Final Report

Hyperparameter Tuning for Asian Option Deep Hedging

2017151006 이수민

Index

Introduction

시장에 참여하는 투자자들은 옵션의 가격을 효율적으로 계산하는 것과, 포트폴리오의 위험성을 최소한으로 하는 방법에 대한 궁금증에 필연적으로 직면하게 된다. 옵션의 가격을 효율적으로 계산해야 투자의 성공률을 높일 수 있고, 포트폴리오의 위험성을 효과적으로 제거해야 악영향에 의해 손해를 보는 상황에서 손해를 최소화할 수 있기 때문이다.

포트폴리오의 위험성을 제거하는 전략으로 폭넓게 쓰이는 것이 Hedging이다. Hedging은 여러 방법으로 포트폴리오에서 위험성을 제거하는 전략이다. Option을 통해 Hedging하는 방식을. Option을 이용한 Hedging은 통계적, 수학적인 공식을 활용하는 전통적인 방식인 Greek Hedging, Delta Hedging이 존재하였으나, 컴퓨터 공학의 발전으로 새로운 Hedging 전략이 등장하게 되었다. 3세대 Hedging으로도 불리는 Deep Hedging은 AI가 데이터를 기반으로 파생 상품의 위험성과 가격 책정을 하는 Hedging이다.

3세대 Hedging과 기존 Hedging의 가장 큰 차이점은 3세대 Hedging은 입력 변수들의 관계를 통해 Option값을 책정하는 중간 계산 과정에만 사용되는 것이 아니라, 입력 변수들의 관계를 가중치를 통해 조정하고, 이를 통해 가장 최적의 Hedging 결과를 내는 방법을 알아낸다. 즉 어떤 Option Pricing 공식을 주입하여 이용하는 것이 아니라, Option Pricing을 하는 최적의 공식 자체를 만들어내고, 이를 활용하는 것이 3세대 Hedging이라는 것이다.

이 글에서는 2세대, 3세대 Hedging인 Delta Hedging과 Deep Hedging을 KOSPI의 시가총액이 가장 큰 기업 9개를 대상으로 실제로 진행해본다. 진행 결과를 통해 Insight를 얻어보고, Deep Hedging의 성능을 높일 방법에 대해 논의해보겠다.

Theoretical Foundation

1. Transaction

Transaction에는 4 종류가 존재한다. 첫 번째는 Product와 Money의 교환이 현 시점에서 동시에 이루어지는 경우이다. 두 번째는 판매자가 먼저 Product를 제공하고, 미래에 구매자가 Money를 지불하는 경우이다. 이 경우 Product의 가치, 가격이 시간에 따라 변화하기 때문에 구매자의 지불액이 미래 상황에 따라 변화한다. 이에 대한 예시로, 주택을 구매할 시에 은행에서 대출을 받아 주택을 구매하고, 원금과 이자를 같이 은행에 상환하는 것이 있다. 세 번째는 구매자가 Money를 먼저 판매자에게 지불하고, 미래에 Product를 받는 경우이다. 이는 과거 1400년대 향신료 무역을 하던 시절부터 자주 사용된 개념으로, 주식과 같은 개념을 공유한다. 네 번째는 상호 동의 하에 미래 어떤 시점에 상품의 가격을 결정한 다음, 약속된 시점에 약속된 Price로 Product를 거래하는 경우이고, 이를 '선물 거래'라 한다.

2. Payoff Diagram

Payoff Diagram이란 미래의 이익 혹은 손해를 그래프로 표현한 것으로, 투기, 투자의 목적으로 Transaction을 할 시에 유용하다. 미래 어느 시점에 Product를 1.1달러에 구매하기로 판매자와 구매자가 계약을 했을 시, 그 시점에 Product의 실제 가격에 따른 서로의 입장 차이는 아래와 같다.

<판매자>

판매자의 경우 계약한 가격이 그 시점의 실제 가격보다 낮으면 손해를 보고, 높으면 이득을 본다. 판매자의 입장에서 Payoff Diagram은 계약한 가격을 x절편으로 가지고, 기울기가 -1인 우하향 1차함수 형태를 띈다. 거래에서 이러한 Payoff Diagram을 가진 사람을 Short Position에 있다고 한다.

<구매자>

구매자의 경우 계약한 가격이 그 시점의 실제 가격보다 낮으면 이득을 보고, 높으면 손해를 본다. 구매자의 입장에서 Payoff Diagram은 계약한 가격을 x절편으로 가지고, 기울기가 1인 우상향 1차함수의 형태를 띈다. 거래에서 이러한 Payoff Diagram을 가진 사람을 Long Postion에 있다고 한다.

3. Interest

Interest는 현재 재화의 가치가 시간에 따라 변화한다는 개념으로, 예시를 통해 이해하자면 "현재의 10달러의 가치는 미래의 11달러와 동일하다" 라는 개념이다. Interest는 Position에 따라 받아들이는 바가 달라질 수 있다. 이를 또한 예시를 따라 설명하자면, Long Position에서의 Interest는 10달러를 예금하면, 미래 어느 시점에서는 11달러가 되어있는 것을 뜻한다. 그

리고 Short Position에서의 Interest는 10달러를 대출하면, 미래 어느 시점에서는 11달러를 갚아야 하는 것을 뜻한다.

기본적인 Interest의 형태는 Credit Risk(상황에 따라 이자를 상환을 못 받는 경우가 생길 위험도)를 가정하지 않으며, 가정할 시에 상황과 대상에 따라 변화한다. Interest가 존재함에도 대출의 개념이 존재 가능한 이유는, Interest는 고정되어 있기 때문이다. 만약 대출한 금액으로 이자율 이상의 수익을 얻을 경우엔, 대출을 받은 사람은 이득을 보는 구조이기 때문이다.

