}_{t,T,K}$ of many data instances during the last 30 days have not been observed. The data instances, which are observed in the previous 36 months but are not used in the validation set, are used as the training set. The models are updated daily.

We note that the length of our backtest time is longer than a decade. Since learning a single DKL_{SPL} model for all options will be computationally expensive, a separate model is built for each delta bucket when training DKL_{SPL}, which has the feature set # 2

$$\{\text{MONEYNESS, TIME-TO-EXPIRY, BLACK–SCHOLES DELTA}\}.$$

In addition, we use the efficient leave-one-out cross-validation (LOOCV) as in section 3.3 to choose regularization parameter λ_P for DKL_{SPL}. For each month, LOOCV is used to choose a value of the regularization parameter from a candidate set:

$$\lambda_P \in \{10^7, 10^6, 10^5, 10^4, 10^3, 10^2, 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}\}.$$

For the call option, the *DKLs* for each bucket is estimated using all options **traded** in a 36 months window. For the put option, there is much larger variation in the number of data instances in each bucket. To ensure a reasonable training data size, we have used a time window of 24 months for delta buckets -0.1 and -0.2, window of 36 months for delta buckets -0.3 and -0.4, window of 48 months for delta buckets -0.5 and -0.6, and window of 72 months for delta buckets -0.7, -0.8 and -0.9. Hence, the training data for puts ranges from Jan 2001 to August 31 2015. We train the direct hedging function model DKL_{SPL} using all options **traded** in a fixed time window, as described above, and determine the hedge each day in the following month using the trained model, following the same sliding training-testing window procedure in [90].Furthermore, the DKL_{SPL} model is updated monthly due to the limitation of computationally cost.

The training set, the validation set, and the testing set are the same for GRU_δ and NN_δ. The training procedure is the same for GRU_δ and NN_δ which is described in section 4.2. In addition, GRU_c is trained using the data instances, observed in the previous 36 months and procedure described as in section 4.3.

For additional reference, we also include the performance of the MV model (2.3.4) and Bartlett delta on weekly and monthly hedging. For MV in Table 5.1 & 5.2, on each day, the model parameter, a, b and c are estimated using all traded options in a 36-month-window. [1]

For DKL_{SPL} and GRU_δ, we present out-of-sample daily hedging performance test result using either traded data or all data in the database.

[1] Note that, in the OptionMetric Database, not all the option data instances are actually traded. For many data instances, the trading volumes are zero. It is not clear to us whether the reported test results in [90] are computed using all data or traded data only.

5.1.1 Weekly and Monthly Hedging Comparisons

We first present hedging comparisons for weekly and monthly hedging. We assume a period of 5-business-days for weekly hedging and a period of 20-business-days for monthly hedging. We note that the MV hedging method is only considered for daily hedging in [90].

Specifically, we compare GRU_δ with the following methods,

- GRU_δ: The sequential learning framework trained as in section 4.2.
- MV: MV hedging based on formulation (2.3.4),
- Bartlett: Barlett corrective delta based on (2.3.16),
- DKL_{SPL}: Direct spline kernel hedging method as in 3.2,
- NN_δ: NN with the decoder only described in Figure 4.3.
- GRU_c: The sequential learning model excluding the output gate and trained as in section 4.3.

Table 5.1 present comparisons for calls while Table 5.2 demonstrate comparisons for puts. We further note that the parameters of MV are updated daily in this section while the parameters of MV in [90] are updated monthly. We also note that the out-of-sample results for the MV model in [90] are obtained with daily changes of the index price and option price in [90] while, in Table 5.1 and 5.2, the out-of-sample results are obtained with weekly and monthly changes of the index price and option price. The SABR model used to compute the Bartlett delta is also calibrated daily.

Delta	Comparing Model(%)											
	Weekly						Monthly					
	MV	Bartlett	DKL_{SPL}	NN_δ	GRU_c	GRU_δ	MV	Bartlett	DKL_{SPL}	NN_δ	GRU_c	GRU_δ
0.1	26.3	-16.9	38.9	35.6	36.6	**47.8**	13.5	-8. 2	22.7	29.7	34.8	**53.9**
0.2	21.6	-5.6	29.0	36.4	39.6	**48.5**	16.4	0.4	23.5	38.4	38.9	**51.7**
0.3	20.1	11.9	23.5	38.6	39.7	**48.5**	17.9	2.1	24.0	40.2	41.7	**50.2**
0.4	18.1	17.3	20.8	38.7	38.9	**45.9**	16.9	2.7	21.0	38.6	42.6	**47.8**
0.5	16.0	21.7	19.9	42.3	37.5	**46.6**	15.2	5.7	13.5	36.3	42.3	**44.5**
0.6	12.1	24.1	17.3	43.4	33.5	**44.8**	12.7	8.4	14.3	36.0	40.7	**44.6**
0.7	8.1	26.3	16.8	45.6	31.1	43.9	5.9	7.5	6.1	30.2	26.3	**35.3**
0.8	3.7	25.5	12.5	39.6	31.7	**37.7**	-1.2	4.2	5.3	22.3	26.3	**24.8**
0.9	2.4	21.7	6.2	26.3	28.7	16.4	-1.8	9.8	4.1	21.1	17.3	10.5
Overall	15.1	18.6	20.2	39.9	33.5	**43.7**	13.4	4.5	16.3	35.4	38.0	**44.5**

Table 5.1: S&P 500 Call Options Hedging Comparison on Traded Data, bold entries indicating best Gain from GRU_δ

From Table 5.1 and Table 5.2, we observe that, overall, GRU_δ outperforms GRU_c, MV, Bartlett, DKL_{SPL}, and NN_δ. In addition, the hedging performance of the four data-driven models, DKL_{SPL}, NN_δ, GRU_c, and GRU_δ, is significantly better than that of the MV model and Bartlett except for put option weekly hedging where Bartlett correction performs better than DKL_{SPL}. This is not surprising since the MV formula (2.3.4) and Bartlett formula (2.3.16) are based on instantaneous hedging analysis. Furthermore, by comparing DKL_{SPL} with NN_δ, we see that the decoder only NN_δ still achieves significant improvement over DKL_{SPL}. The improvement possibly comes from inclusions of more local features in NN_δ and more frequent updates of NN_δ, timely addressing the market changes. The overall improvement of GRU_δ over NN_δ illustrates the important role of the GRU encoder, which incorporates sequential feature information, in weekly and monthly hedging. Lastly, the overall improvement of GRU_δ over GRU_c illustrates the important role of the output gate, the robust huber loss and the robust training procedure in section 4.2.

Delta	Comparing Model(%)											
	Weekly						Monthly					
	MV	Bartlett	DKL_{SPL}	NN_δ	GRU_c	GRU_δ	MV	Bartlett	DKL_{SPL}	NN_δ	GRU_c	GRU_δ
-0.9	23.9	9.1	10.1	29.5	32.1	**34.7**	16.9	1.2	6.5	28.3	27.4	**32.6**
-0.8	21.5	-0.1	18.3	39.6	40.1	**44.2**	11.5	5.6	6.1	41.7	35.6	**49.5**
-0.7	19.1	0.4	20.2	44.0	40.6	**49.6**	9.6	6.7	7.3	43.4	42.1	**52.4**
-0.6	16.1	9.1	20.8	43.0	43.6	**51.3**	8.1	8.6	10.3	42.1	42.5	**51.6**
-0.5	15.3	20.9	22.4	43.4	41.3	**53.5**	7.7	13.2	13.9	41.2	42.7	**51.4**
-0.4	12.3	25.7	21.0	41.4	45.6	**53.2**	6.8	14.4	15.6	40.7	42.9	**53.4**
-0.3	9.7	29.4	22.2	38.2	44.6	**51.1**	4.7	13.6	19.5	34.1	47.5	**48.4**
-0.2	7.8	33.1	20.8	29.8	40.4	**46.3**	2.9	10.7	20.6	21.7	36.7	**44.7**
-0.1	4.9	30.5	19.2	15.5	36.4	**37.2**	-1.8	10.8	13.0	12.3	24.1	**26.8**
Overall	14.4	26.4	20.4	38.7	42.4	**49.1**	8.6	12.1	13.5	38.6	43.5	**49.5**

Table 5.2: S&P 500 Put Options Hedging Comparison on Traded Data, bold entries indicating best Gain from GRU_δ

