How Can Deep Learning Crack the Puzzle of Financial Derivatives Pricing?

Unveiling the Neural SDE Model

A. Theoretical explanation

A.1. Background, Challenges and Opportunities of pricing Financial derivatives

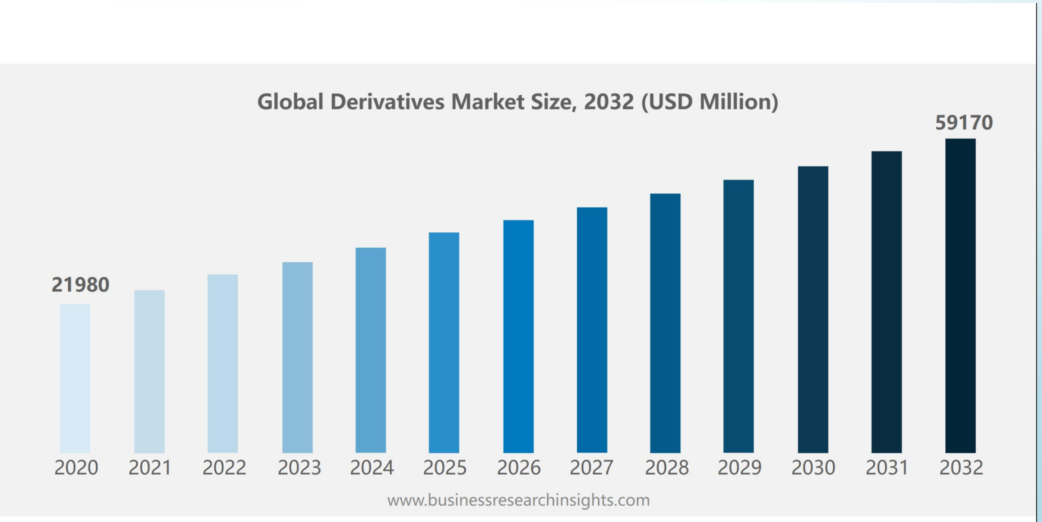
A.1.1. The Importance of pricing Financial derivatives

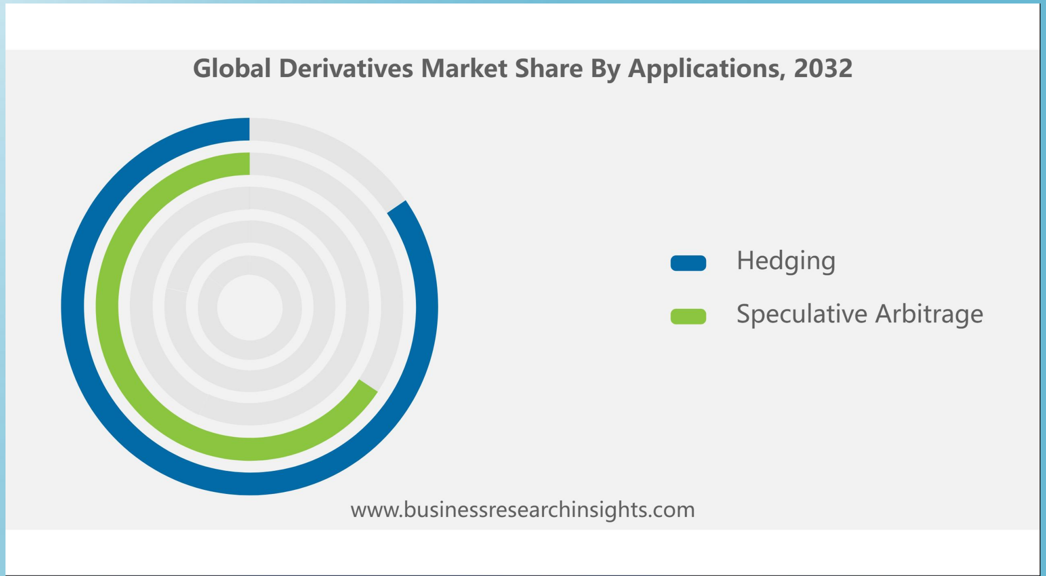
1. Risk management: through reasonable pricing and application, it can hedge and transfer all kinds of financial risks, such as market risks and credit risks.

2. Market efficiency: reasonable pricing can guide the effective allocation of market resources and promote price discovery. Increase the transparency of the market and reduce information asymmetry.

3. Product innovation: It provides theoretical basis and quantitative tools for the innovation of financial products. Develop financial products that meet the needs of different investors and enrich the investment options in the financial market.

4. Trading strategy: Through the prediction and analysis of derivatives prices, traders can find market mispricing and arbitrage opportunities, and formulate corresponding trading strategies to obtain excess returns.





A.1.2. Challenges faced by traditional pricing models

- The option pricing model based on Black-Scholes-Merton(BSM) formula has some ***major limitations:***
- ***1. Constant Volatility assumption:*** The BSM model assumes that the volatility of the underlying asset is constant, while in the actual situation, the volatility changes dynamically over time. This assumption is too simplistic. SDE models such as Heston model can well capture the stochastic evolution process of volatility.
- ***2. Lognormal distribution assumption:*** The BSM model assumes that the log return rate of the underlying asset follows a normal distribution, but in fact, financial time series data have non-normal distribution characteristics such as "Fat Tail". However, the SDE model can better fit the real distribution shape by setting different diffusion processes.
- ***3. Single option type:*** The BSM formula is mainly used for the pricing of common European call/put options, while other types of options, such as barrier options and lookback options, need to be solved by other complex numerical integration methods. However, the neural network method can directly learn the pricing maps of these complex options by providing enough rich training data.
- ***4. Static model:*** BSM is a static model, which requires fixed parameters such as volatility when pricing, while the actual parameters will change dynamically with time. SDE model can dynamically describe the stochastic evolution process of parameters.
- ***5. Limitations of model extension:*** In the BSM framework, extending the model to capture more complex financial phenomena (e.g., jump processes, stochastic volatility, etc.) can lead to complex or impossible analytical solutions. However, the data-based neural network method does not need to solve the problem and is more flexible in model expansion.

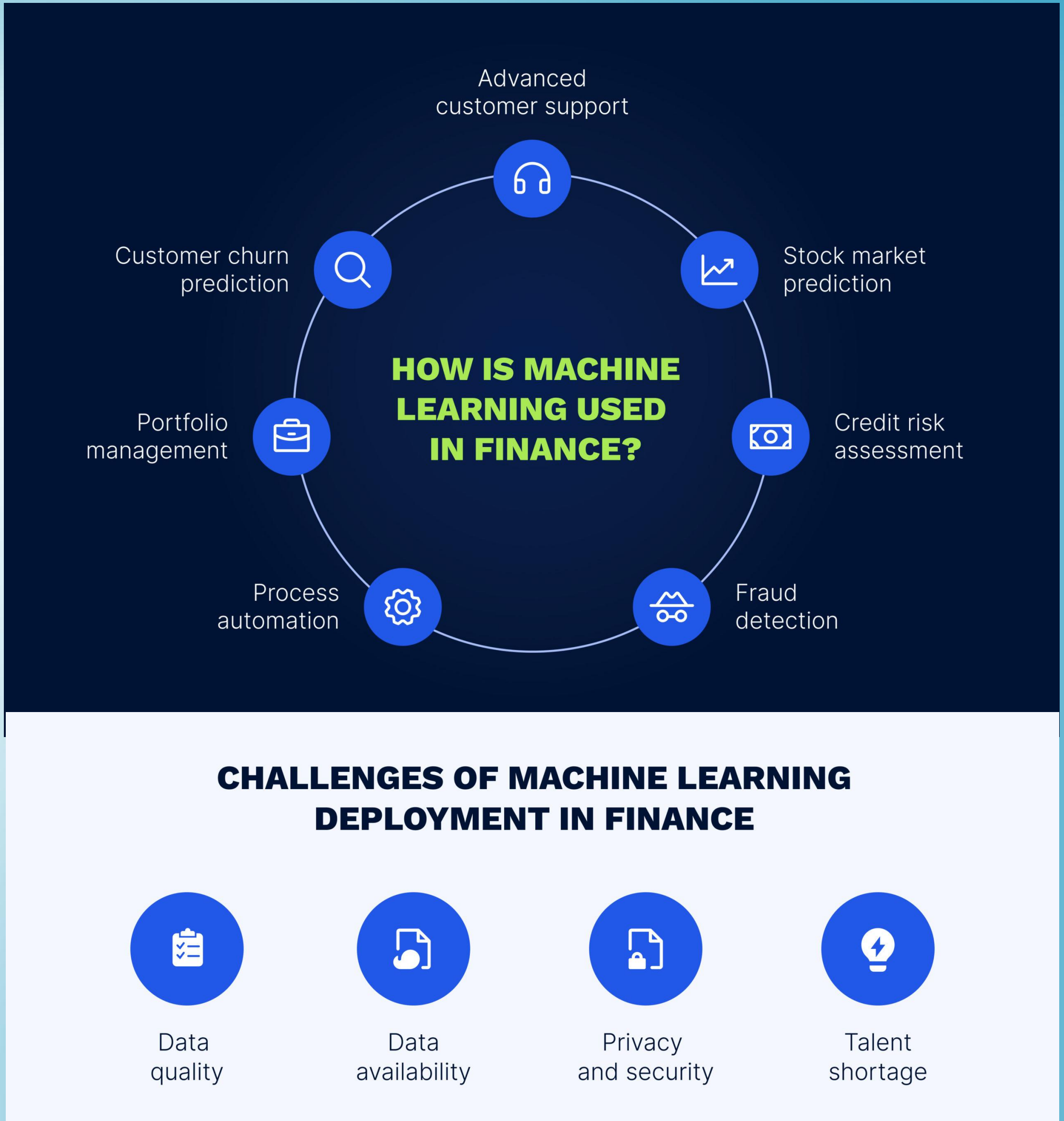
A.1.3. Promising Applications of Machine Learning Methods in Finance