Interest의 Credit Risk에서 볼 수 있듯이, Risk는 그 자체로 어떤 가치를 가진다. 만약에 돈을 예금할 시에 1기간 후 50%의 확률로 22달러를 받고, 50% 확률로 0달러를 받는다고 가정해보자. 1기간 후의 E(x)값이 11달러이므로 만약 이자율을 10%로 가정할 시 10달러를 예금한 것이겠지만, 50%의 확률로 예금액을 다 잃을 Risk를 감수하고 예금을 한 것이므로 실제로는 10달러보다 낮은 돈을 예금한 것이 합리적인 상황이다.

4. Option

Option은 두 가지가 존재한다. Call Option은 미래에 상품을 살 수 있는 권리로, 보통 미래에 가격이 오른다고 예상한 경우 매입한다. Put Option은 미래에 상품을 팔 수 있는 권리로, 보통 미래에 가격이 내린다고 예상한 경우 매입한다. Long Position과 Short Position을 적용하여 미래 예상 수익 그래프를 경우 우측과 같다.

Option 거래의 특징으로는 권리를 행사했을 시 자신이 손해를 볼 상황이 오면 Option 거래로 얻은 권리를 포기할 수 있다는 것이다. 만기일에 기초 자산의 가격과 행사 가격의 관계를 파악했을 때에 Option 을 행사하지 않은 경우에 손해를 최소화 할 수 있다고 판단할 경우, 행사를 멈춰버리고 손해를 0 으로 만들 수 있다. 그 예시를 들면, 현재 1 억짜리 차를 미래에 1 억에 구매할 수 있는 권리를 구매한다. 만약 미래에 차의 가격이 1 억보다 낮게 된다면, 차를 구매할 권리를 포기하고, 그냥 차를 구매하지 않으면 된다. 그러면 실제로 옵션의 가격을 고려하지 않는다면, 옵션에 대한 손실액은 0 이하로 떨어질 수 없다.

하지만 이런 논리라면 구매자(Long Position)에 있는 사람은 상황에 따라 이득은 얼마든지 얻을 수 있지만 그 어떤 상황에서도 손해를 보지 않는다. 실제 상황에서는 Option이라는 권리 자체도 그 가치를 지니고 있다. 즉, Option을 구매할 때에 그 가치만큼 일정 재화를 지불해야 하므로, Option 권리를 포기한다면 구매했을 때의 사용한 재화만큼의 손해를 감수해야 한다.이에 따라 사례를 실제 상황에 더 맞도록 다시 구성한다면 아래와 같다:

사례) 현재 1억짜리 차를 미래에 1억에 구매할 수 있는 권리를 100만원에 구매한다. 만약 미래에 차의 가격이 1억 2000만원이 되었다면, 그 차를 구매함으로 1900만원의 이득을 보는 것이다. 만약 차의 가격이 9000만원처럼 1억 100만원보다 아래일 경우(차 값이 1억 100만원일때 정확한 본전이다.) 구매할 수 있는 권리(옵션)을 포기하고 100만원(Option Price)만큼의 손해를 얻는다.

5. Asian Option

표준적인 옵션의 거래에서 변형된 이색 옵션 중 하나이다. 표준 옵션의 경우 만기일 당일에 기초자산의 가격(market price)와 미리 정한 행사가격(striking price)에 의해 현금이 움직이게 되는데, 기초 자산의 가격 또는 행사가격을 일정 기간 동안의 기초자산의 평균 가격으로 대체하는 것이다. 일정 기간은 총 기간이 될 수도 있고, 일정 길이의 기간으로 특정도 가능하다. (예시: 국내 기업이 1년간 일정 규모의 물품을 달러로 수입할 시에, 모든 거래에 개별적으로 달러/원콜옵션을 매입하지 않고, 그 해의 평균 환율을 일정 크기 아래로 보장하는 평균환율 콜옵션을 매입하거나, 달별로 평균환율 콜옵션을 매입한다.)

Asian Option의 장점은 만기에 가까워질수록 기초 금융상품이 시장이 조작할 위험을 감소시키고, 급격한 변화로 인한 예기치 못한 손해를 완화할 수 있다. 또한 변동성이 낮은 상품 시장에서 주로 사용하기 때문에 옵션의 가격이 European, American Option보다 저렴하다.

Asian Call 과 Put은 아래 8가지 특성으로 분류가 가능하다.

- Average Strike Option(Fixed Strike) vs. Average Rate Option(Fixed Pricing)
- Arithmetically vs. Geometrically Averaging
- Discrete vs Continuous Averaging
- American vs European exercise

여기서는 Discrete, Arithmetically or Geometrically averaging, average strike option으로 call과 put의 가격을 결정하는 Asian Option을 사용할 것이다. Average는 아래의 공식으로 계산한다:

$$\text{Arithmetic, } A(0,T) = \frac{1}{N}\sum_{l=1}^{N} S(t_l)$$

$$\text{Geometric, } G(0,T) = \exp\left(\frac{1}{N}\sum_{i=1}^{N} \log\left(S(t_i)\right)\right)$$

$S(t_l)$: 해당 시점에서의 현물 가격

N = 동일한 크기로 나누어진 간격$\left(\text{기간}\right)$

T = 만기까지의 총 기간

6. Option Pricing

European Option이나 그것의 변형 형태인 Geomeric Asian외 등등에는 폐쇄적인 환경 내에서 옵션의 가격을 결정하는 방법이 있으나, 다른 옵션들은 이를 계산하는 방법이 명확히 존재하지 않는다. 그런 옵션들에는 계량된 통계 모델을 사용한다. 가장 일반적으로 사용하는 기술은 이항 옵션 가격 책정 모델과 Black-Scholes 모델 이 두 가지이다.