5.1.2 Feature Importance

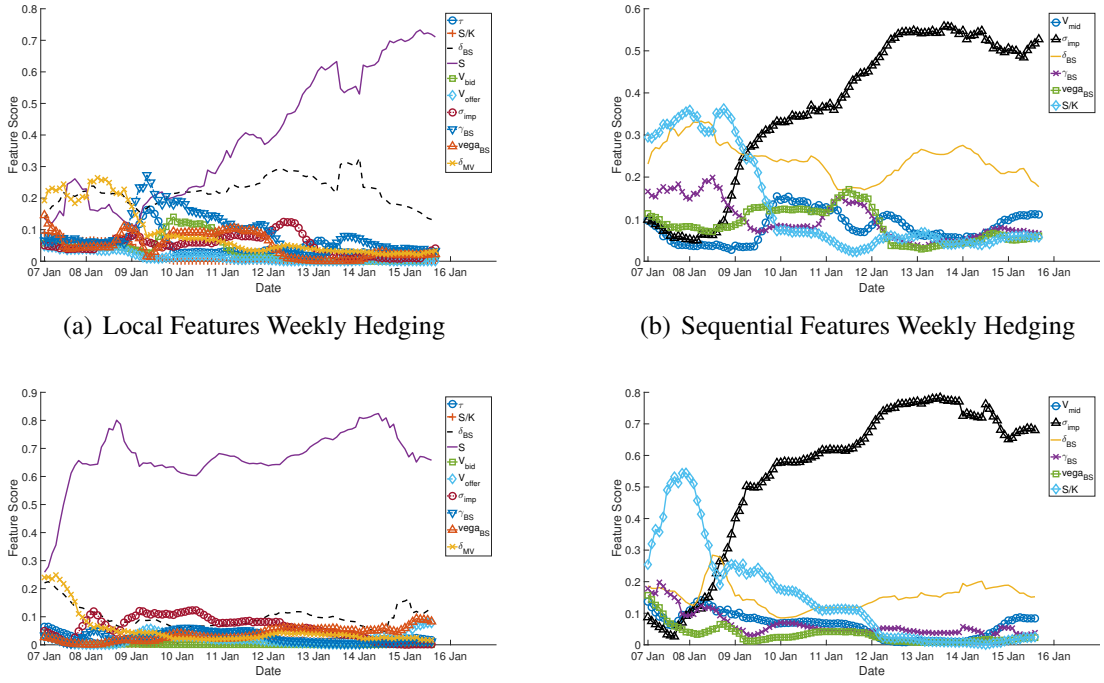
Next, we discuss feature importance in the trained model GRU_δ. Specifically, we show how the normalized feature weight, for both the local features $\mathbf{x}_t^{T,K}$ and for the sequential features $\mathbf{X}$, changes from Jan 2007 to Aug 2015. The normalized feature weight vector are:

$$\frac{exp(\omega^L)}{\sum_{i=1}^{d_l} exp(\omega_i^L)}$$

and

$$\frac{exp(\omega^S)}{\sum_{i=1}^{d_s} exp(\omega_i^S)}$$

respectively. The feature score, shown in Figure 5.1& 5.2, represents the normalized feature weights averaged over a calendar month for the model trained using traded data instance. Note that the model is updated daily and the normalized feature weights vary with the training date.



(a) Local Features Weekly Hedging

(b) Sequential Features Weekly Hedging

(c) Local Features Monthly Hedging

(d) Sequential Features Monthly Hedging

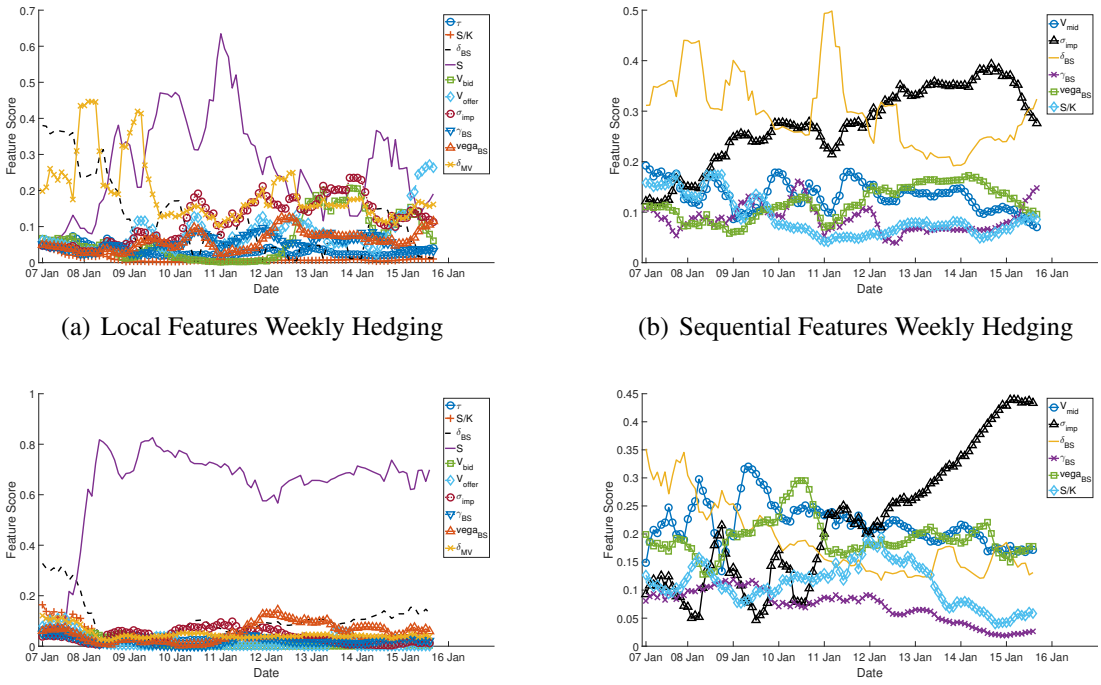
Figure 5.1: Feature Scores for Weekly and Monthly Hedging Models on Traded S&P500 Call Option Data

From subplots (a) and (b) in Figure 5.1, it can be observed that, for weekly hedging call options, the index price S is ranked as the most important local feature after Jan 2010. Moreover, the average weight for the index price S exhibits an increasing trend after Jan 2009. The past implied volatility sequence is ranked as the most important sequential feature after March 2009. The average weight for the past implied volatility sequence also exhibits an increasing trend after Jan 2009. From subplots (c) and (d) in Figure 5.1, it can be observed that, for monthly hedging the call option, the index price S is always ranked as the most important local feature. In addition, the average weight of the index price S is always higher than 0.5 after Jan 2008. The past implied volatility sequence is always ranked as the most important sequential feature after Jan 2009. The average weight for the past implied volatility sequence also is always greater than 0.4 after Jan 2008.

For the put option, the situation is slightly different. From subplots (a) and (b) in Figure 5.2, we can see that, for weekly hedging the put option, the index price S is often ranked as the most important local feature. The past implied volatility sequence and the past BS delta sequence are identified as the two most important sequential features. From subplots (a) and (b) in Figure

5.2, we can see that, for monthly hedging put options, the index price S is ranked as the most important local feature after Jan 2008. The past implied volatility sequence is ranked as the most important sequential feature after Jan 2013.

Overall, the index price S has often been identified as an important local feature for hedging both call and put options. Since the BS delta of the implied volatility is one of the local features, this confirms the fact that the BS delta does not capture all dependence on the underlying In addition, the past implied volatility has often been identified as an important sequential feature for hedging both calls and puts.



(a) Local Features Weekly Hedging

(b) Sequential Features Weekly Hedging

(c) Local Features Monthly Hedging

(d) Sequential Features Monthly Hedging

Figure 5.2: Feature Score for Weekly and Monthly Hedging Models on Traded S&P500 Put Option Data

5.1.3 Daily Hedging Comparison

In section 5.1.1 - 5.1.2, we have have presented hedging comparisons for weekly and monthly hedging. Now we compare the proposed GRU_δ with the following methods for daily hedging,

- MV: MV hedging based on formulation (2.3.4),
- LVF: MV hedging in [90] from local volatility function based on formulation (2.3.12),

- SABR_{MV} : MV hedging in [90] from SABR model based on formulation (2.3.18),
- δ^{BS}: implied volatility Black–Scholes delta,
- Bartlett: Bartlett formula based on (2.3.16),
- DKL_{SPL}: Direct spline kernel hedging method as in 3.2.

We have omitted NN_δ and GRU_c in the daily hedging comparison since it has been shown to underperform GRU_δ. But we have added LVF and SABR_{MV} since they are based on instantaneous hedging analysis and are more suitable for daily hedging.

Since the same S&P 500 index option data from the OptionMetric Database is used here, we simply quote the results presented in [90] for MV, LVF, and SABR_{MV}, and the results presented in [118] for DKL_{SPL}.

As stated in [90] and section 2.3.1, the SABR_{MV} model is calibrated daily, from which the hedge position is determined and applied the next day. For LVF, the partial derivative of the expected implied volatility to the underlying is calculated from the slope of the observed implied volatility smile daily. For MV, the model parameter, a, b and c are estimated using all options in a 36-month-window and then applied to determine the hedging position daily in the subsequent month.

Table 5.3 presents daily hedging comparisons for call options. From Table 5.3, using either traded data or all data for testing, GRU_δ outperforms the minimum variance hedging methods reported in [90] in the overall performance. For most delta buckets, delta greater than or equal to 0.3, the performance of GRU_δ is better than those of the minimum variance hedging methods. However, for the bucket of the delta value 0.1 or 0.2, the performance is slightly worse.

In addition, from Table 5.3, the performance of the GRU_δ model is slightly better than that of the kernel hedging DKL_{SPL} in overall performance. For the the delta buckets 0.3-0.9, GRU_δ outperforms the direct kernel hedging DKL_{SPL}. However, GRU_δ performs slightly worse for the delta bucket 0.1 and 0.2.

Table 5.4 presents daily hedging comparisons for put options. Table 5.4 shows that both overall performance and the performances for different delta buckets of the GRU_δ model are better than those from minimum variance hedging methods in [90] and the direct kernel hedging DKL_{SPL}, for testing result using all data. Similarly to the call, GRU_δ has model bold entries in most of delta buckets, indicating outperforming other methods.

Both GRU_δ and DKL_{SPL} are non-parametric data-driven hedging methods, with a main difference in features included. The hedging performance comparison between the two suggests that including the sequential features and more local features improves the overall hedging performance slightly for daily hedging.