1. Risk management and fraud detection: Establish a credit risk assessment model to predict the default probability of borrowers by analyzing historical data. Anomaly detection algorithms can be applied to identify fraudulent transactions, such as credit card fraud, money laundering, etc.

2. Investment decision and trading strategy: identify key factors affecting asset prices by analyzing massive market data, predict future price trends, and provide decision support for investors.

3. Customer relationship management: identify the characteristics and needs of different customer groups by analyzing customer data, and provide customized financial products and services.

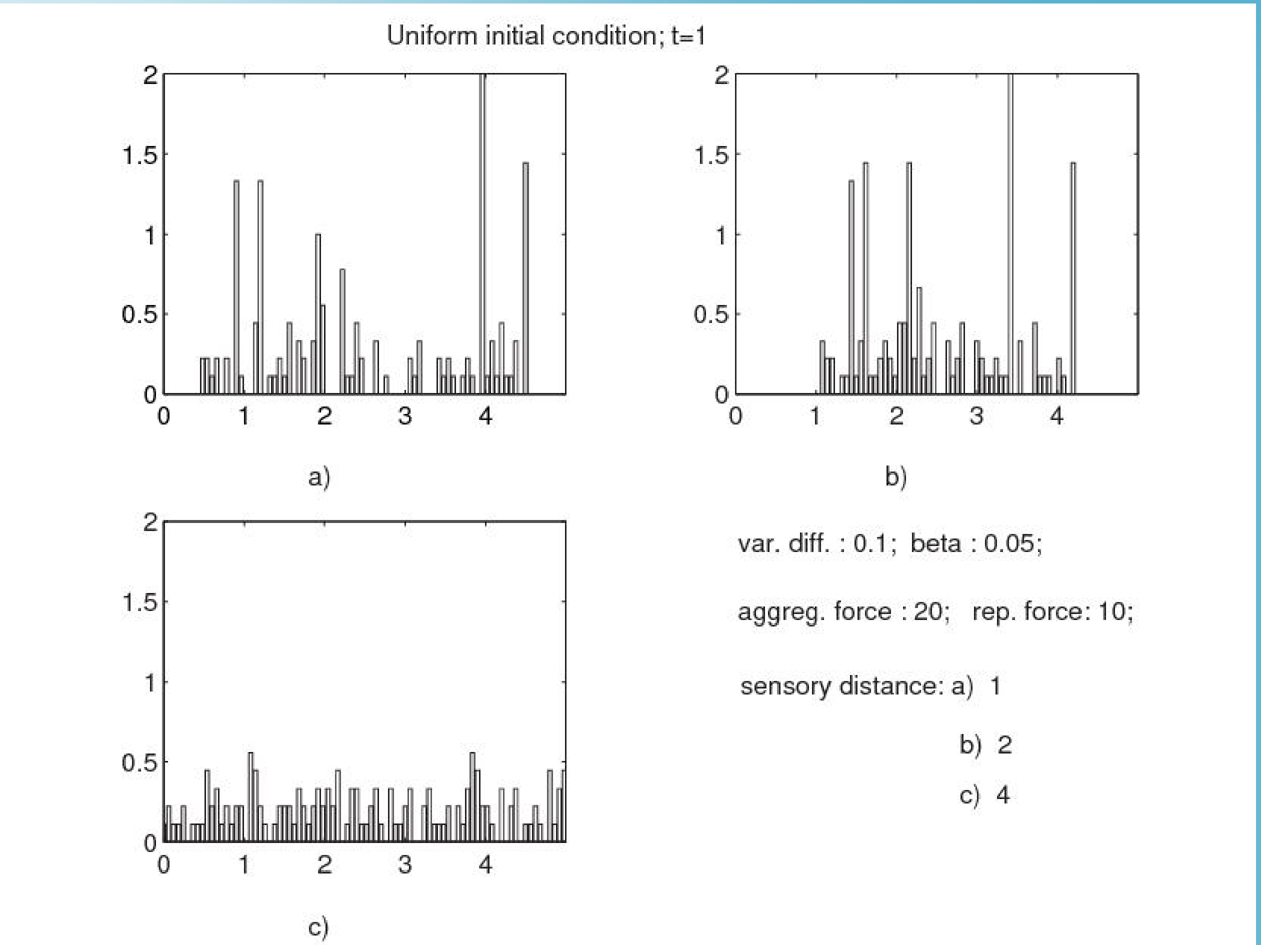
4. Financial forecasting: forecasting macroeconomic indicators, corporate performance, etc. By analyzing economic data, financial statements, etc., the forecast model is established to provide reference for investors and decision makers.



A.2. Introduction to the classical SDE model and its limitations

A.2.1. Basic Concepts of the SDE model

- 1. Stochastic differential equations (SDE) can describe the ***stochastic evolution of asset prices.***
- 2. Classical SDE models include ***local volatility models***, ***stochastic volatility models***, etc.
- Stochastic Differential Equation (SDE) is a mathematical model ***describing the evolution of stochastic processes***. The general form is:
- ***dXt = b(t, Xt)dt + σ(t, Xt)dWt***
- Where ***Xt is a random process,b(t, Xt) is a drift term,σ(t, Xt) is a diffusion term, and Wt is a standard Brownian motion***. The drift term characterizes the deterministic trend of the process, and the diffusion term characterizes the stochastic fluctuation of the process.