6-1 이항 옵션 가격 책정 모델

이항 옵션 가격 책정 모델(Binominal Option Pricing Model, 이하 BOPM)은 이항 트리를 사용하여 기초 자산의 가격 변화를 나타낸다. BOPM은 차익거래가 없는 시장에서 기초 자산의 가격이 특정 금액만큼 증가하거나 감소한다고 가정한다. 단순한 모델 구조로 인해 다양한 옵션에 활용할 수 있는 범용성을 갖춘 것이 특징이다.

BOPM에서 한 기간동안 Call Option의 가격은 아래의 공식에 따른다:

$$C = e^{-rh}\left(C_u \frac{e^{(r-\delta)h} - d}{u - d} + C_d \frac{u - e^{(r-\delta)h}}{u - d}\right)$$

$u = up$
$d = down$
$C_u = up$ 일때 옵션의 가격
$C_d = down$ 일때 옵션의 가격
$r =$ 무위험 수익률$\left(\text{순수 이자율}\right)$
$\delta =$ 배당 수익률
$h =$ 단위 기간(시간)

BOPM은 여러 옵션 가격 책정에 널리 사용되지만, 기초 자산의 가격이 두 가지 경우의 수로만 움직일 수 있다는 단점이 있다. 이는 현실 세계와는 괴리를 보일 수 있다. 이를 해결하기 위해서는 한 기간으로 잡는 단위 시간을 매우 짧게 가져가는 것이다.

6-2 Black Scholes 모델

Black과 Scholes가 주식이 기하학적 브라운 운동을 따른다고 가정하여 옵션 가격의 변동을 설명하고, 가격을 책정하는 방법을 고안한 모델이다. Richardson(2009)에 따르면, 기초 자산의 가격은 다음 확률적 미분 방정식으로 설명된다:

$$dS = \alpha S dZ + (\mu - \delta)S dt$$

$S =$ 기초자산 가격
$\sigma =$ 변동성
$\mu =$ 기대 수익
$\delta =$ 위 기초자산에 따른 무위험 수익률$\left(\text{순수 이자율}\right)$
dS : 자산 가치의 변화
dZ : 위너 확률 공정 (Wiener Process)
dt : 시간의 변화

위너 확률 공정(Wiener Process)은 시간차 Δt의 증분의 확률 분포가 평균 0, 분산 Δt의 정규 분포를 이루고, 각 증분이 서로 독립이며 그 궤적이 연속적인 연속 시간 확률 과정으로, 여기서는 Mcdonald의 정의를 빌려 연속적인 움직임으로 연속적인 시간에 발생하는 무작위 확률 발생 과정으로 정의한다.

위의 방정식에서 $\delta = 0$으로 가정하면, 옵션 $V(S,t)$에 대해 아래와 같이 풀 수 있다:

$$dV = \sigma S V_S dX + \left(\mu S V_s + \frac{1}{2}\sigma^2 S^2 V_{SS} + V_t\right) dt$$

위의 방정식을 활용하여 하나의 V(S, t) 롱 옵션을 매수하고 기초 자산의 Δ를 매도하는 포트폴리오 Π를 구성하여 미분하면 다음과 같다:

$$\therefore \Pi = V - \Delta \cdot S \implies d\Pi = dV - \Delta dS$$

위의 포트폴리오에서 차익거래가 일어나지 않고, 무위험 수익률로 투자하고 매각한다고 가정함으로 포트폴리오를 조정하여 아래의 Black Scholes Equation(BSE)을 도출해냈다:

$$V_t + \frac{1}{2}\sigma^2 S^2 V_{SS} + rSV_S - rV = 0$$

Black Scholes Equation을 도출해나가는 과정에서 사용한 가정은 아래와 같다:

- 기초 자산은 일정한 변동성을 가진다.
- 발행한 주식수가 일정하다.
- 배당금은 존재하지 않는다.
- 주식의 가격은 평균 μ과 표준편차 σ를 갖는 로그 정규분포를 따른다.
- 무위험율이 일정하다.
- 시장에 참여하고 있는 사람들은 무위험 이율로 차입 또는 대출이 가능하다.
- 거래 비용이 없다.

7. Hedging

자산의 불리한 가격 변동 위험을 줄이기 위한 투자 전략이다. 포트폴리오나 포지션에 악영향을 끼치는 상황이 도래했을 때에 손해를 최소화 하는 전략으로, hedging을 하지 않은 사람들을 naked position에 있다고 한다. Hedging을 하는 방식에는 여러 가지가 있지만, 여기서는 옵션을 통해 Hedging하는 전략에 집중할 것이다.

7.1 Option Greeks

Option Greeks는 다른 입력 변수에 따른 옵션 가격의 민감도를 측정한다. Greeks는 위험 노출도나 Hedging을 측정하는 데에 광범위하게 사용된다. 주요 가정은 한번에 입력 변수가 하나만 변경되고 나머지 변수는 일정하게 유지되는 것이다. Option의 가격을 책정하는 데에 사용하는 Greeks 변수는 아래와 같다:

- Delta, Δ : 기초 자산 가격 대비 옵션 가격의 민감도
- Gamma, Γ : 기초 자산 가격 대비 델타의 변화 정도
- Vega : 변동성에 대한 옵션 가격 민감도
- Theta, θ : 만기일에 대한 옵션 가격 민감도
- Rho, ρ : 무위험 수익율에 대한 옵션 가격 민감도
- Psi, Ψ : 배당 수익율에 대한 옵션 가격 민감도

Greeks 변수들로 위의 Black Scholes Equation의 변수들을 새롭게 정의한다면, 아래와 같다:

V_s = option의 delta
V_{SS} = option의 gamma
V_t = option의 theta

7.2 Delta hedging

Call Delta는 아래 공식을 따른다:

$$\text{Euro Call delta}, \Delta = e^{\delta T} N(d_1)$$

Delta hedging을 통해 위험을 최소화하려면, 우선적으로 생각해야 하는 중심 아이디어는 적절하게 hedge한 position을 가질 경우 무위험 수익율을 가져가야 한다는 것이다. Call delta의 공식을 통해 옵션의 가격을 계산할 수 있으며, delta값을 통해 옵션의 기초 및 차입금 등가

물에서의 포지션을 정하는 데에도 사용이 가능하다. 현물 가격 S로 Δ개의 주식을 구매하고 $Ke^{-\delta T}N(d_2)$만큼을 대출을 받는다면, 전체 포트폴리오의 cost는 아래 공식으로 표현이 가능하다:

$$Se^{-\delta T}N(d_1) - Ke^{-rT}N(d_2)$$

위의 공식은 $\delta = 0$ 일 때 Black-Scholes에서 Call option의 가격과 동일하다. 따라서 위의 공식을 통해 기초 자산을 구매하고 무위험 수익률로 차입하는 콜 옵션을 종합적으로 생성할 수 있다.

7.2.1 Example of Delta Hedging

딜러가 아래 표와 같은 정보를 가진 주식에 대해 Euro Put Option을 하나 판매하려 한다고 가정한다. 딜러는 position을 hedge하여 균형을 이루는 포트폴리오를 구성한다고 가정한다. Position을 hedge하기 위해 시장에 참여한 사람은 기초자산에 Long position을 취하고 있고, Δ개의 상품을 구매할 것이며, Put option 상품은 매일 시장에서 판매된다고 가정한다.(위의 가정은 동적 hedging의 한 예시이다.) 아래 주식에 대한 정보는 한국금융위원회의 주식시세정보 API를 활용하여 구성하였고(Code 1), 삼성전자의 주식에 대한 상품이다.

상황 1 - <Put Option 정보>

행사 가격, K	78800.0원	배당 수익률, δ	0.0%
변동성, σ	3.73%	시작일	2022.01.11
무위험 수익률(이자율), r	3.86%	만기일	2022.02.24

Example Day 0: 2022.01.11에 삼성전자는 78900.0원으로 마감하였다. Put Option의 가격은 위의 Euro Option Pricing에 따르면 약 281.1원이 된다(Code 2). Hedge하기 위해 딜러는 $\Delta(-0.3157)$개의 주식을 판매해야 한다. 그럼 총 투자액은 아래와 같다:

$$(-0.3157 \times ₩78900.0) - ₩281.1 = -25189.83$$

무위험 수익률(이자율)이 3.86%이므로 하룻밤 동안 딜러는 $₩25189.82 \times \left(e^{\frac{0.0386}{365}} - 1\right) = ₩2.67$의 이익을 얻었다.

Example Day 1: 2022.01.12에 삼성전자는 78900.0원으로 마감하였다. 이때 Put Option의 가격은 약 279.9원이 된다(Code 3). Hedge하기 위해 딜러는 $\Delta(-0.3169)$개의 주식을 판매해야 한다. 당일 수익은 다음과 같다:

$$₩(25189.83 + 25283.31) \times \left(e^{\frac{0.0386}{365}} - 1\right) = ₩5.33$$

Example Day 2: 2022.01.13에 삼성전자는 77900.0원으로 마감하였다. 이날의 당일 수익은 아래와 같이 계산한다.

Put Option 수익(₩)	281.1 + 279.9 – 540.3	20.7
Short Position의 주식 판매 수익	(77900 – 78900) * (-0.3157-	632.6

(₩)	0.3169)	
총 수익(₩)		653.3
무위험 수익(₩)	2.67 + 5.33	8
당일 수익(₩)		661.3

7.3 Hedging Asian Option

Asian Option 자체가 European Option의 변형 Option이기 때문에 Δ중립 포트폴리오를 위해 기초 자산을 구매하거나 매도하는 Position을 잡으며 자금을 투입하여 동적 Hedge를 실행한다는 기본 전제는 위의 European Option을 활용한 동적 Hedging과 공유한다.

Asian Option을 Hedging하기 위해 위의 예시와 같은 상황(상황 1)에서 10일간 Asian Option Hedging을 진행한다.(코드4) 여기서 Asian Option은 전체 기간의 geometric average를 이용하여 strike price를 결정한다.(Fixed Strike = ₩78,247)

Geometric average Asian Option(삼성전자)				
날짜	2022-01-11	2022-01-12	2022-01-13	2022-01-14
당시 주식 가격(₩)	78900.0000	78900.0000	77900.0000	77300.0000
Call 가격(₩)	864.9483	857.9172	843.8214	822.6042
Option Delta	0.7798	0.7865	0.8005	0.8228
총 투자금(₩)	60661.2717	61196.9328	61515.1286	62779.8358
Call 수익(₩)		7.0311	14.0958	21.2172
선물 수익(₩)		0	-786.5	-480.3
자본 수익(₩)		7.0311	-772.4042	-459.0828
무위험 수익(이자)(₩)		-6.4155	-6.4721	-6.5058
당일 이익(₩)		0.6156	-778.8762	-465.5886
총 이익(₩)			-778.2606	-1243.8492

해석은 다음과 같다.

Day 0:
주식의 가격이 78900원이고 Call Price가 864.9483원이다. Option Delta가 0.7798이라 함은 기초 자산의 가격이 1 변할 때 옵션 가격은 0.7798 변한다는 의미이다. 내가 Call Option을 하나 가지고 있는 상황에서 Delta를 0으로 유지하려면 주식을 0.7798 매입한다. 총 투자금은 (주식 매입에 사용한 가격) – (Call Option 가격)이다.