We similarly demonstrate feature importance in Figure 5.3&5.4 for daily hedging. From Figure 5.3, it can be observed that, for daily hedging call options, MV delta δ^{MV} and index price S are the local features that are often ranked as the most important under GRU_δ. The history of

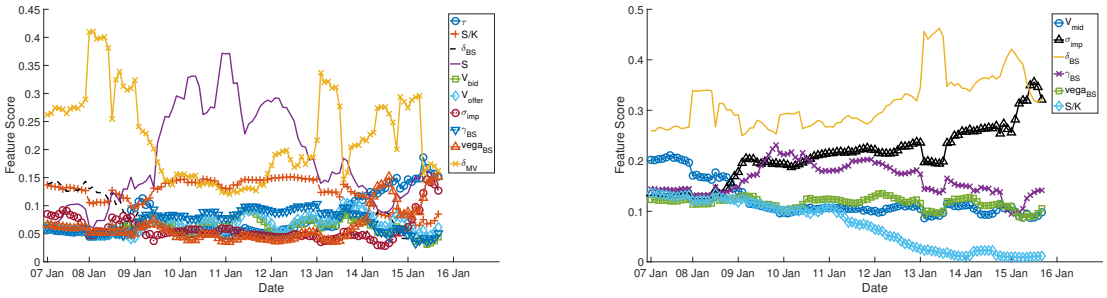
Delta	MV (%)	$SABR_{MV}$(%)	LVF(%)	Bartlett		Data-Driven Model			
				Traded	All	DKL_{SPL} (%)		GRU_{δ} (%)	
						Traded	All	Traded	All
0.1	42.1	39.4	42.6	29.0	35.1	47.1	48.6	32.3	33.8
0.2	35.8	33.4	36.2	28.2	32.3	37.8	40.0	33.7	36.4
0.3	31.1	29.4	30.3	27.7	28.9	34.1	35.1	34.1	**35.5**
0.4	28.5	26.3	26.7	28.7	27.3	32.3	32.0	**33.7**	**34.2**
0.5	27.1	24.9	25.5	26.9	26.7	29.3	29.4	**35.1**	**33.0**
0.6	25.7	25.2	25.2	28.3	26.6	29.9	28.4	**35.6**	**32.1**
0.7	25.4	24.7	25.8	28.5	26.4	29.0	26.8	**31.8**	**29.7**
0.8	24.1	23.5	25.4	23.1	24.9	25.9	24.7	**28.6**	**26.5**
0.9	16.6	17.0	16.9	14.0	15.6	17.7	13.9	**19.3**	**18.9**
Overall	25.7	24.6	25.5	27.1	24.8	31.3	26.0	**32.9**	**28.7**

Table 5.3: S&P 500 Call Option Hedging For 1-Business Day: bold entries indicating best Gain from GRU_{δ}

Delta	MV (%)	$SABR_{MV}$(%)	LVF(%)	Bartlett		Data-Driven Model			
				Traded	All	DKL_{SPL} (%)		GRU_{δ} (%)	
						Traded	All	Traded	All
-0.9	15.1	11.2	-7.4	9.1	23.4	8.6	13.6	**15.1**	**17.2**
-0.8	18.7	19.6	6.8	3.2	21.9	6.5	16.7	**23.2**	**28.5**
-0.7	20.3	17.7	9.1	1.5	20.1	10.6	19.8	**28.5**	**32.8**
-0.6	20.4	16.7	9.2	6.1	19.2	14.9	21.0	**28.3**	**33.9**
-0.5	22.1	16.7	10.8	15.5	21.3	22.5	23.1	**29.2**	**34.5**
-0.4	23.8	17.7	12.0	21.0	24.4	24.2	25.2	**29.9**	**34.7**
-0.4	27.1	21.7	16.8	26.7	29.0	27.7	28.3	**30.6**	**33.6**
-0.2	29.6	25.8	20.6	29.3	31.6	30.1	30.8	25.4	29.9
-0.1	27.5	26.9	17.7	31.4	32.5	29.1	31.2	18.7	21.4
Overall	22.5	19.0	10.2	20.0	24.8	23.4	23.2	**26.2**	**29.7**

Table 5.4: S&P 500 Put Option Hedging For 1-Business Day: bold entries indicating best Gain from GRU_{δ}

BS delta δ^{BS} is ranked as the most important sequential feature during most of the months. The past implied volatility sequence is often ranked as the second most important sequential feature.

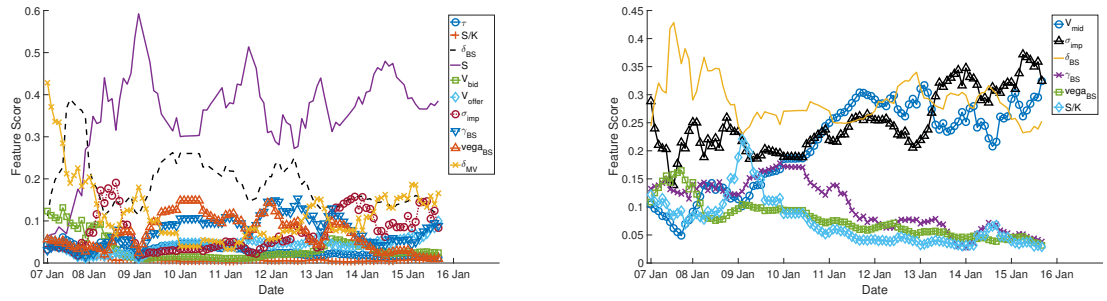


(a) Local Features Daily Hedging

(b) Sequential Features Daily Hedging

Figure 5.3: Feature Score for Daily Hedging Model: S&P500 Call Option (Traded Data)

From subplots (a) and (b) in Figure 5.4, it can be observed that the index price S is often ranked as the most important local feature for daily hedging put options. The past implied volatility sequence, the past BS delta sequence, and the past option price V_{mid} are often identified as the three important sequential features.



(a) Local Features Daily Hedging

(b) Sequential Features Daily Hedging

Figure 5.4: Feature Score for Daily Hedging Model: S&P500 Put Option (Traded Data)

5.2 Concluding Remarks for Data-Driven Local Hedging Models

In this thesis, we have proposed a direct kernel hedging model DKL_{SPL} and a novel encoder-decoder model, GRU_{δ} , for discrete local risk option hedging. The DKL_{SPL} is our preliminary exploration on computing a data-driven local hedging model without estimating a option pricing model. The GRU_{δ} is our enhancement to data-driven local hedging model. GRU_{δ} consists of an

encoder, which generates a concise representation of the past market information. The decoder uses the Black-Scholes delta as a pre-trained model and utilizes a gate to generate a predicative hedging model, combining the pre-trained delta model and the outputs from the encoder. Feature selections are implemented through normalized weights embedded in the model training. In addition, a data instance adaptive Huber loss function is incorporated for robustness, with the error from the pre-trained Black-Scholes delta model for each instance as the thresholding parameter.

Using the S&P 500 index and the index option data, from January 1, 2004, to August 31st, 2015, we assess and compare hedging performance of the GRU_δ and DKL_{SPL} with other hedging strategies. For weekly and monthly hedging, computational results demonstrate that performance of the proposed GRU_δ significantly surpasses that of the MV model , SABR-Bartlett, regularized spline kernel model DKL_{SPL} , all of which perform significantly better than the Black–Scholes model with implied volatility. In addition, the DKL_{SPL} also outperforms MV model in terms of weekly and monthly hedging results. We further demonstrate that the encoder for the sequential features plays a significant role in GRU_δ, since removing the encoder deteriorates hedging performance. Lastly, by comparing the weekly and monthly hedging performance from GRU_c and GRU_δ, we demonstrate that the output gate, robust loss and training procedure, also play a significant role in GRU_δ.

We further demonstrate that the daily hedging performance of the proposed GRU_δ also surpasses that of the MV hedging method , LVF and SABR corrective methods (implemented in [90]), data-driven regularized spline kernel network model DKL_{SPL} , and SABR-Bartlett. In addition, DKL_{SPL} also outperforms MV hedging method , LVF and SABR corrective methods (implemented in [90]).

Besides, from the testing hedging performance, we assess feature importance in GRU_δ for the S&P 500 index option hedging. The monthly average feature weights identify the underlying as the most important local feature and the past implied volatility sequence as the most important sequential feature during normal market periods.

Overall, our research demonstrates the potential role of a market data-driven approach for financial derivative risk management.

A natural extension is to extend the sequential local hedging model to be a total hedging model where we rebalance multiple times until the expiries of the options. In the following chapter 6.4, we discuss the challenges associated with building such models and present our exploration on the data-driven total hedging model.

Chapter 6

Data-Driven Sequential Learning Framework for Total Hedging Risk

As we have discussed in section 2.2, in reality, one often want to hedge until the expiry of the option. Enhancing the local hedging model GRU_δ to output the multiple hedging positions by minimizing a total hedging objective is quite attractive. From the model perspective, such enhancement is achievable. Indeed, recently a deep hedging model was proposed to minimize the total option hedging risk evaluated at the option expiry [24]. It is shown that using a RNN to represent the hedging position model can be a computationally efficient framework to determine the optimal hedging function when the market is incomplete, e.g., under the transaction cost [24].

Although minimizing the total hedging risk, which is hedging portfolio value at the expiry T, is more desirable. There are several major obstacles in obtaining enough market data to build a data-driven total hedging model. The deep hedging model [24] is built on synthetic data. It is hard to directly apply their methodology in real world due to the lack of market data needed to build a sufficient model.

In this chapter, we provide several achievable remedies to deal with issue of lack of market data. A enhanced sequential total hedging model $\text{GRU}_{\text{TOTAL}}$ is introduced to minimize a discrete total hedging objective. We then build the total hedging model $\text{GRU}_{\text{TOTAL}}$ based on the remedies we proposed and demonstrate the effectiveness of each of the remedies using real market data experiments.