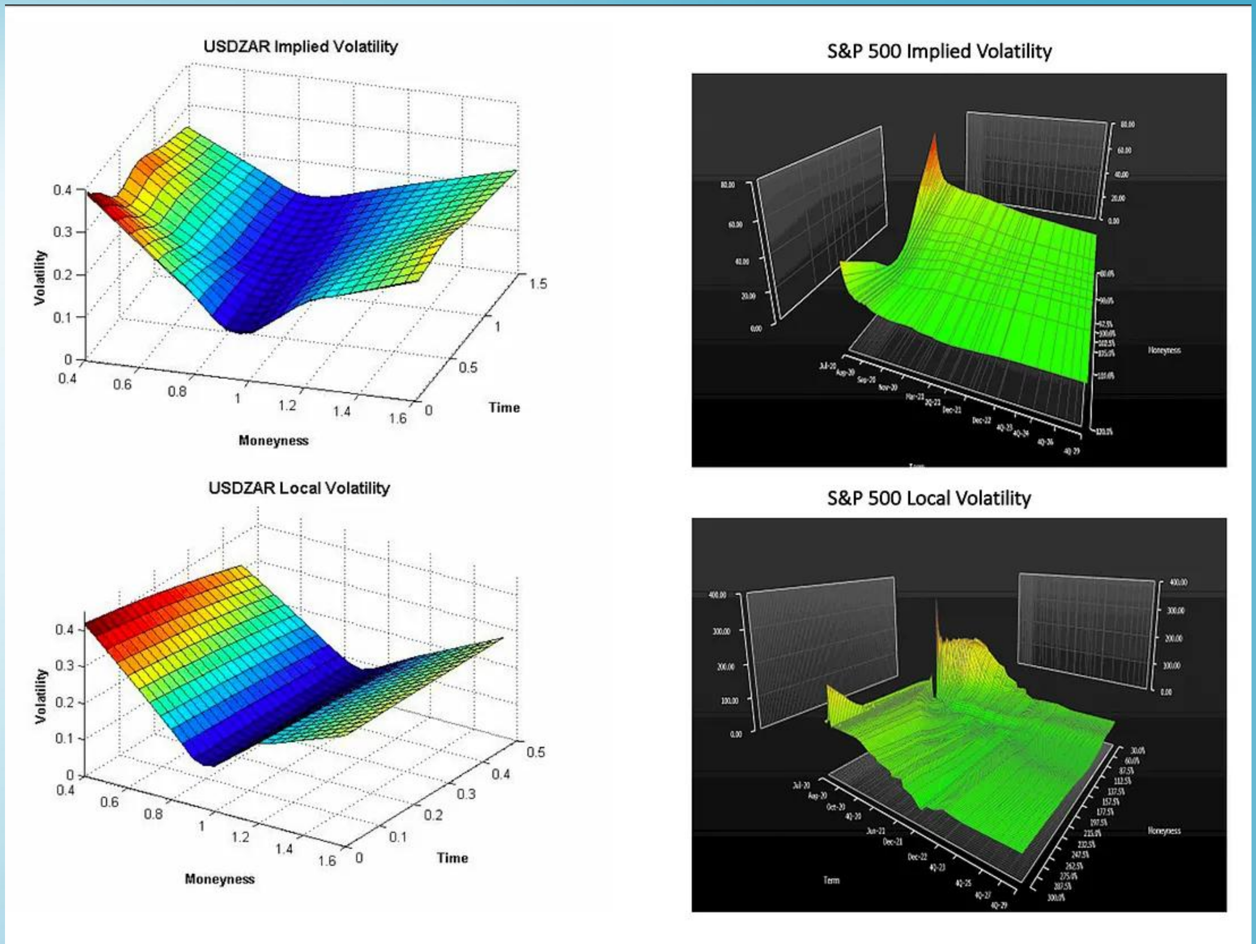


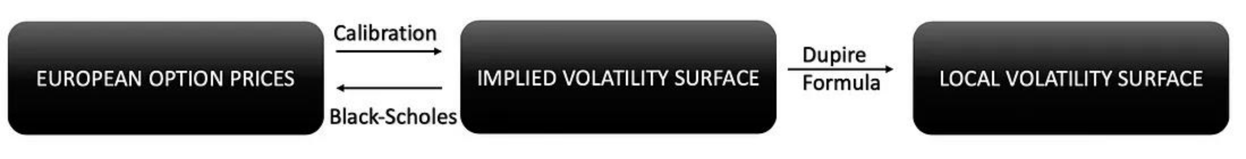
A.2.1. Basic Concepts of the SDE model

- ***var.diff. (variance of diffusion): 0.1*** This means that the variance of the diffusion term is 0.1. In stochastic differential equations, the diffusion term controls the magnitude of the random noise. The greater the variance, the greater the influence of noise, and the solution path will exhibit more randomness.
- ***beta: 0.05*** controls the size of the deterministic drift term. The larger β is, the stronger is the effect of drift, and the solution path will move in the direction of deterministic drift.
- ***aggreg.force (aggregation force): 20*** The aggregation force parameter is 20. In some models of biology or physics, the aggregation force describes the attractive effect between individuals such that they tend to cluster together. The larger the aggregation force, the solution path will show more obvious aggregation characteristics.
- ***rep.force (<s:1> force): 10*** The repulsive force parameter is 10. In contrast to the aggregation force, the repulsive force describes the repulsive effect between individuals, making them tend to move away from each other. The larger the repulsive force, the more obvious dispersion characteristics will be displayed in the path of the solution.
- ***sensory distance: a) 1 b) 2 c) 4*** The sensor distances are 1, 2, and 4, respectively. In some individual-based models, sensor distance describes the range within which an individual perceives its surroundings. The larger the sensor distance, the more information an individual can perceive, and its behavior will be influenced by other individuals in a larger range.
- Together, these parameters affect the behavioral characteristics of the SDE model. By tuning these parameters, we can simulate complex systems with different dynamical properties, such as biological population dynamics, particle motion, etc. At the same time, it also shows that the SDE model is a flexible and powerful modeling tool capable of integrating multiple interaction mechanisms and generating rich and diverse dynamic behaviors.

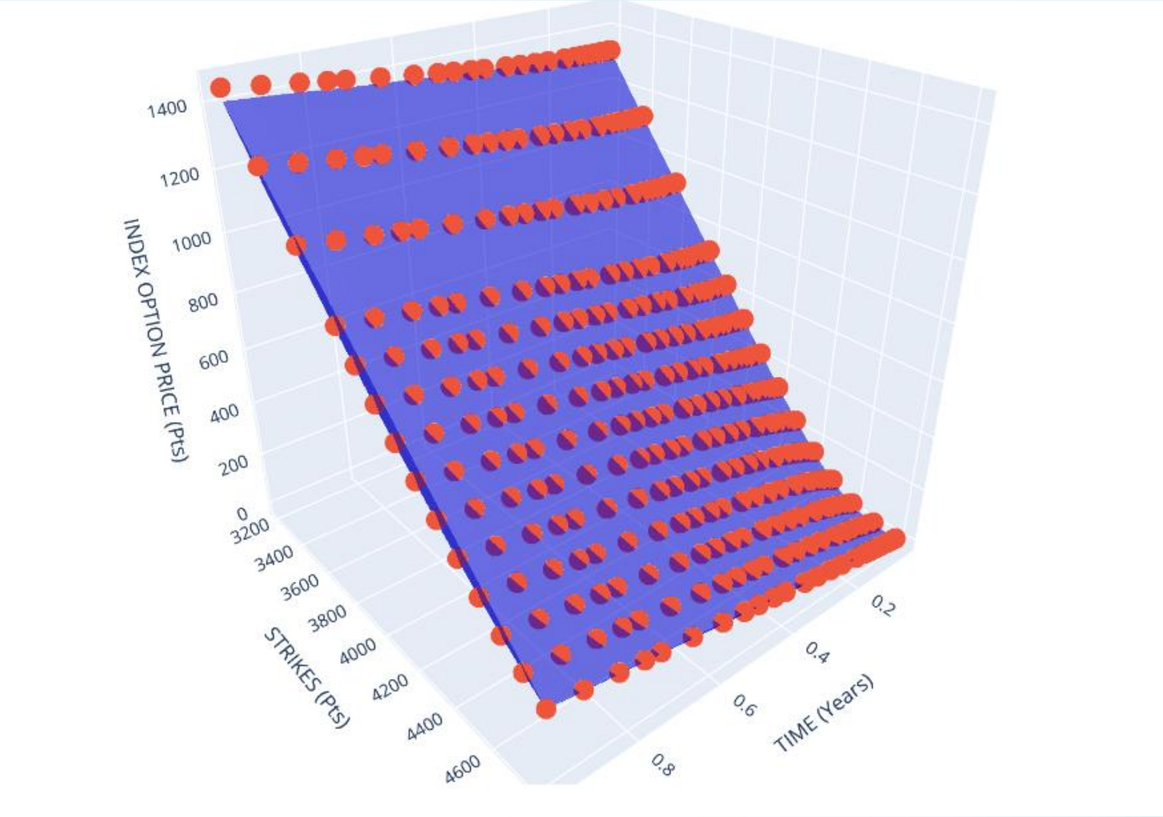
A.2.2. Local Volatility model

- The model assumes that the ***instantaneous volatility of the underlying asset is a deterministic function of the asset price and time, namely σ(S,t)***.
- Unlike the ***constant volatility in the Black-Scholes model***, the ***local volatility varies with price and time***, so it can better ***reflect the volatility structure of the market***.
- The ***underlying asset price S satisfies*** the following stochastic differential equation (SDE):
- $dS = \mu S dt + \sigma(S,t) S dW$
- Where ***μ is the drift rate,σ(S,t) is the local volatility function, and W is the standard Brownian motion***. The ***drift rate μ is usually set to the risk-free interest rate r*** to ensure the risk-neutral nature of the model.
- The key is to ***determine the local volatility function σ(S,t)***. This is usually achieved by a ***calibration process whereby the implied local volatility surface is inversely derived from the option prices observed by the market***.





A.2.3. Stochastic Volatility Model



- 1. Introduce the ***"volatility of volatility"*** by assuming that the volatility itself also follows a stochastic process. 2. The ***randomness of volatility*** is introduced, which can ***better explain the characteristics such as volatility smiles and spikes observed in the market***.
- The ***underlying asset price S and volatility V satisfy*** the following stochastic differential equation (SDE) system:
- $dS = \mu S dt + \sqrt{V_t} S dW_1$
- $dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t} dW_2$
- $dW_1 dW_2 = \rho dt$
- Where ***μ is the drift rate, Vt is the stochastic volatility process, κ is the mean-return speed of volatility, θ is the long-term mean of volatility, σ is the volatility of volatility, W1 and W2 are correlated standard Brownian motions, and ρ is the correlation coefficient of the two Brownian motions.***

A.2.4. Limitations of the traditional SDE model

1. Limitations of model assumptions:

The ***local volatility model*** assumes that volatility is a deterministic function of asset prices and time, ***ignoring the stochastic nature of volatility***.

Although ***stochastic volatility models*** introduce the stochastic process of volatility, it is usually ***assumed that the volatility process follows specific dynamics, such as the mean-reverting*** process, etc.