Day 1:
주식의 가격이 변동이 없고 Call Price가 857.9172원이다. Option Delta가 0.7865이므로 주식을 그만큼 매입한다. Call Price의 변동으로 얻은 자본 수익은 (Day 1 Call Price) – (Day 0 Call Price). Day 0에 매입했던 주식은 주가의 변동이 없으므로 선물 거래에서 0원의 수익을 얻는다. Day 0날 투자한 금액에 대한 이자를 지불하면 당일 얻은 총 이익은 0.6156원이 된다.

Day 2:

주식의 가격이 1000원 감소했고, Call Price는 843.8214원이다. 전날에 비해 주가가 1000원 감소했으므로 전날 매입한 주식에 의해 (Day 1 주가 – Day 2 주가) * (Day 1 Option Delta) 만큼 손해를 본다. 나머지 계산은 Day 1과 같다.

8. Artificial Neural Network

Artificial Neural Network(인공 신경망)은 인간의 두뇌 신경세포인 뉴런을 기본으로 한 기계학습 기법으로 하나의 뉴런이 다른 뉴런들과 연결되어 신호를 전달, 처리하는 구조를 본떴다. 입력 데이터는 신호의 강도에 따라 가중치 처리 후에 활성화 함수를 통해 출력되는 데 학습을 통해 가장 결과가 좋은 가중치로 조정된다

인공신경망의 층수를 늘릴 때마다 입력 데이터가 도중에 소멸되어 학습이 잘 일어나지 않는 Vanishing Gradient 현상이나, 데이터의 수가 적을 경우 Overfitting의 위험이 있다. 허나 2000년 중후반부터 pretraining으로 Vanishing Gradient 현상을 해결하고, Initialize Point와 Dropout으로 Overfitting의 방지가 가능해지자, 기존 인공신경망의 층수를 매우 크게 늘려 복잡한 상황의 연산이 가능하도록 된 것이 Deep Learning이다.

9. Deep Learning

Deep Learning은 Raw Data에서 Algorithm을 만드는 것을 뜻한다. 관련 용어들에 대해 정리하자면 AI, Machine Learning, Deep Learning 순으로 하위 개념에 속한다. 즉 Deep Learning은 인간의 행동을 모방하는 컴퓨터 기술이며, 별도의 명시된 프로그램 없이 학습하는 성능을 지니며, 인공 신경망을 통해 데이터에서 특정 패턴을 추출하는 기술을 뜻한다. 즉 Deep Learning은 Raw Data에서 Algorithm을 만드는 것이다.

Deep Learning이 현재 각광받고 있는 이유는 컴퓨터 학습에서 사람의 개입을 최소화할 수 있기 때문이다. 전통적인 알고리즘 학습은 학습 과정에서 Hands Engineer의 개입이 필요하였다. 허나 Deep Learning은 스스로 데이터를 통해 학습하는 것이 가능하다. Deep Learning에서 활용하는 Neural Network의 개념은 1960년대에 나왔지만, 그 당시에는 Deep Learning에 활용하기에 기술적인 어려움이 컸다. 허나 시간이 지나면서 데이터의 절대적인 양이 기하급수적으로 증가함에 따라 데이터 저장 및 활용 기술이 발전하였고, 학습을 실행할 정도의 발전된 하드웨어, 소프트웨어가 생겨남에 따라 Deep Learning이 실생활에서 활용이 가능하게 되었다.

10. API

Application Programming Interface의 줄임말로, 정의 및 프로토콜 집합을 사용하여 두 소프트웨어 구성 요소가 서로 통신할 수 있게 하는 메커니즘이다.

10.1 REST API

REST는 Representational State Transfer의 줄임말이다. 이는 클라이언트가 서버 데이터에 액세스하는 데 사용하는 여러 함수 집합을 정의한다. REST API를 사용할 경우 클라이언트와 서버가 HTTP를 사용하여 데이터를 교환한다. 클라이언트의 요청은 브라우저에 입력하는 URL과 유사하며, 서버의 응답은 웹페이지의 일반적인 그래픽 렌더링이 없는 일반 데이터이다. REST API를 활용하면 어느 기업의 API를 활용해 유저가 내부 데이터베이스에 유사한 액세스 권한을 부여받을 수 있다.

Practical Application

실제 KOSPI에서 시가 총액이 가장 큰 9개의 기업을 대상으로 Delta Hedging과 Deep Hedging을 진행해보고, Deep Hedging의 Hyperparameter tuning을 통해 Deep Hedging의 성능을 평가하여 어떤 상황에서 Hedging 하는 Model의 성능이 극대화 되는지 확인해본다.

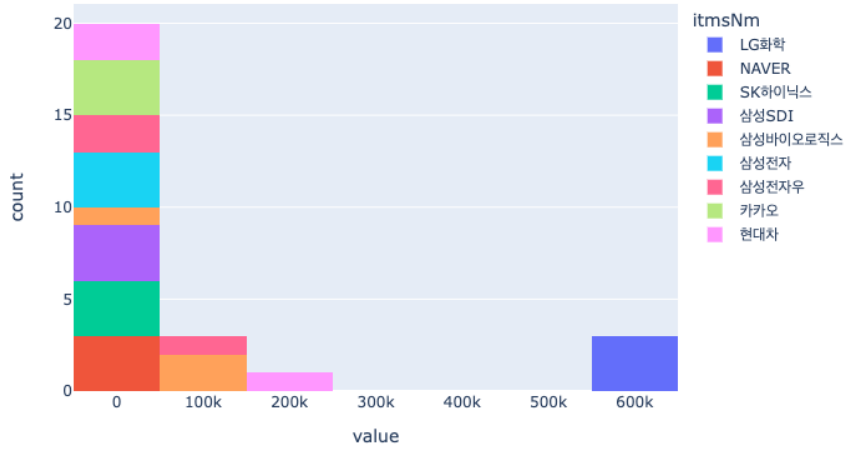
우선 한국금융위원회의 주식시세정보 API를 활용하여 기업의 주가 정보를 얻고(Code 1), Hedging을 진행하는 데에 기본 가정을 한다:

- Black Scholes Equation의 기본 가정을 따른다.
- Option Pricing은 Asian Option을 따른다.
- Geometric average를 사용하며, Striking Price를 G(0, T)로 대체한다.
- 전체 기간을 30개의 기간으로 하며, 1 기간은 1일로 정한다.
- 시작일을 2022년 4월 26일, 만기일을 2022년 6월 10일로 설정한다.
- 변동성, 이자율, 배당금은 <상황 1>과 동일하게 설정한다.