6.1 Testing with Synthetic Data

We have conducted primitive experiments with the Black-Scholes Model, Heston Model, GARCH model.

The input is $\mathbf{y}_{t_i} = \{S_{t_i}/K, t, \delta_{BS}^{t_i}\}$. The previous output from the previous rebalancing time is also added as one of the input. In terms of performance, the performance of the model is

also similar to the spline model proposed in [40] and the analytical solution of the total risk minimization. The model we proposed is the simple model in Figure 3. The model we proposed is the simple model in Figure 3. The evaluation criteria are same as in [40]:

- Total risk: Average of the absolute value of the final portfolio value:

$$\begin{aligned} |P_{t_N^-}| &= \left| e^{-rT} \left(e^{r\Delta t} B_{t_{N-1}} - V_{t_N} + S_{t_N} \delta_{t_{N-1}} \right) \right| \\ &= \left| e^{-rT} \left(\sum_{j=0}^{N-1} \left\{ \left[e^{r(N-j-1)\Delta t} S_{t_{j+1}} - e^{r(N-j)\Delta t} S_{t_j} \right] \delta_{t_j} \right\} + e^{rN\Delta t} V_{t_0} - V_{t_N} \right) \right| \end{aligned}$$

- Total cost: Average of the total cost in rebalancing the portfolio:

$$e^{-rT} \left(V_{t_N} - \sum_{j=0}^{N-1} \left\{ \left[e^{r(N-j-1)\Delta t} S_{t_{j+1}} - e^{r(N-j)\Delta t} S_{t_j} \right] \delta_{t_j} \right\} \right)$$

6.1.1 Black-Scholes Model

We adopt the same experimental setting as [40]. Specifically, we assume $S_0 = 100$,$K = 100$, $T = 1$, $\mu = 0.15$,$r = 0.04$, and $\sigma = 0.2$. We have 600 time steps with $\Delta t = 1/600$. We can hedge every 25,50,100,300 time steps.

method	25	50	100	300
GRU	0.8448	1.1845	1.6518	2.7989
GRU (Refined)	0.8376	1.1426	1.6896	2.8038
Spline	0.8563	1.1789	1.6518	2.7843
Analytical	0.8295	1.1636	1.6479	2.7914
Delta	0.9481	1.3385	1.9128	3.4582

Table 6.1: Total Risk

method	25	50	100	300
GRU	5.8943	5.8812	5.7378	5.2507
GRU (Refined)	5.9682	5.8370	5.8549	5.1759
Spline	5.9118	5.8445	5.7119	5.2530
Analytical	5.9413	5.8773	5.7399	5.2565
Delta	6.0483	6.0897	6.1734	6.5382

Table 6.2: Total Cost

6.2 Data Augmentation for Market Option Data

Recall that the discrete total hedging risk (2.2.8) is defined as the the hedging portfolio value at expiry:

$$P_H(T,T,K) = \sum_{j=0}^{N-1} \left\{ \left[e^{r(N-j-1)\Delta t} S_{t_{j+1}} - e^{r(N-j)\Delta t} S_{t_j} \right] \delta_{t_j,T,K} \right\} + e^{rN\Delta t} V^{mkt}_{t_0,T,K} - V^{mkt}_{T,T,K}$$

Suppose we have M hedging scenarios, a data driven model can be built by minimizing total hedging loss function such as MSE:

$$MSE_{total} = \sum_{i=1}^{M} P_H(T_i,T_i,K_i)^2$$

Clearly, in order to build a efficient data-driven total hedging model, one needs to gather a sufficiently large number of sample M. In addition, ideally, one can also use non-overlapping underlying asset paths to generate training and testing hedging scenarios. However, in reality, we have the following challenges:

1. Using non-overlapping underlying asset path to generate training and testing scenarios is not practical. For instance, assuming we are building a data-driven model for hedging 3 months until expiry, with non-overlapping underlying asset paths, 20 years of market data can provide only 80 different non-overlapping underlying asset paths which is clearly not enough to build a data-driven model.

2. Option data in real market is limited.

   (a) Options listed in exchange in real market only have fixed expiry dates. For example, the expiration date for S&P 500 index option is fixed at the third Friday of each month. Therefore, if we only use market available expiration dates, the number of training scenarios will also be severely limited.

   (b) In addition, option with specific strike K and expiry T is not traded on every data. Even we use market available expiration dates, on the initial date when we want to set up the hedging portfolio, we may not have the initial market option price $V^{mkt}_{t_0,T,K}$.

In order to overcome the above challenges, we propose the following solution.

1. Overlapping underlying asset path is allowed to be used.

2. Instead of using only market available expiration dates, we assume every business day can be the expiration dates of the options.

3. A no-arbitrage price surface is calibrated every day t to obtain option price for any arbitrary combination of K and T: $V^{Aug}_{t,T,K}$.

Therefore, we can greatly increase the number of training scenarios, since now the hedging scenarios can have arbitrary combination of T and K even if the combination of T and K is not observed in the market.

6.2.1 No-Arbitrage Price Surface

In this section, we describe how to calibrate an arbitrage-free price surface on each business day. We want to create a price surface that is **arbitrage-free**. In addition, we want to able to **extrapolate** for deep in-money options and deep out-of-money options. Let $Surf_t(T,K)$ be the option price surface function at time t. We want to make sure that the price surface satisfies the following no-arbitrage conditions:

1. No call spread arbitrage or put spread arbitrage:

$$-1 \leq \frac{\partial Surf_t(T,K)}{\partial K} \leq 0 \text{ (Call)}$$

or

$$0 \leq \frac{\partial Surf_t(T,K)}{\partial K} \leq 1 \text{ (Put)}$$

2. No butterfly spread arbitrage:

$$\frac{\partial^2 Surf_t(T,K)}{\partial K^2} \geq 0$$

3. No calendar spread arbitrage:

$$\frac{\partial Surf_t(T,K)}{\partial T} \geq 0$$

In the following subsection, we will discuss the steps of calibration of the arbitrage-free surface.

SABR Interpolation and Extrapolation

In this thesis, we will use SABR model to obtain the option price for arbitrary strike K unobserved in the market. On each business day t, we have observed option quotes for a set of market observed expires. We denote the market available expires as $T_1 < \cdots < T_{max}$. We thus have a grid of expires $t = T_0 < T_1 < \cdots < T_{max}$ where we have market quotes for option (Note for for T_0, the market option price is just the option payoff at t).

We firstly use SABR model to obtain option price for a predetermined grid of strike $0 = K_0 < K_1 < \cdots < K_{max}$. More specifically, for a fixed expiry T_i, let us define

$$K^{mkt} = \{\mathbb{K}^{mkt}_1, \ldots, K^{mkt}_{N_K}\}$$

to be the strikes for which market option price is available for T_i. We solve

$$\min_{\alpha,\beta,\nu,\rho} \sum_{j=1}^{N_K} \left(V_{SABR}(S,t,T_i,K_j^{mkt},r,q;\alpha,\beta,\nu,\rho) - V^{mkt}(t,T_i,K_j^{mkt}) \right)^2$$

where $V_{SABR}(S,t,T,K,r,q;\alpha,\beta,\nu,\rho)$ is option pricing function described in section 2.1.3.

For each T_i, we calibrate a separate set of SABR parameters and we then use the calibrated SABR parameters to compute SABR price of the predetermined grid of strikes for each T_i. The SABR prices of the predetermined grid of strikes are used as the calibration targets for the volatility interpolation procedure described in the following subsection.

Although SABR model is computationally efficient and can match the market volatility smile well, it is not arbitrage-free. Butterfly and call spread arbitrage can exist. In addition, for each expiry, a separate set SABR parameters are calibrated so calender spread arbitrage can also exist. Therefore, before using the SABR price as the target for the LVF calibration, we need to fix the potential arbitrage.

In this thesis, we use the sufficient conditions for no arbitrage [28] to check whether arbitrage exists for the grid of strikes: $0 = K_0 < K_1 < \cdots < K_{max}$, and the grid of expiries $t = T_0 < T_1 < \cdots < T_{max}$. If arbitrage is detected, we will manually adjust the option prices from SABR model to satisfy the no-arbitrage condition.

Volatility Interpolation

Andreasen and Huge [5] have introduced an efficient and arbitrage free volatility interpolation method based on a one step finite difference implicit Euler scheme applied to a local volatility parametrization. In this thesis, we use the volatility interpolation approach to compute option price for arbitrary expiry T unobserved in market.