These assumptions may ***not fully capture the complex dynamics and nonlinear features of volatility in financial markets***.

2. Difficulty of parameter estimation and calibration:

Local and stochastic volatility models usually ***need to estimate a large number of parameters***, such as ***drift rate, volatility, correlation coefficient***, etc.

The estimation of these parameters is usually ***based on historical data and option price data*** and ***requires complex optimization and calibration processes***.

Uncertainty and error in parameter estimation may have a ***significant impact on the pricing results*** of the model.

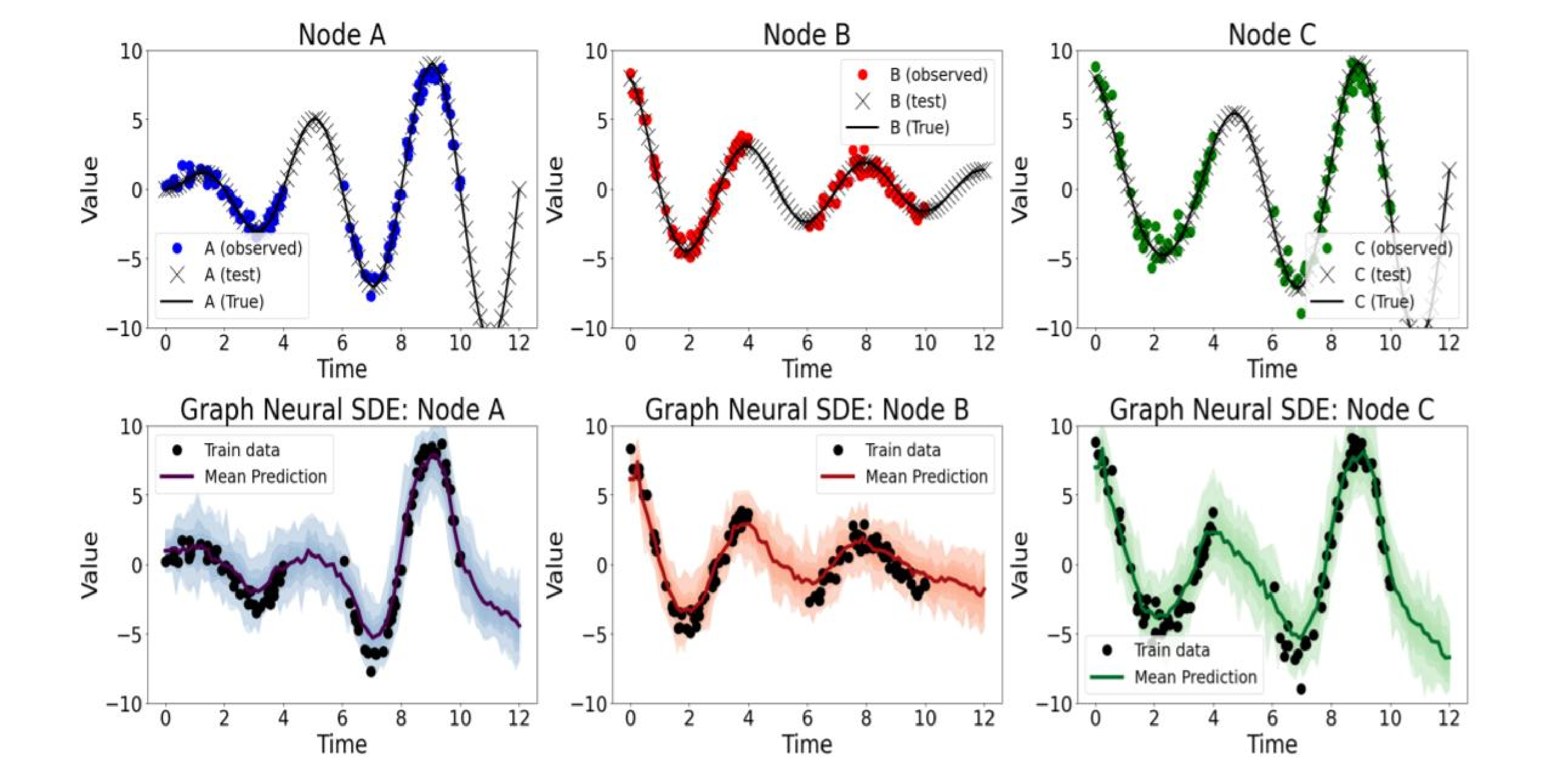
A.2.4. Limitations of the traditional SDE model

3. These two types of models ***underutilize other information*** such as market microstructure, transaction flow, and sentiment, which may affect the accuracy of pricing.

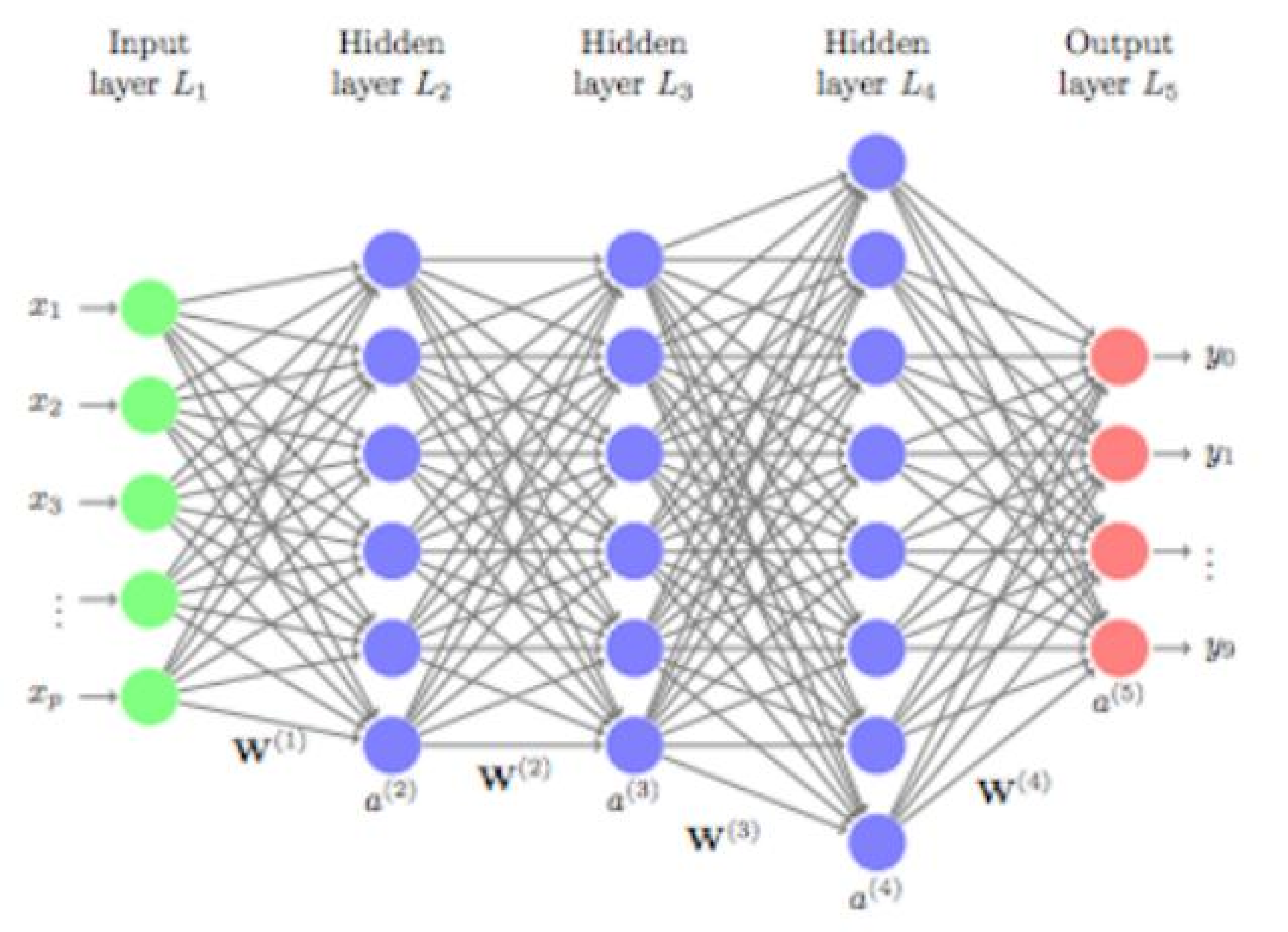
4. The impact of ***extreme market volatility*** and black swan events may be underestimated, leading to a bias in risk assessment and pricing.

5. Local volatility models and stochastic volatility models usually have ***specific mathematical forms and assumptions***, and it is difficult to extend and modify the models.

A.3. Neural SDE model innovations, network structure, loss function, and other details



A.3.1. Neural Networks?



A.3.1. Neural Networks?