Delta Hedging

<Code 4>, <Code 5>를 통해 Hedging한 결과를 시각화하면 아래와 같다.

Range	LG화학	NAVER	S K 하이닉스	삼성SDI	삼성바이오로직스	삼성전자	삼성전자우	카카오	현대차
0426 – 0610	583076.050783	2.550235e-10	7.748298e-15	-4.371495e-14	5.039998e+04	-1.590177e-14	5.829027e+04	-1.421086e-14	2.896713e-10
0427 – 0613	583076.050783	-3.513114e-14	2.249844e-14	6.390536e-15	5.039998e+04	-2.519381e-14	-1.421085e-14	-2.883585e-14	1.733828e+05
0428 – 0614	583076.050783	8.092295e-15	4.855491e-15	-7.949378e-15	-1.953993e-14	9.209818e-15	2.799667e-14	-2.210934e-14	-1.421085e-14

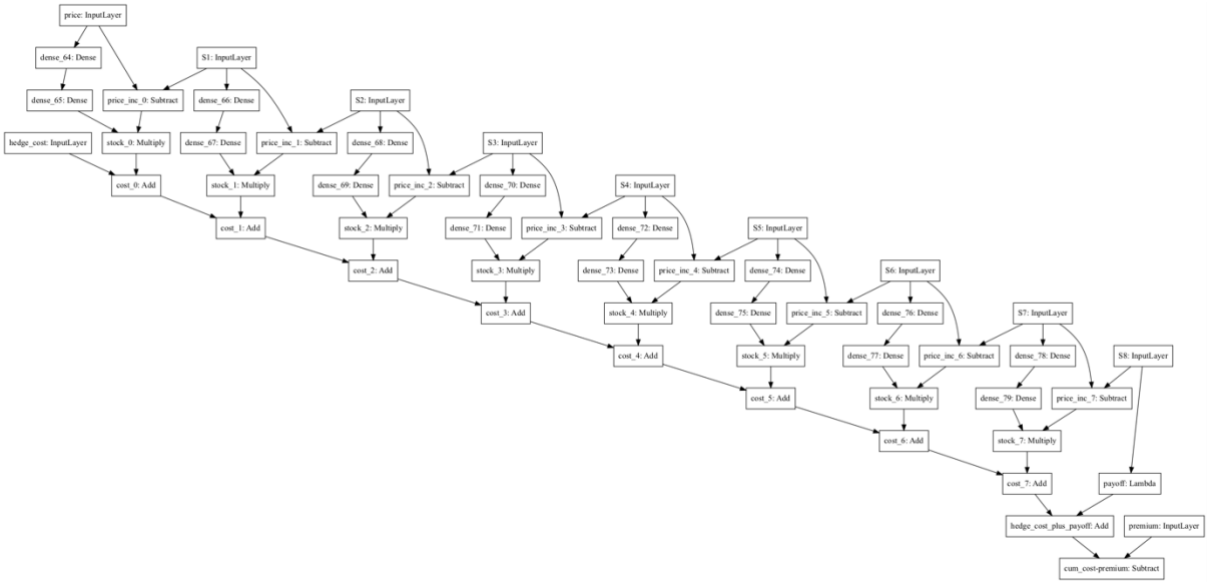


Hedging 결과가 0에 가까울수록 Hedging의 성능이 좋다는 뜻이다. 즉, Risk를 최소화하려면 Hedging을 완료했을 때의 값이 0에 가까워야 한다는 것이다. 위의 그래프를 보면, 24개의

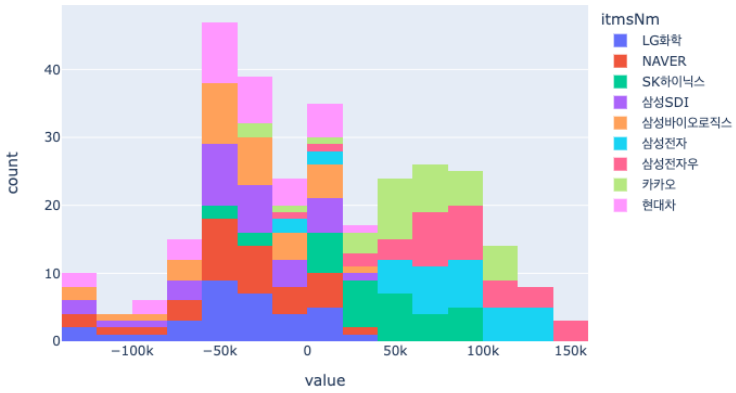
Hedging 결과 중 대부분이 0에 가까운 값을 결과로 가지면서 Hedging이 완료되었음을 알 수 있었지만, 3번의 Hedging이 0과의 거리가 먼 이상치(Outlier)로 결과가 나왔고, 이를 기업으로 나누어 확인해본 결과 3개의 결과 모두 LG화학의 주가를 토대로 Hedging한 결과임을 알 수 있다. 이를 통해 상황에 따라 Delta Hedging이라는 방식이 기업 상황이나 여러 변수들을 통해 유의미한 Hedging 결과를 얻을 수 없다는 것을 알게 되었다.