The volatility interpolation method is based on the Dupire formula. The Dupire enables us to deduce the volatility function in a local volatility model from quoted put and call options in the market. Under a risk-neutral measure, we assume:

$$\frac{dS_t}{S_t} = (r-q)\,dt + \sigma(t,S_t)dZ_t$$

Let C be the call option pricing function [1]. Dupire's equation states:

$$\frac{\partial C}{\partial T} = \frac{1}{2}\sigma^2(T,K)K^2\frac{\partial^2 C}{\partial K^2} - (r-q)K\frac{\partial C}{\partial K} - qC$$

Define the normalized call price in terms of discount factor $D(t,T)$ and forward price $F(t,T)$

[1] In the following discussion, we use call option as the example but the analysis also holds for put options.

and the normalized strike $\widehat{K}$ as:

$$
\begin{aligned}
D(t,T) &= e^{-r(T-t)} \\
F(t,T) &= S_t e^{(r-q)(T-t)} \\
\widehat{K} &= \frac{K}{F(t,T)} \\
\widehat{C}(T,\widehat{K}) &= \frac{C(T,\widehat{K}F(t,T))}{D(t,T)F(t,T)} = \frac{C(T,K)}{D(t,T)F(t,T)}
\end{aligned}
$$

Dupire's equation can be simplified as:

$$
\frac{\partial \widehat{C}}{\partial T} = \frac{1}{2}\widehat{\sigma}^2(T,\widehat{K})\widehat{K}^2\frac{\partial^2 \widehat{C}}{\partial \widehat{K}^2}, \;\; \widehat{\sigma}(T,\widehat{K}) = \sigma(T,K)
$$

We recursively solve the finite difference discretization of the Dupure forward equation using fully implicit method:

$$
\frac{\partial \widehat{C}}{\partial T} = \frac{1}{2}\widehat{\sigma}^2(T,\widehat{K})\widehat{K}^2\frac{\partial^2 \widehat{C}}{\partial \widehat{K}^2}
$$

with initial condition $\widehat{C}(T_0,\widehat{K}) = max(1-\widehat{K},0)$, lower boundary condition $\widehat{C}(T,0) = 1$, upper boundary condition $\widehat{C}(T,\widehat{K}_{max}) = 0$. Given a grid of expiries $0 = T_0 < T_1 < \cdots < T_{max}$ and the grid of normalized strike: $0 = \widehat{K}_0 < \widehat{K}_1 < \cdots < \widehat{K}_{max}$, we assume $\sigma(T_i,K)$ to be a piecewise constant functions for a given T_i.

$$
\sigma(T_i,K) = \begin{cases}
\sigma_{T_i,K_0}, & \text{if } K \le K_0 \\
\vdots & \vdots \\
\sigma_{T_i,K_j}, & \text{if } K_{j-1} < K \le K_j \\
\vdots & \vdots \\
\sigma_{T_i,K_Max}, & \text{if } K_{max} \ge K
\end{cases}
$$

We further assume:

$$
\frac{\partial^2 \widehat{C}(T,0)}{\partial \widehat{K}^2} = \frac{\partial^2 \widehat{C}(T,\widehat{K}_{max})}{\partial \widehat{K}^2} = 0
$$

The fully implicit finite difference method in matrix form is:

$$
\begin{bmatrix}
\widehat{C}(T_i,\widehat{K}_0) \\
\widehat{C}(T_i,\widehat{K}_1) \\
\widehat{C}(T_i,\widehat{K}_2) \\
\vdots \\
\widehat{C}(T_i,\widehat{K}_{max})
\end{bmatrix}
= \mathbb{M}
\begin{bmatrix}
\widehat{C}(T_{i+1},\widehat{K}_0) \\
\widehat{C}(T_{i+1},\widehat{K}_1) \\
\widehat{C}(T_{i+1},\widehat{K}_2) \\
\vdots \\
\widehat{C}(T_{i+1},\widehat{K}_{max})
\end{bmatrix}
$$

where $\mathbb{M}$ is a tri-diagonal matrix parametrized by $\sigma(T_i,.)$

$$\mathbb{M}^{-1}\begin{bmatrix}\widehat{C}(T_i,\widehat{K}_0)\\ \widehat{C}(T_i,\widehat{K}_1)\\ \widehat{C}(T_i,\widehat{K}_2)\\ \vdots\\ \widehat{C}(T_i,\widehat{K}_{max})\end{bmatrix}=\begin{bmatrix}\widehat{C}(T_{i+1},\widehat{K}_0)\\ \widehat{C}(T_{i+1},\widehat{K}_1)\\ \widehat{C}(T_{i+1},\widehat{K}_2)\\ \vdots\\ \widehat{C}(T_{i+1},\widehat{K}_{max})\end{bmatrix}$$

We try to find $\mathbb{M}$ that so that $\widehat{C}(T_{i+1},\widehat{K}_j)=\widehat{C}_{SABR}(T_{i+1},\widehat{K}_j)$. Where

$$\widehat{C}_{SABR}(T_{i+1},\widehat{K}_j)=\frac{C_{SABR}(T_{i+1},K_j)}{D(t,T_{i+1})F(t,T_{i+1})}$$

$$K_j=\widehat{K}_jF(t,T_{i+1})$$

and $C_{SABR}(T_{i+1},K_j)$ is the SABR price for the grid point (T_{i+1},K_j).

Given $\widehat{C}(T_i,\widehat{K}_j)$, This can be done by:

$$\inf_{\sigma(T_i,.)}\sum_j\left(\frac{\widehat{C}(T_{i+1},\widehat{K}_j)-\widehat{C}_{SABR}(T_{i+1},k_j)}{Vega_B(T_{i+1},\widehat{K}_j)}\right)^2 \tag{6.2.1}$$

Note that $\widehat{C}(T_0,\widehat{K})=max(1-\widehat{K},0)$ is given, we can therefore recursively solve the optimization problem for $T_1,\ldots,T_{max}$.

After optimization, the local volatility functions are translated into arbitrage-consistent prices for a discrete set of expiries but it does not directly specify the option prices between the expiries. For $T\in[T_i,T_{i+1})$, we can fill in the gaps by:

$$\begin{bmatrix}\widehat{C}(T,\widehat{K}_0)\\ \widehat{C}(T,\widehat{K}_1)\\ \widehat{C}(T,\widehat{K}_2)\\ \vdots\\ \widehat{C}(T,\widehat{K}_{max})\end{bmatrix}=\mathbb{M}^{-1}\begin{bmatrix}\widehat{C}(T_i,\widehat{K}_0)\\ \widehat{C}(T_i,\widehat{K}_1)\\ \widehat{C}(T_i,\widehat{K}_2)\\ \vdots\\ \widehat{C}(T_i,\widehat{K}_{max})\end{bmatrix}$$

where $\mathbb{M}$ is a tri-diagonal matrix parametrized by $\sigma(T_i,.)$. Note that $\mathbb{M}$ is known after the calibration (6.2.1). We then recover the call price by:

$$C(T,K)=\widehat{C}(T,\widehat{K})D(t,T)F(t,T)$$

Interpolation based on the above procedure can guarantee the option prices computed is arbitrage-free. Detailed proof can be found in [5]. To sum up, with the help of SABR model and the volatility interpolation procedure, we can compute the arbitrage-free option price for any arbitrary T for the predetermined grid of strikes $0=K_0<K_1<\cdots<K_{max}$.

6.3 Proposed Model For Synthetic Experiments

In this section, we propose a model that is mainly used for synthetic cases. We assume that we are rebalance the hedging portfolio at discrete time periods.

For a rebalance time steps, t_i, we assume the following features are available:

- Local features: $\mathbf{y}_{t_i}$ which records the information for current time steps. Typical features can the: the current underlying S_{t_i}. time to expiry τ, and moneyness S_{t_i}
- Sequential features $\mathbf{X}_{t_i} = \{x_1^{t_i}, \ldots, x_K^{t_i}\}$: which records information in between the two adjacent rebalancing time periods. For example, if we rebalance every 20 business days. Then the time difference between two adjacent rebalancing time periods is $\Delta t = t_i - t_{i-1} = 20/250$. $\mathbf{X}_{t_i} = \{\mathbf{x}_1^{t_i}, \ldots, \mathbf{x}_K^{t_i}\}$ will record the price changes of underlying of the 20 business days. In this case K=20. At the initial rebalance time step t_0, X_{t_0} is the sequential feature set recording the past history before t_0.

We again have a encoder to encode the information contained in X_{t_i} as $\widehat{\mathbf{h}}_E^{t_i}$. The overall structure is similar to what we have for local hedging cases. However, the decoder now is a RNN network too. The hidden states from previous relancing time $c_{t_{i-1}}$ is passed to current relancing time. The initial option price is computed in the following ways:

- V_{t_0} can be supplied as one the parameter and optimized together with the parameters in the hedging position function.
- V_{t_0} can be given by another NN or RNN model based on features we supplied. The option pricing function is optimized together with the the hedging position function.

For synthetic data, we can fix the dynamic of the underlying and generate training and testing paths based on the same dynamic. We then train and test the total hedging performance of the model. We will compare the performance of the model with the performance of using analytical delta and BS delta with implied volatility. Furthermore, we will test the model with and without the hedging encoder.