- Neural Networks are mathematical models that mimic the structure and function of biological nervous systems and consist of a large number of interconnected nodes, or neurons. Here are some basic concepts and principles of neural networks:

1. Neuron model:

- Neurons are the basic units of neural networks. Each neuron ***receives a set of input signals*** and ***generates an output signa***l by ***weighted summation and nonlinear transformation***. Mathematically, it is expressed as follows.
- ***y = f(∑i wi xi + b)***
- Where ***xi is the input signal,wi is the corresponding weight,b is the bias term***, and ***f is the activation function***, such as ***sigmoid, tanh, ReLU***, etc.

2. Network structure:

- Neurons are organized in ***layers*** to form neural networks. Generally, it includes ***input layer, hidden layer and output layer***. Neurons in ***each layer are fully connected to neurons in adjacent layers***, while neurons within the ***same layer are not connected to each other***. The ***number of layers*** of the network and the ***number of neurons in each layer*** determine the ***complexity and expressiveness*** of the network.

3. Feed-Forward:

- Given an input, the signal is passed from the input layer to the output layer in a process called Feed-Forward. The output of each neuron in turn becomes the input to the neurons of the next layer until the final output is produced. Feed-Forward is used to compute the predicted values of the network.

A.3.1. Neural Networks?

4. Loss function:

- The loss function measures the ***difference between the predicted value*** of the network ***and the true value***. Common loss functions include mean squared error, cross entropy, etc. The goal of network training is to minimize the loss function in order to improve the prediction accuracy.

5. Backpropagation:

- Backpropagation is an efficient gradient calculation method used to ***update network parameters (weights and biases)*** during training. Through the chain rule, starting from the output layer, the gradient of the loss function with respect to each parameter is calculated layer by layer, and the parameters are updated according to the gradient descent method to minimize the loss function.

A.3.1. Neural Networks?

6. Optimization algorithm:

Objective: To ***find the optimal model parameters*** by ***adjusting the parameters of the model and minimizing the loss function***

Target: The optimization algorithm directly acts on the training process of the model, ***guiding the direction and size of the parameter updates***

Common algorithms: Gradient Descent (GD), Stochastic Gradient Descent (SGD), Momentum, Adaptive learning rate (AdaGrad, RMSprop, Adam), etc

Points: Optimization algorithms focus on ***how to efficiently and quickly find the optimal parameters to make the model perform best on the training set***

7. Regularization:

Objective: To ***control model complexity, reduce overfitting, and improve generalization ability*** by ***imposing additional constraints or penalties*** on the model

Target: Regularization ***modifies the loss function*** to ***constrain or penalize model parameters during training***

Common techniques: L1 regularization, L2 regularization, Dropout, Early Stopping, etc

Points: How does regularization concern ***limit the complexity of the model*** so that it also ***performs well on unseen data*** and ***improves generalization***

A.3.2. Basic idea of Neural SDE model

What follows is heavily based on and inspired by the following main references in this paper:

Gierjatowicz, P., Sabate-Vidales, M., Šiška, D., Szpruch, L. andŽ urič, Ž., 2020. Robust pricing and hedging via neural SDEs. arXiv preprint arXiv:2007.04154.

A.3.2. Basic idea of Neural SDE model

1. Stochastic Differential Equation Modeling: Neural SDE models stochastic dynamical systems ***based on stochastic differential equations*** (SDE). SDE consists of two parts: ***a drift term and a diffusion term***. The ***drift term describes the deterministic evolution trend of the system***, while the ***diffusion term describes the stochastic fluctuations of the system***.

2. Neural Network Parameterization: In Neural SDE, a ***neural network*** is used to ***parameterize the drift and diffusion terms*** of the SDE. This means that the ***output of the neural network*** is used to ***represent the functional form of the drift and diffusion terms***. By adjusting the weights and biases of the neural network, a functional representation of the drift and diffusion terms suitable for the data can be learned.

A.3.2. Basic idea of Neural SDE model

3. End-to-end training: The training process of Neural SDE is end-to-end, that is, the ***parameters of the neural network*** are optimized to ***minimize the discrepancy between the predicted values of the model and the observed data***. During training, the observations are passed to the neural network as input features, and the output of the neural network is used as the drift and diffusion terms of the SDE. Then, a loss function is used to measure the difference between the predicted value of the model and the true observation value, and the parameters of the neural network are updated by backpropagation and optimization algorithm to minimize the loss function.

4. Prediction and Inference: The trained Neural SDE model can be used to make predictions and inferences on new inputs. ***Given new input features***, they are ***passed to the trained neural network*** to ***obtain estimates of the corresponding drift and diffusion terms***. These estimates can be used to ***model the evolution of the SDE*** to obtain a ***prediction of the future state***.

A.3.3. Compare to traditional SDE model...

In comparison, the main advantages of Neural SDE are as follows:

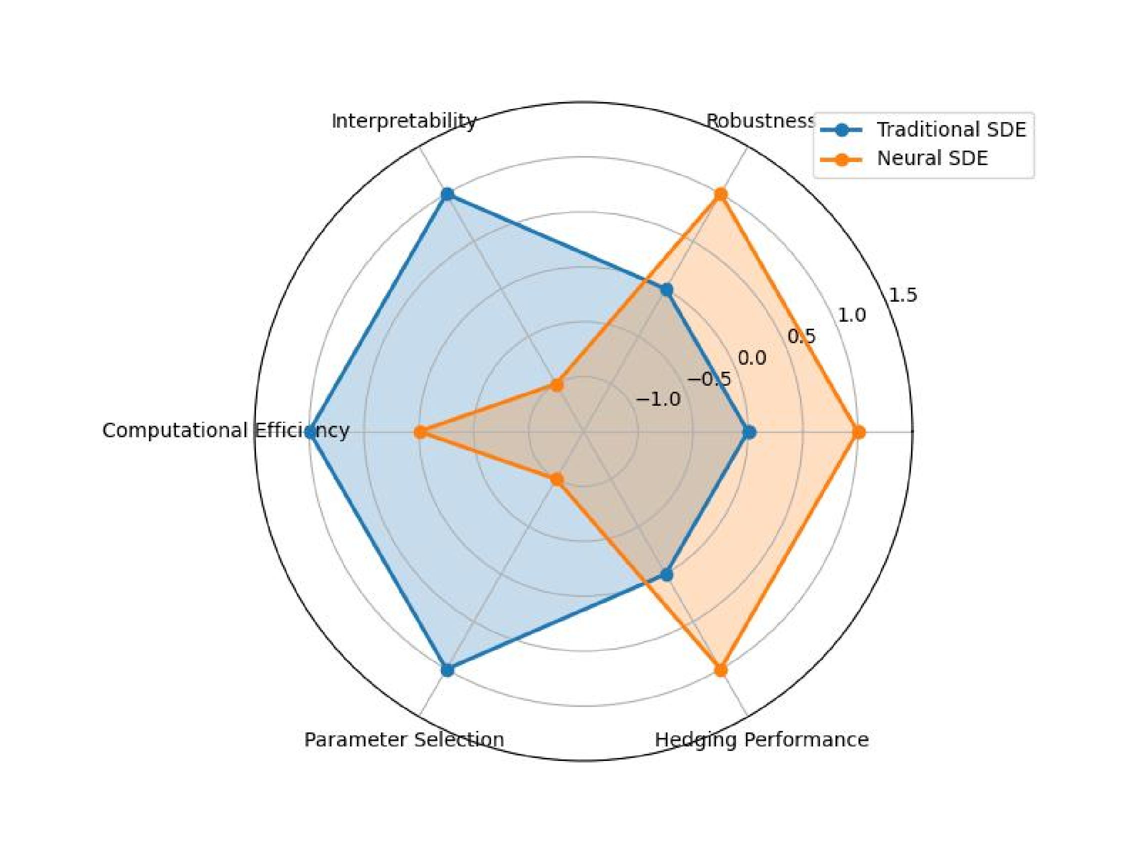
1. With flexible ***neural network parameterization***, we are able to ***learn more complex volatility dynamics***, reducing the limitations of model assumptions.

2. In an ***end-to-end training*** manner, the ***model parameters are directly learned from the data***, which reduces the difficulty of parameter estimation and calibration.

3. It can flexibly ***include all kinds of market information as input features***, and make ***full use of market information*** for pricing and forecasting.

4. With a flexible neural network architecture, ***nonlinear and non-Gaussian features can be better captured to model extreme events and tail risks***.

5. Based on flexible neural network parameterization, ***new factors and features can be easily incorporated***, which has better scalability and adaptability.