또한 원화(₩)를 기준으로 Hedging을 하다 보니, Value값 자체도 1,000₩ ~ 10,000₩ 정도를 단위로 크게 나오게 되는데, 달러보다 조금 Hedging의 성능을 가늠할 때에 그래프의 한 구간의 크기를 결정하는 데에 어려움이 있다. 위의 Delta Hedging에서는 9개의 기업을 대상으로 구간을 달리하여 3번씩, 총 27번의 Hedging을 진행하였고, 그 값들의 분포에 따라 한 구간의 크기를 100k로 지정하여 시각화했지만, 아래 Delta Hedging의 경우

이제 Delta Hedging과 같은 가정과 환경에서 Deep Hedging을 진행한다.<Code 6>을 통해 Model을 생성하고 모델을 plot한다.<Code 7>



Hedging이 완료된 결과를 시각화 하면 아래와 같다.<Code 8>



한 기업당 기간을 달리 한 3개의 다른 전체 기간을 총 10번씩, 즉 9개의 기업을 3개의 기간을 가지게 10번의 Deep Hedging을 하여 전체 270 번의 Hedging을 진행해본 결과, 위의 그래프

처럼 0을 중심으로 하였다. Delta Hedging을 했을 때와는 다르게 그래프 한 구간의 크기를 절반으로 줄여서 조금 더 0을 중심으로 Hedging의 결과값이 어떻게 분포하고 있는지를 확인한 결과, Delta Hedging에서는 대부분의 값이 0 – 200k 사이에 분포하였는데, Deep Hedging을 한 결과 -150k에서 150k 사이로 값이 분포되어있는 것을 알 수 있다.

가장 인상깊었던 점은 Delta Hedging을 진행했을 때, 이상값이 나왔던 LG화학의 주가로 Hedging한 결과가 제대로 Hedging되어 분포되어 있다는 것인데, 특정 수식을 사용하지 않고 값을 스스로 학습하여 Hedging하는 Deep Hedging의 방식이 Delta Hedging이 사용하는 수식이나 가정에 따라 Hedging의 결과가 실용적으로 사용할 수 없을 수 있는 것과 더 다양한 상황에서 일반적으로 사용할 수 있는 Hedging 방식이라는 것을 알 수 있다.

Deep Hedging

Option Greeks에서 Greeks에 따라 Hedge 결과가 변화하고 Black Scholes 환경에서 5개의 변수로 Option의 가격을 측정하여 Hedging을 진행하였다면, Deep Learning에서는 여러 Hyperparameter의 조정으로 Deep hedging의 성능을 향상시킬 수 있다. 허나 Computing에서는 원인과 결과로 이어지는 인과 관계로 데이터를 사용하는 것이 아니라, 서로 연관 관계가 있음을 증명하는 상관 관계로 이루어지는 논리 구조를 활용하기에, 성능 향상을 확인할 수 있는 방법은 여러 Hyperparameter를 전부 사용하고 성능 평가를 통해 더 나은 Model이 나왔음을 확인하는 방법을 활용해야 한다. 데이터와 Hyperparameter 중에 Model의 성능 개선을 위해 변화를 줘 볼 요소는 아래와 같다:

- Model Loss Function
- Model Optimizer
- Activation Function
- Batches
- Epochs
- Valid_splits

1. Model Loss Function

Tensorflow에서 지원하는 주요 Loss Function은 아래와 같다.

- MeanSquaredError()

$$L = \frac{1}{2}\sum_{i=1}^{N}(y_i - t_i)^2 \cdots (MSE)$$

y_i = 예측값

t_i = 실제값

- KLDivergence()

$$L = t_i * \log(y_i/t_i)$$

y_i = 예측값

t_i = 실제값

2. Optimizer

Tensorflow에서 지원하는 주요 Optimizer는 아래와 같다.

(1) SGD()

SGD는 확률적 경사 하강법으로, 손실함수의 기울기를 따라가다 최종적으로 손실함수가 가장 작은 지점에 도달하게 하는 알고리즘이다. 단순한 구성으로 계산이 단순하고 명확하나, 랜덤으로

점 하나를 지정하여 시작하는 특성 덕에 목표 지점으로 직선으로 가지 못하는 경우가 있다.

$$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

W = 가중치

$\frac{\partial L}{\partial W}$ = 손실함수의 기울기

η = 학습률

(2) Adagrad()

개별 매개변수에 적응적으로 학습률을 조정하면서 학습을 진행하는 알고리즘으로, 처음에는 큰 폭으로 학습하다가 최적점에 가까워질 수록 학습률을 줄여 학습해나간다. h에 기존 기울기 값을 제곱하여 계속 더해주며, 매개변수 생성 시에 학습률을 조정한다.

$$h \leftarrow h + \frac{\partial L}{\partial W} \odot \frac{\partial L}{\partial W}$$

$$W \leftarrow W - \eta \frac{1}{\sqrt{h}} \frac{\partial L}{\partial W}$$

(3) Adam()

학습률, 일차 모멘텀 계수, 이차 모멘텀 계수 3가지 hyperparameter를 사용한다. 모멘텀 방식과 Adagrad 방식의 결합으로 모멘텀 계수를 따라 최적점을 찾아가다 Adagrad로 갱신 강도가 조정되어서 좌우로 흔들리는 정도를 줄여 최적점에 더 빠르게 도달할 수 있도록 한다.

3. Activation Function

입력신호의 총합을 그대로 사용하지 않고 출력신호로 변환하는 함수. 노드의 활성화 유무를 결정한다. 대표적인 함수로 Sigmoid와 ReLU가 있다. Sigmoid는 참에 가까워지면 0.5~1 사이의 값을 출력하고, 거짓에 가까워지면 0 ~ 0.5 사이의 값을 출력한다. ReLU는 0보다 크면 값을 그대로 출력하나, 0이하의 값은 0으로 출력한다. 두 함수는 모두 이진 분류 함수로, 가중치 매개변수의 적절한 값을 데이터로부터 직접 학습하는 특징을 가진다.