In [40], the authors proposed that they can achieve better or similar hedging performance using quadratic or linear risk measures. At each rebalancing time, there is a separate function computing the hedging position based on the current asset price S_{t_i} and past history $S_{t_i}^H = [S_{t_0}, \ldots, S_{t_{i-1}}]$:

$$\delta_{t_i} = f_{t_i}(S_{t_i}, S_{t_i}^H)$$

The current proposed model is different from the model in [40] in the following ways:

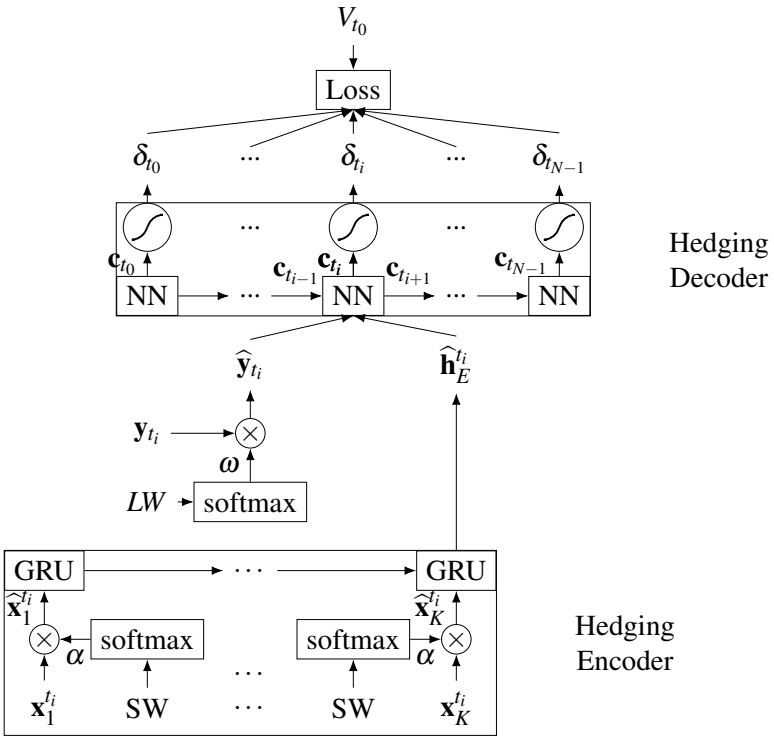


Figure 6.1: Proposed Model For Synthetic Cases

1. Hedging position not only depends on the asset price. The feature we includes can contain many other features

2. A single function is used for all hedging periods. That is to say:

$$\delta_{t_i} = f(\mathbf{X}_{t_j}, \mathbf{y}_{t_j})$$

3. Note that, time to expire is also one of the features, so potentially, $f(\mathbf{X}_{t_j}, \mathbf{y}_{t_j})$ can replace the roles of $f_{t_i}(S_{t_i}, S_{t_i}^H)$.

From [40], we can already know the importance of using the entire path of stock instead of the current stock in determining the hedging position. Even the analytical formula for quadratic formulation clearly demonstrate the dependence of the current hedging position on the past information. Therefore, we want to demonstrate the following things:

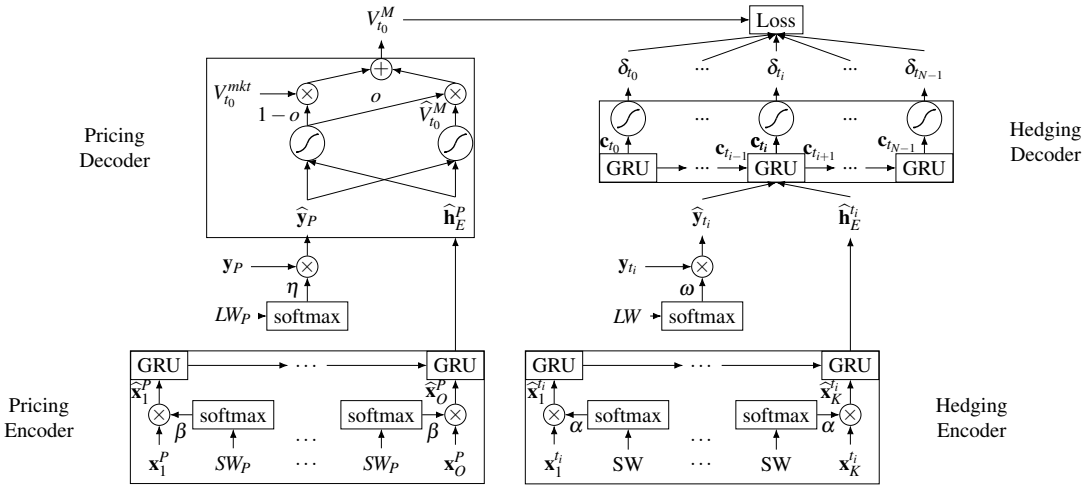


Figure 6.2: Refined Model For Real Cases

6.4 Overview of the Discrete Total Hedging Model

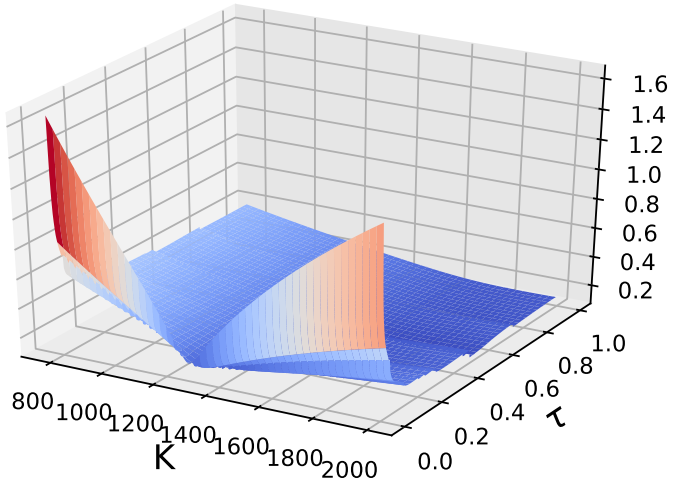
6.4.1 Synthetic Exploration

6.5 Data-Driven Total Discrete Hedging Model and the Different Choices of Features

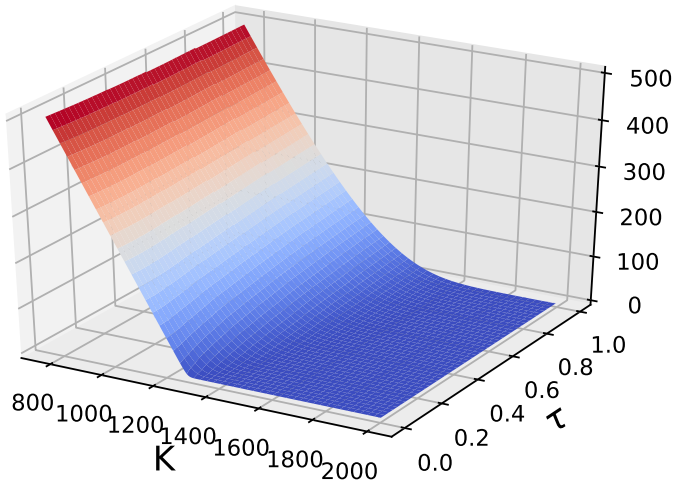
6.6 Total Discrete Hedging Performance Comparison Using S&P 500 index Options

method	Mean Abs	VaR95	CVaR95	VaR99	CVaR
BS	0.3900	0.5321	1.1404	1.3621	1.9858
Bartlett	0.3200	0.4137	0.8702	1.068	1.9219
Local	0.3042	0.3063	0.6233	0.7542	1.3794
Total	0.2730	0.2698	0.6210	0.7498	1.2501

Table 6.3: Total Risk Monthly



(a) Implied Volatility Surface



(b) Price Surface

Figure 6.3: The Price Surface and Implied Volatility Surface for 2012-01-04 SP500 index option

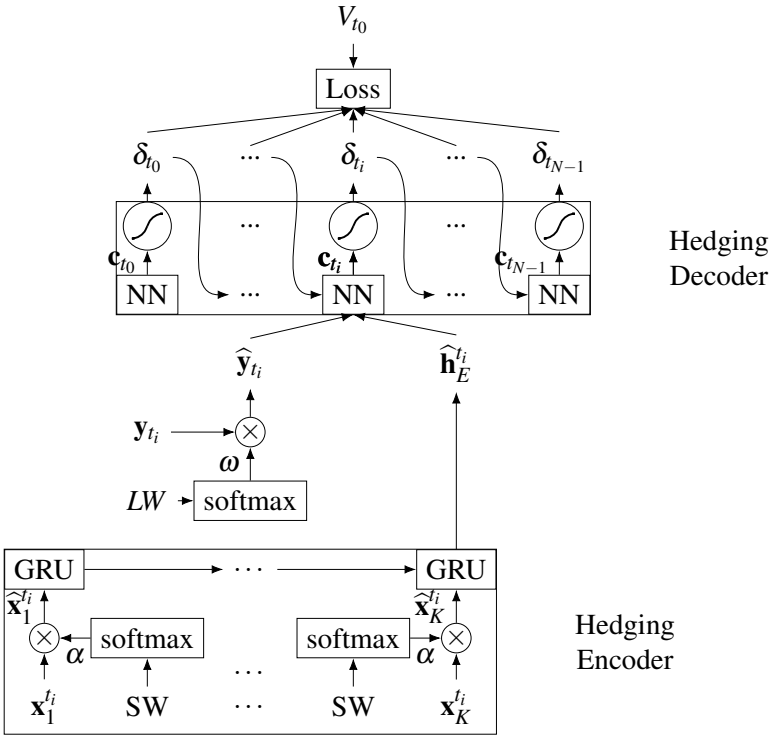


Figure 6.4: Planned Model

method	Mean Abs	VaR95	CVaR95	VaR99	CVaR99
BS	0.2820	0.4233	0.7325	0.9397	1.2573
Bartlett	0.2575	0.4159	0.7911	1.0114	1.4209
Local	0.2785	0.2661	0.5652	0.6613	1.1151
Total	0.2255	0.3243	0.6838	0.8266	1.2011

Table 6.4: Total Risk Weekly

Chapter 7

Conclusion and Future Work

References

[1] Yacine Aıt-Sahalia and Jefferson Duarte. Nonparametric option pricing under shape restrictions. *Journal of Econometrics*, 116(1):9–47, 2003.

[2] Yacine Aït-Sahalia and Andrew W Lo. Nonparametric estimation of state-price densities implicit in financial asset prices. *The Journal of Finance*, 53(2):499–547, 1998.

[3] C. Alexander, A. Kaeck, and L. M. Nogueira. Model risk adjusted hedge ratios. *Journal of Futures Markets*, 29(11):1021–1049, 2009.