A.4. Model Training and Prediction - Hands-on programming

A.4.1. BSM Model Reference Group-Code

```
def bsm_call_option_price(S, K, T, r, sigma):
    d1 = (np.log(S / K) + (r + 0.5 * sigma ** 2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    call_price = S * norm.cdf(d1) - K * np.exp(-r * T) * norm.cdf(d2)
    return call_price

# Calculate the BSM option price
bsm_prices = bsm_call_option_price(S, K, T, r, sigma)
```

A.4.2. Model Training

```python
# define neural SDE model
class NeuralSDE(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers):
        super(NeuralSDE, self).__init__()
        self.input_dim = input_dim
        self.hidden_dim = hidden_dim
        self.num_layers = num_layers

        self.drift_net = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            *[nn.Sequential(nn.Linear(hidden_dim, hidden_dim), nn.ReLU())
             for _ in range(num_layers - 1)],
            nn.Linear(hidden_dim, 1)
        )

        self.diffusion_net = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            *[nn.Sequential(nn.Linear(hidden_dim, hidden_dim), nn.ReLU())
             for _ in range(num_layers - 1)],
            nn.Linear(hidden_dim, 1),
            nn.Softplus()
        )

    def forward(self, x):
        drift = self.drift_net(x)
        diffusion = self.diffusion_net(x)
        return drift, diffusion
```

The __init__ method initializes the parameters of the model, including the input dimension, the hidden layer dimension, and the number of hidden layers.

Two neural networks, drift_net and diffusion_net, are defined to compute the drift and diffusion terms, respectively.

The forward method defines the forward propagation process of the model with input x and output drift and diffusion terms.

A.4.2. Model Training

```python
# train model
def train_model(model, dataloader, num_epochs, learning_rate, patience):
    criterion = nn.MSELoss()
    optimizer = optim.Adam(model.parameters(), lr=learning_rate)

    best_loss = float('inf')
    epochs_without_improvement = 0

    for epoch in range(num_epochs):
        epoch_loss = 0.0
        for batch in dataloader:
            optimizer.zero_grad()
            x_batch = batch[:, :-1].float()
            y_batch = batch[:, -1].float()
            drift, diffusion = model(x_batch)
            loss = criterion(drift, y_batch.unsqueeze(1))
            loss.backward()
            optimizer.step()
            epoch_loss += loss.item()

        epoch_loss /= len(dataloader)
        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss:.4f}")

        if epoch_loss < best_loss:
            best_loss = epoch_loss
            epochs_without_improvement = 0
            torch.save(model.state_dict(), "best_model.pth")
        else:
            epochs_without_improvement += 1
            if epochs_without_improvement >= patience:
                print(f"Early stopping at epoch {epoch+1}")
                break

    model.load_state_dict(torch.load("best_model.pth"))
    return model
```

1. The loss function criterion (mean squared error) and the optimizer (Adam) are defined.

2. The function trains the model by iterating over the batch data in the data loader for multiple training rounds.

3. In each batch, the gradient is first cleared, and then the input data x_batch is passed to the model to calculate the drift and diffusion terms. The loss function is used to calculate the loss between the predicted and true values, and backpropagation and parameter updates are performed.

4. EARLY STOPPING

A.4.3. Model Predictions

```python
# 定义 neural SDE 模型
class NeuralSDE(nn.Module):
    def __init__(self, input_dim, hidden_dim, num_layers):
        super(NeuralSDE, self).__init__()
        self.input_dim = input_dim
        self.hidden_dim = hidden_dim
        self.num_layers = num_layers

        self.drift_net = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            *[nn.Sequential(nn.Linear(hidden_dim, hidden_dim), nn.ReLU())
              for _ in range(num_layers - 1)],
            nn.Linear(hidden_dim, 1)
        )

        self.diffusion_net = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            *[nn.Sequential(nn.Linear(hidden_dim, hidden_dim), nn.ReLU())
              for _ in range(num_layers - 1)],
            nn.Linear(hidden_dim, 1),
            nn.Softplus()
        )

    def forward(self, x):
        drift = self.drift_net(x)
        diffusion = self.diffusion_net(x)
        return drift, diffusion

# 使用模型进行预测
def predict_price(model, option_params):
    option_params = torch.tensor(option_params, dtype=torch.float32)
    with torch.no_grad():
        drift, diffusion = model(option_params)
    return drift.item()
```

The function takes two arguments:

1. model: Trained neural SDE model.
2. option_params: A list or array of option parameters used to predict the option price.

The forward method passes option_params as input to the model.
The model returns two values: the drift term and the diffusion term.
The scalar value of the drift term is returned as the predicted option price.

A.4.4. Observation of prediction results

Mean Absolute Error (MAE) of BSM: 4.8146

Neural SDE Mean Absolute Error (MAE): 2.35911.

1. ***<u>Neural SDE</u>*** model has ***<u>smaller Mean absolute error (MAE)</u>*** and ***<u>better fit</u>***.

BSM Mean square Error (MSE): 50.8169

Neural SDE Mean square Error (MSE): 15.9046

2. ***<u>Neural SDE</u>*** model has a ***<u>smaller mean square error (MSE)</u>*** and a ***<u>better fit</u>***.

BSM Mean Absolute Percentage Error (MAPE): 24.1860

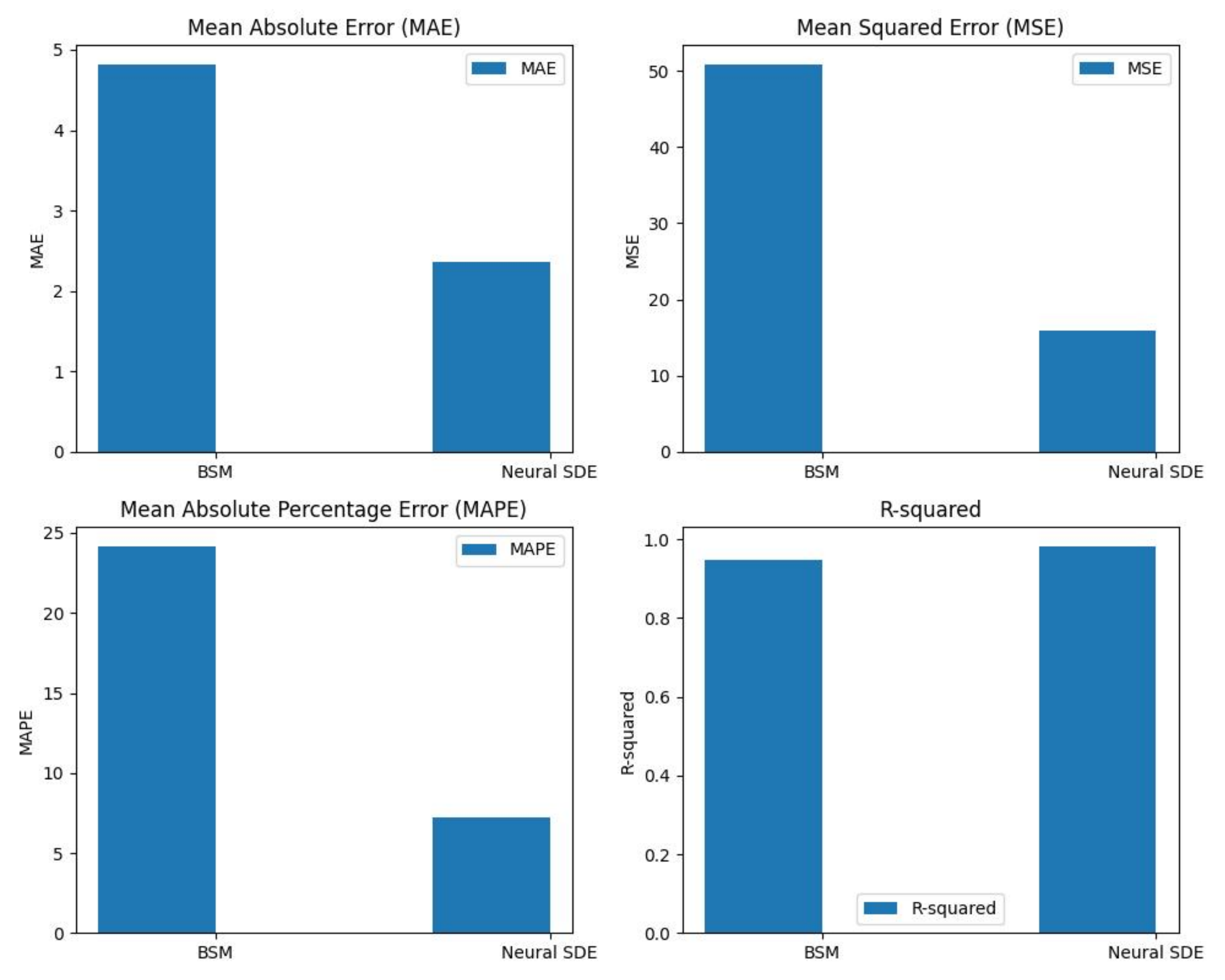
Neural SDE Mean Absolute Percentage Error (MAPE): 7.2606

3. ***<u>Neural SDE</u>*** model has ***<u>smaller Mean Absolute percentage error (MAPE)</u>*** and ***<u>better fit</u>***.