[4] Leif Andersen and Rupert Brotherton-Ratcliffe. The equity option volatility smile: an implicit finite-difference approach. *The Journal of Computational Finance*, 1(2):5–32, 1998.

[5] Jesper Andreasen and Brian Norsk Huge. Volatility interpolation. *Available at SSRN 1694972*, 2010.

[6] F. Angelini and S. Herzel. Measuring the error of dynamic hedging: a laplace transform approach. *Journal of Computational Finance*, 13:47–72, 2009.

[7] F. Angelini and S. Herzel. Explicit formulas for the minimal hedging strategy in a martingale case. *Decisions in Economics and Finance*, 33:63–79, 2010.

[8] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[10] Gurdip Bakshi, Charles Cao, and Zhiwu Chen. *The Journal of finance*, 52(5):2003–2049, 1997.

[11] Gurdip Bakshi, Charles Cao, and Zhiwu Chen. *The Journal of finance*, 52(5):2003–2049, 1997.

[12] Gurdip Bakshi, Charles Cao, and Zhiwu Chen. Empirical performance of alternative option pricing models. *The Journal of finance*, 52(5):2003–2049, 1997.

[13] Gurdip Bakshi, Charles Cao, and Zhiwu Chen. *The Journal of finance*, 52(5):2003–2049, 1997.

[14] Gurdip Bakshi, Charles Cao, and Zhiwu Chen. *The Journal of finance*, 52(5):2003–2049, 1997.

[15] Bruce Bartlett. Hedging under sabr model. *Wilmott magazine*, 4:2–4, 2006.

[16] Julia Bennell and Charles Sutcliffe. Black–scholes versus artificial neural networks in pricing ftse 100 options. *Intelligent Systems in Accounting, Finance and Management*, 12 (4):243–260, 2004.

[17] Fischer Black. The pricing of commodity contracts. *Journal of financial economics*, 3 (1-2):167–179, 1976.

[18] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The journal of political economy*, pages 637–654, 1973.

[19] Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

[20] Tim Bollerslev, Julia Litvinova, and George Tauchen. Leverage and volatility feedback effects in high-frequency data. *Journal of Financial Econometrics*, 4(3):353–384, 2006.

[21] Matthew Brand. Fast online svd revisions for lightweight recommender systems. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 37–46. SIAM, 2003.

[22] Mark Broadie, Jérôme Detemple, Eric Ghysels, and Olivier Torrés. American options with stochastic dividends and volatility: A nonparametric investigation. *Journal of Econometrics*, 94(1):53–92, 2000.

[23] Mark Broadie, Jérôme Detemple, Eric Ghysels, and Olivier Torrès. Nonparametric estimation of american options' exercise boundaries and call prices. *Journal of Economic Dynamics and Control*, 24(11):1829–1857, 2000.

[24] Hans Buehler, Lukas Gonon, Ben Wood, Josef Teichmann, Baranidharan Mohan, and Jonathan Kochems. Deep hedging: Hedging derivatives under generic market frictions using reinforcement learning-machine learning version. *Available at SSRN*, 2019.

[25] Richard H Byrd, Robert B Schnabel, and Gerald A Shultz. A trust region algorithm for nonlinearly constrained optimization. *SIAM Journal on Numerical Analysis*, 24(5):1152–1170, 1987.

[26] P. Carr. European put call symmetry. pages 509–537. Cornell University working paper, 1994.

[27] P. Carr, K. Ellis, and V. Gupta. Static hedging of exotic options. *Journal of Finance*, 53: 1165–1190, 1998.

[28] Peter Carr and Dilip B Madan. A note on sufficient conditions for no arbitrage. *Finance Research Letters*, 2(3):125–130, 2005.

[29] Xiaohong Chen. Large sample sieve estimation of semi-nonparametric models. *Handbook of econometrics*, 6:5549–5632, 2007.

[30] Xiaohong Chen, Jeffrey Racine, and Norman R Swanson. Semiparametric arx neural-network models with an application to forecasting inflation. *IEEE Transactions on neural networks*, 12(4):674–683, 2001.

[31] Xilun Chen and K Selcuk Candan. Lwi-svd: low-rank, windowed, incremental singular value decompositions on time-evolving data sets. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 987–996. ACM, 2014.

[32] Shunfeng Cheng and Michael Pecht. Using cross-validation for model parameter selection of sequential probability ratio test. *Expert Systems with Applications*, 39(9):8467–8473, 2012.

[33] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[34] Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Transactions on Multimedia*, 17 (11):1875–1886, 2015.

[35] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[36] T. F. Coleman, Y. Li, and M. Patron. Total risk minimization using Monte-carlo simulations. In John R. Birge and Vadim Linetsky, editors, *Handbook of Financial Engineering*. Elsevier, 2007.

[37] Thomas F. Coleman, Yuying Li, and Arun Verma. Reconstructing the unknown local volatility function. *The Journal of Computational Finance*, 2(3):77–102, 1999.

[38] Thomas F Coleman, Yohan Kim, Yuying Li, and Arun Verma. Dynamic hedging with a deterministic local volatility function model. *The Journal of Risk*, 5(6):63–89, 2001.

[39] Thomas F. Coleman, Yuying Li, and M. Patron. Discrete hedging under piecewise linear risk minimization. *The Journal of Risk*, 5:39–65, 2003.

[40] Thomas F Coleman, Yuying Li, and Maria-Cristina Patron. Total risk minimization using monte carlo simulations. *Handbooks in Operations Research and Management Science*, 15:593–635, 2007.

[41] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. Trading convexity for scalability. In *Proceedings of the 23rd international conference on Machine learning*, pages 201–208. ACM, 2006.

[42] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

[43] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3): 273–297, 1995.

[44] John C Cox, Jonathan E Ingersoll Jr, and Stephen A Ross. A theory of the term structure of interest rates. In *Theory of valuation*, pages 129–164. World Scientific, 2005.

[45] S. Crépey. Delta-hedging vega risk. *Quantitative Finance*, 4(October):559–579, 2004.

[46] Toby Daglish, John Hull, and Wulin Suo. Volatility surfaces: theory, rules of thumb, and empirical evidence. *Quantitative Finance*, 7(5):507–524, 2007.

[47] E. Derman and I. Kani. Riding on a smile. *Risk*, 7:32–39, 1994.

[48] E. Derman, I. Kani, and J. Zou. The local volatility surface: Unlocking the information in index option prices. *Financial Analysts Journal*, pages 25–36, 1996.

[49] J Duan, Genevieve Gauthier, J Simonato, and Caroline Sasseville. Approximating the gjr-garch and egarch option pricing models analytically. *Journal of Computational Finance*, 9(3):41, 2006.

[50] Jin-Chuan Duan. The garch option pricing model. *Mathematical finance*, 5(1):13–32, 1995.

[51] Jin-Chuan Duan, Genevieve Gauthier, and Jean-Guy Simonato. *An analytical approximation for the GARCH option pricing model*. École des hautes études commerciales, Groupe de recherche en finance, 1997.

[52] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

[53] Bernard Dumas, Jeff Fleming, and Robert E Whaley. Implied volatility functions: Empirical tests. *The Journal of Finance*, 53(6):2059–2106, 1998.

[54] B. Dupire. Pricing with a smile. *Risk*, 7:18–20, 1994.

[55] Bruno Dupire et al. Pricing with a smile. *Risk*, 7(1):18–20, 1994.

[56] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[57] Robert F Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica: Journal of the Econometric Society*, pages 987–1007, 1982.

[58] Marcelo Espinoza, Johan AK Suykens, and Bart De Moor. Fixed-size least squares support vector machines: A large scale application in electrical load forecasting. *Computational Management Science*, 3(2):113–129, 2006.

[59] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in computational mathematics*, 13(1):1–50, 2000.

[60] Yunlong Feng, Yuning Yang, Xiaolin Huang, Siamak Mehrkanoon, and Johan AK Suykens. Robust support vector machines for classification with nonconvex and smooth losses. *Neural computation*, 2016.

[61] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[62] H. Föllmer and M. Schweizer. Hedging by sequential regression: An introduction to the mathematics of option trading. *The ASTIN Bulletin*, 1:147–160, 1989.

[63] Kenneth R French, G William Schwert, and Robert F Stambaugh. Expected stock returns and volatility. *Journal of financial Economics*, 19(1):3–29, 1987.

[64] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[65] René Garcia and Ramazan Gençay. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics*, 94(1):93–115, 2000.

[66] Jim Gatheral. *The volatility surface: a practitioner's guide*, volume 357. John Wiley & Sons, 2011.

[67] Ramazan Gençay and Min Qi. Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *IEEE Transactions on Neural Networks*, 12(4):726–734, 2001.

[68] Ramazan Gençay, Aslihan Salih, et al. Degree of mispricing with the black-scholes model and nonparametric cures. *Annals of Economics and Finance*, 4:73–102, 2003.

[69] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269, 1995.

[70] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[71] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[72] S. Goutte, N. Oudjane, and F. Russo. Variance optimal hedging for discrete time processes with independent increments, application to electricity markets. *Journal of Computational Finance*, 17:71–111, 2013.

[73] Nikola Gradojevic, Ramazan Gençay, and Dragan Kukolj. Option pricing with modular neural networks. *IEEE transactions on neural networks*, 20(4):626–637, 2009.

[74] Ming Gu and Stanley C Eisenstat. A stable and fast algorithm for updating the singular value decomposition, 1994.

[75] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[76] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

[77] Patrick Hagan and Andrew Lesniewski. Bartlett's delta in the sabr model. *Available at SSRN 2950749*, 2017.

[78] Patrick S Hagan, Deep Kumar, Andrew S Lesniewski, and Diana E Woodward. Managing smile risk. *The Best of Wilmott*, page 249, 2002.

[79] C. He, J.S. Kennedy, T. F. Coleman, P. A. Forsyth, Y. Li, and K. R. Vetzal. Calibration and hedging under jump diffusion. *Review of Derivative Research*, 9(1):1–35, 2006.

[80] D. Heath, E. Platen, and M. Schweizer. Numerical comparison of local risk-minimisation and mean-variance hedging. In *Option pricing, interest rates and risk management*, pages 509–537. (ed. E. Jouini, J. Cvitanic and, M. Musiela), Cambridge Univ. Press, 2001b.

[81] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of financial studies*, 6(2):327–343, 1993.

[82] Steven L Heston and Saikat Nandi. A closed-form garch option valuation model. *The review of financial studies*, 13(3):585–625, 2000.

[83] Morris W Hirsch. Convergent activation dynamics in continuous time networks. *Neural networks*, 2(5):331–349, 1989.

[84] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[85] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[86] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[87] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[88] Peter J Huber et al. Robust estimation of a location parameter. *The annals of mathematical statistics*, 35(1):73–101, 1964.

[89] John Hull and Alan White. The pricing of options on assets with stochastic volatilities. *The journal of finance*, 42(2):281–300, 1987.

[90] John Hull and Alan White. Optimal delta hedging for options. *Journal of Banking & Finance*, 82:180–190, 2017.

[91] John C Hull. *Options, futures, and other derivatives*. Pearson Education India, 2006.

[92] James M Hutchinson, Andrew W Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889, 1994.

[93] Jens Jackwerth and Mark Rubinstein. Recovering probability distributions from option prices. *The Journal of Finance*, 51(5):1611–1631, 1996.

[94] Ling Jian, Zhonghang Xia, Xijun Liang, and Chuanhou Gao. Design of a multiple kernel learning algorithm for ls-svm by convex programming. *Neural Networks*, 24(5):476–483, 2011.

[95] Licheng Jiao, Liefeng Bo, and Ling Wang. Fast sparse approximation for least squares support vector machine. *IEEE Transactions on Neural Networks*, 18(3):685–697, 2007.

[96] Michael I Jordan. Serial order: A parallel distributed processing approach. Technical report, CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1986.

[97] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[98] Steven G Kou. A jump-diffusion model for option pricing. *Management science*, 48(8): 1086–1101, 2002.

[99] Nathan Lassance and Frédéric Vrins. A comparison of pricing and hedging performances of equity derivatives models. *Applied Economics*, 50(10):1122–1137, 2018.

[100] Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

[101] Felix Lenders, Christian Kirches, and Andreas Potschka. trlib: A vector-free implementation of the gltr method for iterative solution of the trust region problem. *Optimization Methods and Software*, 33(3):420–449, 2018.

[102] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[103] Philip M Long and Rocco A Servedio. Random classification noise defeats all convex potential boosters. *Machine learning*, 78(3):287–304, 2010.

[104] Andrew L Maas, Quoc V Le, Tyler M O'Neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. Recurrent neural networks for noise reduction in robust asr. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[105] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[106] Mary Malliaris and Linda Salchenberger. A neural network model for estimating option prices. *Applied Intelligence*, 3(3):193–206, 1993.

[107] Benoit Mandelbrot. The variation of certain speculative prices. *The journal of business*, 36(4):394–419, 1963.

[108] Elena Marchiori, Niels HH Heegaard, Mikkel West-Nielsen, and Connie R Jimenez. Feature selection for classification with proteomic data of mixed quality. In *Computational Intelligence in Bioinformatics and Computational Biology, 2005. CIBCB'05. Proceedings of the 2005 IEEE Symposium on*, pages 1–7. IEEE, 2005.

[109] James Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, pages 735–742, 2010.

[110] James Martens and Ilya Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1033–1040. Citeseer, 2011.

[111] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[112] James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

[113] R. Merton. The theory of rational option pricing. *Bell Journal of Economics and Management Science*, 4:141–183, 1973.

[114] Robert C Merton. Theory of rational option pricing. *The Bell Journal of economics and management science*, pages 141–183, 1973.

[115] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[116] Michael C Mozer. A focused back-propagation algorithm for temporal pattern recognition. *Complex systems*, 3(4):349–381, 1989.

[117] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547, 1983.

[118] Ke Nian, Thomas F Coleman, and Yuying Li. Learning minimum variance discrete hedging directly from the market. *Quantitative Finance*, pages 1–14, 2018.

[119] Ke Nian, Thomas F Coleman, and Yuying Li. Learning sequential option hedging models from market data. *Journal of Banking and Finance*, 2019. Accepted Pending Revision.

[120] Ke Nian, Thomas F Coleman, and Yuying Li. Learning sequential total hedging models from market data. Submitted, 2019.

[121] Christopher Olah. Understanding lstm networks. *GITHUB blog, posted on August*, 27: 2015, 2015.

[122] LLC OptionMetrics. Ivy db file and data reference manual, 2008.

[123] Tapio Pahikkala, Jorma Boberg, and Tapio Salakoski. Fast n-fold cross-validation for regularized least-squares. In *Proceedings of the ninth Scandinavian conference on artificial intelligence (SCAI 2006)*, pages 83–90. Otamedia Oy, Espoo, Finland, 2006.

[124] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[125] K. Poulsen R., R. Schenk-Hoppé, and C.-O Ewald. Risk minimization in stochastic volatility models: model risk and empirical performance. *Quantitative Finance*, 9(6):693–704, 2009.

[126] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. Early stopping and non-parametric regression: an optimal data-dependent stopping rule. *The Journal of Machine Learning Research*, 15(1):335–366, 2014.

[127] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

[128] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[129] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1-2):23–69, 2003.

[130] Mark Rubinstein. Implied binomial trees. *The Journal of Finance*, 49(3):771–818, 1994.

[131] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[132] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[133] M. Schäl. On quadratic cost criteria for option hedging. *Mathematics of Operation Research*, 19(1):121–131, 1994.

[134] M. Schweizer. Variance-optimal hedging in discrete time. *Mathematics of Operation Research*, 20:1–32, 1995.

[135] Steven E Shreve. *Stochastic calculus for finance II: Continuous-time models*, volume 11. Springer Science & Business Media, 2004.

[136] Alex J Smola and Bernhard Schökopf. Sparse greedy matrix approximation for machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 911–918. Morgan Kaufmann Publishers Inc., 2000.

[137] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[138] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[139] Richard S Sutton. Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proc. 8th annual conf. cognitive science society*, pages 823–831. Erlbaum, 1986.

[140] Johan AK Suykens, Tony Van Gestel, and Jos De Brabanter. *Least squares support vector machines*. World Scientific, 2002.

[141] Mingkui Tan, Li Wang, and Ivor W Tsang. Learning sparse svm for feature selection on very high dimensional datasets. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 1047–1054, 2010.

[142] Aditya Tayal, Thomas F Coleman, and Yuying Li. Primal explicit max margin feature selection for nonlinear support vector machines. *Pattern Recognition*, 47(6):2153–2164, 2014.

[143] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[144] Vladimir Naumovich Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[145] Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1065–1072. ACM, 2009.

[146] Grace Wahba. *Spline models for observational data*, volume 59. Siam, 1990.

[147] Kuaini Wang and Ping Zhong. Robust non-convex least squares loss function for regression with outliers. *Knowledge-Based Systems*, 71:290–302, 2014.

[148] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

[149] Christopher KI Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[150] Yichao Wu and Yufeng Liu. Robust truncated hinge loss support vector machines. *Journal of the American Statistical Association*, 102(479):974–983, 2007.

[151] Linli Xu, Koby Crammer, and Dale Schuurmans. Robust support vector machine training via convex outlier ablation. In *AAAI*, volume 6, pages 536–542, 2006.

[152] Peng Xu, Farbod Roosta-Khorasan, and Michael W Mahoney. Second-order optimization for non-convex machine learning: An empirical study. *arXiv preprint arXiv:1708.07827*, 2017.

[153] Xiaowei Yang, Liangjun Tan, and Lifang He. A robust least squares support vector machine for regression and classification with noise. *Neurocomputing*, 140:41–52, 2014.

[154] Jingtao Yao, Yili Li, and Chew Lim Tan. Option price forecasting using neural networks. *Omega*, 28(4):455–466, 2000.

[155] Zhewei Yao, Peng Xu, Farbod Roosta-Khorasani, and Michael W Mahoney. Inexact non-convex newton-type methods. *arXiv preprint arXiv:1802.06925*, 2018.

[156] Adonis Yatchew and Wolfgang Härdle. Nonparametric state price density estimation using constrained least squares and the bootstrap. *Journal of Econometrics*, 133(2):579–599, 2006.

[157] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[158] Yao-liang Yu, Özlem Aslan, and Dale Schuurmans. A polynomial-time form of robust regression. In *Advances in Neural Information Processing Systems*, pages 2483–2491, 2012.

[159] Yaoliang Yu, Xun Zheng, Micol Marchetti-Bowick, and Eric Xing. Minimizing nonconvex non-separable functions. In *Artificial Intelligence and Statistics*, pages 1107–1115, 2015.

[160] Haynes HM Yung and Hua Zhang. An empirical investigation of the garch option pricing model: hedging performance. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 23(12):1191–1207, 2003.

[161] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

APPENDICES