BSM R-squared: 0.9462

Neural SDE R-squared: 0.9832

4. The ***<u>Neural SDE</u>*** model has a ***<u>higher R-square</u>*** and a ***<u>better fit</u>***.



A.4.4. Observation of prediction results

Interesting observations:

Neural SDE and BSM two models in the comparison of the prediction results close to the true situation of the test, there is a ***clear watershed***!

In the case of ***YearsToMaturity<0.01*** (or DaysToMaturity<3), ***especially when YearsToMaturity=0*** (exercise period), ***BSM*** has ***significantly better prediction results*** than Neural SDE.

For ***options with long time*** to exercise, ***Neural SDE outperforms*** BSM ***by a large margin***, consistent with our overall results.

A.4.4. Observation of prediction results

Near the expiration date, such as <3 days, the ***option price*** is ***mainly determined by*** the ***current price*** and the ***strike price*** of the underlying asset, with ***little effect of volatility***. The ***BSM formulation*** can be seen as an ***approximate analytical solution***, which performs well in this case. ***Neural SDE***, as a ***numerical method***, may ***suffer from Monte Carlo errors*** and is slightly ***inferior*** to analytical solutions ***for very short deadlines***.

When the ***expiration date is far away***, the ***future volatility change*** has a ***great impact*** on the option pricing. The ***standard BSM model assumes that the volatility is constant***, while ***in practice the volatility often changes with time and the underlying price***. ***Neural SDE*** can ***significantly outperform BSM*** models by ***learning from market data and capturing the volatility surface***.

For sellers and buyers of options, I recommend the following ***strategies***:

For ***options with a maturity of <3 days***, it is ***simple and efficient*** to ***use the BSM formula*** to price and trade directly.

For ***options with distant maturity***, advanced models such as ***Neural SDE*** are ***preferred*** for pricing and risk management to ***cope with volatility risk***.

When ***hedging***, the ***strategy is adjusted according to the BSM Greeks(delta/gamma)*** in the ***short term*** and the ***SDE model in the long term***.

A.4.4. Observation of prediction results

For ***risk management*** departments, it is recommended to ***increase the stress test of very short term options***. This type of option has a ***high Gamma risk for the mutation of the underlying asset***, but it is ***easy to be ignored because of the very short time***. ***Special risk limits and monitoring*** should be established.

When validating the model, the ***maturity can be taken into account*** in the future to ***evaluate long-term and short-term performance separately***. It may be a good idea to ***develop a hybrid model***, where ***pricing is directly BSM in the very short term*** and then ***move to SDE in the medium to long term***.

A.4.4. Observation of prediction results

In our case:

1. We only talked about pricing options based on ***Apple stock***. What about ***other companies? Other industries?***

2. ***Only European call options*** are discussed. What about ***American options? Put options?***

3. Only discussed the advantages of ***Neural SDE*** model over the most ***classical BSM*** pricing model, how much advantage does neural SDE have over ***traditional SDE*** model (e.g. Heston)? Compared with the ***advanced binary tree*** model? Relative to ***Monte Carlo simulation***? ***Finite difference method***? These questions need more testing and research...

B. Further reading recommended

B.1. Further reading recommended

<u>1. Original paper:</u>

Gierjatowicz, P, Sabate-Vidales, M, Šiška, D, Szpruch, L. Žurič, Ž. , 2020. Robust Pricing and hedging with Neural SDEs. Preprint arXiv:2007.04154.

This is the original paper that proposed the neural SDE model referred, elaborating on the theoretical basis of the model, the network architecture, and the training method. It is a must-read for learning neural SDE.

<u>2. Advances in frontier research:</u>

Berner, R., Polka, F., Lio, P. and Hernandez-Lobato, j.m., 2023. Graph Neural Stochastic Differential Equations. Preprint arXiv:2308.12316.

This is a recent study that combines graph neural networks with SDE. It explores new ways to model stochastic dynamics on data with complex network structures and represents one of the cutting-edge progress in the field of neural SDE.

B.2. Further reading recommended

3. Application in derivatives pricing:

Reservoir Computing for Option Pricing (2019) - SSRN

Deep Learning for Derivative Pricing (2018) - SSRN

Pricing Options with an Adaptive Neural Network (2020) - SSRN

These literatures explore ***various Neural network-based derivatives pricing methods***, which have similarities with Neural SDE models, and can help students understand the application status and development trend of machine learning in derivatives pricing.

B.3. Further reading recommended

4. Other financial applications:

Deep xVA Solver - A Neural Network Based Counterparty Credit Risk Management Framework (2020) - arXiv

A Deep Learning Approach to Automatic Option Hedging (2020) - SSRN

Pricing Derivatives with Signature Payoffs and Reservoir Computing (2020) - SSRN

This literature shows the application of Neural SDE ideas in ***other financial fields***, such as ***credit risk management, option hedging, and pricing of non-standard derivatives***, which helps students to broaden their horizons and understand the broad application prospects of Neural SDE models.

5. Quantitative trading and investment strategies:

Deep Reinforcement Learning for Trading (2020) - SSRN

Deep Learning for Portfolio Optimisation (2020) - SSRN

Generating Realistic Stock Market Order Streams Using Generative Adversarial Networks (2018) - ICML

This literature explores the ***application of deep learning*** in ***quantitative trading and investment decisions***. Although not directly related to Neural SDE, it reflects the current trend of widespread application of machine learning in the financial field and can help students develop ideas to extend Neural SDE models to a broader application scenario.

C. QUIZ

C.1. QUIZ-1: Application of Neural SDE in Option Portfolio Risk Management

- BG: Let's say you're a ***risk management*** intern at a hedge fund. Your advisor is interested in the ***superior performance of Neural SDE*** models in ***derivatives pricing*** and wants to understand the ***potential application*** of this model to option portfolio risk management.

- Q: Explain in ***1-2 paragraphs*** the ***advantages of Neural SDE*** over traditional pricing models and propose a ***specific application scenario*** of how it would help you better manage the risk of your options portfolio.

QUIZ-1: MY ANSWER

- The biggest advantage of Neural SDE over traditional pricing models is that it can better ***handle complex nonlinear relationships***. This model uses ***neural networks to learn the patterns of asset price changes*** and can automatically find ***hidden patterns from historical data***. This allows Neural SDE to predict asset prices more accurately, while also significantly speeding up the speed of pricing.

- Neural SDE can be a big help when managing the risk of an option portfolio. Using this model, we can quickly calculate a ***reasonable price for each option in the portfolio*** and ***how sensitive they are to various risk factors***. With this information, we can ***monitor the exposure of the portfolio*** in real time and ***adjust our position to avoid the risk if the risk exceeds the limit***.
- Especially ***in the case of sharp market fluctuations***, ***Neural SDE*** can also ***simulate a large number of possible scenarios***, helping us to ***evaluate the losses that the portfolio may suffer in extreme cases***. This is very helpful for ***stress testing***.

QUIZ-1: AI ANSWER

1. Flexibility and Nonlinearity: Neural SDEs can capture complex, nonlinear relationships between underlying assets and option prices. Unlike traditional models (such as Black-Scholes), which assume constant volatility and linear dynamics, Neural SDEs can adapt to changing market conditions and handle more intricate features.

2. Data-Driven Calibration: Neural SDEs can be trained on historical market data, allowing them to learn from observed prices and implied volatilities. This data-driven calibration enhances their accuracy, especially in volatile markets or during events like earnings announcements or geopolitical shifts.

3. Application Scenario: Option Portfolio Risk Management

- Imagine you’re managing a portfolio of options across various underlying assets. Here’s how Neural SDEs could improve risk management:
- Suppose you hold a mix of call and put options on tech stocks. Traditional models might struggle with sudden volatility spikes or non-normal price distributions. However, a Neural SDE could dynamically adjust its parameters based on real-time market data, capturing the changing volatility patterns. By incorporating Neural SDEs into your risk assessment, you’d gain more accurate estimates of option prices, better hedging strategies, and improved risk exposure management. Additionally, Neural SDEs could help you identify anomalies or mispricings, allowing you to exploit arbitrage opportunities effectively. Overall, their adaptability and data-driven nature make Neural SDEs a powerful tool for option portfolio risk management.

QUIZ-1: Comparison?

- Benefits Clarification: My answer mainly emphasized that Neural SDE is better at handling complex non-linear relationships, automatically discovering hidden patterns from historical data, making more accurate predictions and pricing faster. The AI answer, in addition to mentioning nonlinearity, also points out that Neural SDE can adapt to changing market conditions and improve accuracy in markets with high volatility through data-driven calibration. AI's answer is more comprehensive.
- Scenario: I mentioned that Neural SDE can monitor portfolio exposures and perform stress testing in real time. AI Answers envisages a specific scenario of managing a portfolio containing call and put options on tech stocks, illustrates how Neural SDE can tune parameters based on real-time data to obtain more accurate option pricing, better hedging strategies, and risk management. It also mentioned identifying pricing anomalies and arbitrage opportunities. As you can see, the AI example is more specific and vivid.
- Paragraphing: I will first talk about the advantages, then talk about the application, and finally emphasize the advantages of Neural SDE in the extreme market. AI is structured in a 1-2-3 way, explaining the two main advantages and then discussing the application of AI to portfolio risk management in detail. The logic is a little clearer.
- Language expression :AI's expression is more rigorous and neat, and the words are more professional. My language is more colloquial.

- Overall, the AI answers were better in terms of comprehensiveness of content, vividness of examples, clarity of structure, and professionalism of language. But my answer also captures the key point and can serve as a good foundation, which can be supplemented and polished properly to become a good answer.

C.2. QUIZ-2: Incorporating market microstructure information to improve SDE models

- BG: ***Traditional SDE*** models usually assume that ***asset prices move smoothly in continuous time***. However, in ***real financial markets***, ***price changes*** are often ***discrete and limited by the minimum unit of change (tick size)***. In addition, the ***bid-ask spread*** also reflects the ***liquidity conditions of the market***.

- Q: What do you think is the ***significance*** of ***including*** such ***market microstructure information*** in the ***SDE model***? Give a ***concrete idea*** of ***how the model could be improved*** to ***account for*** the ***dispersion of price movements*** or the ***effect of bid-ask spreads***.

QUIZ-2: MY ANSWER

- In ***real trading***, the ***price changes are discrete***, and the price cannot be changed arbitrarily, but must be ***changed by integer multiples of the minimum tick size***. So the price shows a ***step jump feature***. If the model can describe this discrete nature, it can ***more accurately capture the characteristics of price fluctuations in the short term***, thus improving the accuracy of option pricing and risk management.
- In addition, there is a gap between the highest price a buyer is willing to pay and the lowest price a seller is willing to accept. This ***bid-ask spread*** reflects the ***liquidity of the market***. A ***larger spread*** indicates a ***higher transaction cost*** and a ***less liquid market***. When pricing options, if the impact of the spread can be taken into account, the actual value of the option and the cost of hedging can be more reasonably evaluated. We can try to ***add a new stochastic process (as a feature) to the SDE model*** to ***describe the variation of the spread***. This spread process can ***interact with the volatility process of asset prices***. That way, ***liquidity will naturally be taken into account in pricing***.

QUIZ-2: AI ANSWER

1. Discrete Price Movements: Real asset prices don’ t move continuously; they jump in discrete steps due to tick sizes. Ignoring this can lead to inaccuracies in pricing and risk assessment.

2. Bid-Ask Spread: The bid-ask spread reflects liquidity conditions. A wider spread implies higher transaction costs. Failing to account for this affects option pricing and hedging strategies.

3. Concrete Improvement: Discrete-Time SDE with Bid-Ask Spread:

- Consider a discrete-time SDE that models the asset price evolution over small time intervals (e.g., minutes or seconds).
- Price Movement: Instead of continuous diffusion, use a discrete jump process (e.g., Poisson jumps) to capture tick-by-tick price changes.
- Bid-Ask Spread: Modify the drift term to account for bid-ask spread. For example:
- If the bid price is B and ask price is A, set the drift as $\mu=21(A+B)$.
- Adjust volatility to reflect the spread’ s impact on price dynamics.
- Option Pricing: Use this improved SDE to price options. The bid-ask spread affects option premiums, especially for illiquid assets.
- Risk Management: Incorporate bid-ask spread into delta hedging strategies. Optimal hedging requires accurate pricing that considers transaction costs.

QUIZ-2: Comparison?

- There is agreement on discrete price movements. Both answers recognize that price changes in real markets are discrete and bounded by the smallest unit of price change (the smallest quoted unit). If the model can describe this dispersion, it can more accurately capture the characteristics of price fluctuations in the short term.
- Views on bid-ask spreads are also closer. Both answers mentioned that the bid-ask spread reflects the liquidity situation of the market, and a larger spread means that the transaction cost is higher and the market is less liquid. Taking the effect of spread into account when pricing options can provide a more reasonable assessment of the actual value of options and the cost of hedging.

- The idea of improving the model is slightly different:
- My answer suggests that you could try to add a stochastic process to the SDE model to describe the change in the spread, let this spread process interact with the volatility process of the asset price, and thus naturally account for the effect of liquidity in pricing.
- The AI's answer is to consider a discrete-time SDE model, using a discrete jump process (e.g., Poisson jump) to characterize the price change from transaction to transaction, and include the effect of bid-ask spreads in the drift term.
- AI's answer is more specific in terms of option pricing and risk management, mentioning that bid-ask spreads affect option fees, especially for illiquid assets; It is also mentioned that the consideration of bid-ask spread is included in the Delta hedging strategy, and the optimal hedging requires accurate pricing and consideration of transaction costs.

D. REFLECTION

- 1. Selection and organization of content: When writing the teaching materials, I found that the theoretical basis and mathematical derivation involved in the Neural SDE model were relatively complex, and it was difficult to fully cover them in the limited class hours. Therefore, I must carefully consider the content selection, and choose those parts that best reflect the core idea of the model and the most practical value to focus on. At the same time, I also need to carefully design the organizational structure of the content to ensure that the logic of the previous and subsequent chapters is smooth, and the theoretical part and the practical part echo each other, making the course clear and easy for students to understand and master.
- 2. Selection of reference materials: Neural SDE model is an emerging research field, and there are a large number of related literatures. In the preparation of reference materials, I invested a lot of time and energy to select the most representative papers and books with the highest academic value from the massive literature. I hope that by providing these high-quality reference materials, on the one hand, it can help students expand their horizons and understand the cutting-edge progress in this field; On the other hand, it also provides guidance for their further research and exploration. I recognize that as a teacher, I have a responsibility to bridge the gap between classroom knowledge and the frontiers of the subject.
- 3. The importance of teaching reflection: In the process of writing the syllabus and handouts, I always remind myself to think from the perspective of students. I put myself in the shoes of a student who was first exposed to Neural SDE models. What confusion and obstacles might I encounter? This perspective makes me pay more attention to the explanation of the concept, the presentation of the derivation process and the disassembly of the steps of the code implementation in the instructional design. At the same time, I would like to reserve enough classroom discussion and Q&A interactive sessions to encourage students to speak freely and solve their doubts in time. I am well aware that teaching reflection is a process of repeating and constantly self-updating. Only by maintaining the consciousness of reflection at all times can we continuously improve the quality of teaching.

I will upload all the course materials (including codes) to GITHUB, feel free to browse and ask about them!

https://github.com/HarryCCC/Various-Codes/tree/6e9639c499b35b25877d774b8b17a68be748049c/%E9%87%91%E8%9E%8D/NeuralSDE%E6%9C%9F%E6%9D%83%E5%AE%9A%E4%BB%B7

E. REFERENCES

- Gierjatowicz, P., Sabate-Vidales, M., Šiška, D., Szpruch, L. and Žurič, Ž., 2020. Robust pricing and hedging via neural SDEs. arXiv preprint arXiv:2007.04154.
- Businessresearchinsights.com. 2024. Derivatives Market Size Analysis, 2022-2029. [online] Available at: <https://www.businessresearchinsights.com/market-reports/derivatives-market-100600> [Accessed 3 April 2024].
- Effectivesoft.com. 2024. Machine Learning in Finance: Use Cases and Applications - Effective Software. [online] Available at: <https://www.effectivesoft.com/blog/machine-learning-in-finance.html> [Accessed 3 April 2024].
- Morale, D. and Capasso, V., 2005. An interacting particle system modelling aggregation behavior: From individuals to populations. Journal of Mathematical Biology, 50(1), pp.49-66.
- Implied and Local Volatility Surfaces for South African Index and Foreign Exchange Options, Journal of Risk and Financial Management ISSN 1911–8074.
- Verma, J., 2020. Which one is your volatility? Constant, local or stochastic?. [online] Medium. Available at: <https://towardsdatascience.com/which-one-is-your-volatility-constant-local-or-stochastic-61508ef560c1> [Accessed 3 April 2024].
- Quantpy.com.au. 2024. Stochastic Volatility Models Archives - quantpy. [online] Available at: <https://quantpy.com.au/category/stochastic-volatility-models/> [Accessed 3 April 2024].
- Bergna, R., Opolka, F., Liò, P. and Hernandez-Lobato, J.M., 2023. Graph Neural Stochastic Differential Equations. arXiv preprint arXiv:2308.12316.
- Bradleyboehmke.github.io. 2024. UC Business Analytics R Programming Guide. [online] Available at: <http://uc-r.github.io/feedforward_DNN> [Accessed 3 April 2024].