4. Batches

훈련 데이터 중 일부를 무작위로 선택한 데이터를 Batch라고 하며, Batches를 통해 Batch의 크기를 결정한다. Batch의 손실값(실제값 - 예상값)을 최소화하는 목적으로 학습을 진행한다. Batch의 크기가 클 경우 병렬연산 구조를 사용하기에 효과적이고, 작을 경우 학습 도중에 더 많은 가중치 업데이트가 가능하다.

5. Epoch

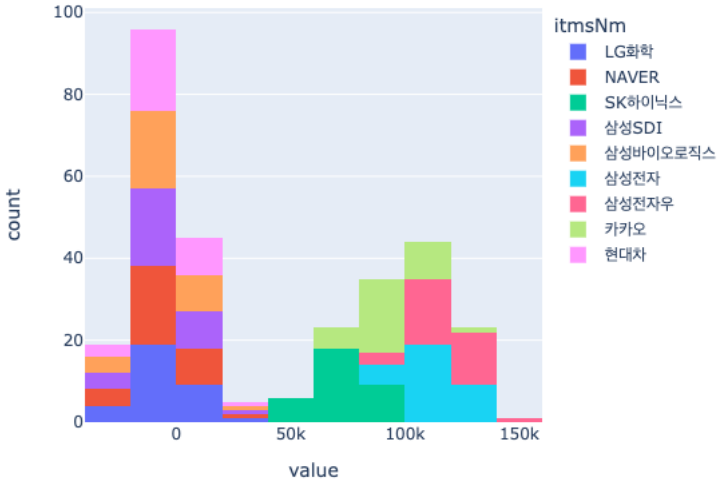
횟수로 학습의 조기 종료를 결정하는 변수가 된다. Epoch의 값이 클수록 값은 정규분포의 모양을 띄나, 연산 시간이 몇 배로 증가한다.

6. Hidden Layer/Node 개수

Hidden Layer 층이 많아질수록 특정 훈련 데이터에 더 최적화시킬 수 있다. 보통은 모든 은닉층에 있는 뉴런의 개수를 동일하게 유지하고, 입력층의 뉴런의 개수보다 은닉층의 뉴런의 갯수를 많게 한다.

Loss Function, Optimizer, Batched, Epoches, Validation_Splits를 H_params를 활용하여 Grid를 설정하고, MeanSquaredError로 Model의 성능을 평가했을 경우, 초기 모델과 성능에서 유의미한 변화를 주지 못했다. 이는 위에서 사용한 Hedging Model의 경우 Model의 구조와

Depth가 단순하고 얕은 편에 속하여 hyperparameter의 변화로 큰 의미있는 변화를 보기 어려웠다. 오히려 위의 hyperparameter의 조정이 아닌 Depth(Delta Dense = 64)에 변화를 주었을 때 아래와 같이 유의미한 변화를 볼 수 있었다.



Depth를 2배로 변화를 주었을 때, 조금 더 많은 결과들이 0에 가깝게 Hedging 된 것을 볼 수 있고, 이는 Hedging의 성능을 향상하는 데에 Depth가 큰 영향을 주었다고 볼 수 있다. 허나 Depth가 너무 과하게 많아질 경우, 주어진 자료에 과하게 학습하여 다른 자료들에는 성능이 떨어지게 되는 Overfitting이 일어날 수 있으므로, 이를 주의해야 한다.

Conclusion

실제 주식 정보를 토대로 Delta Hedging과 Deep Hedging을 진행했을 때, Delta Hedging의 성능은 수학적은 공식을 토대로 Hedging을 하는 것이라, 성능을 향상시킬 수 있거나 결과를 변화를 주는 방식이 한 기간의 크기를 조정하는 것 이외에는 힘든 것이 현실이다 또한 기간의 크기 또한 주식 시장은 열리고 닫는 시간이 명확한 특성 상 하루를 단위 기간으로 잡는 것이 일상적이므로 Hedging 성능을 증진시키려면 Hedging에 사용하는 Option Pricing 공식이 아예 새로운 공식으로 대체해야 한다는 점이 번거롭다.

허나 Deep Learning은 수학적인 공식이 아닌 입력 변수들을 토대로 가중치를 스스로 학습하여 목표에 맞는 값을 출력해 나가기 때문에, Machine Learning의 학습 환경을 Hyperparameter로 조정하여 Hedging의 성능 변화를 기대해 볼 수 있다. 실제로 활용할 수 있는 Hyperparameter는 Loss Function, Optimizer, Epoch, Depth 등등 무한하며, 이러한 Hyperparameter를 활용한 성능 평가의 장점은 기존 Model의 뼈대를 그대로 재활용이 가능하고, 기존 Model과의 성능평가 또한 가능하여 더 쉽고 빠르게 강력한 Hedging을 할 수 있다는 것이다.

Appendix

출처

나홍석, 배원성, 이건길, 이혜영, 고려사이버대학교 Ai & Big Data 연구소. 이기적 빅데이터분석기사 필기. 238-239, 301-303
Buehler, H., Gonon, L., Teichmann, J., & Wood, B. (2019). Deep hedging. Quantitative Finance, 19(8), 1271-1291.
Lakhlani, V. B. (2013). Pricing and Hedging Asian Options.
(Amazon Web Service)[https://aws.amazon.com/ko/what-is/api/]
(investopedia)[https://www.investopedia.com/terms/h/hedge.asp]
(investopedia)[https://www.investopedia.com/terms/b/blackscholes.asp]
(Wikipedia)[https://ko.wikipedia.org/wiki/%EC%9C%84%EB%84%88_%ED%99%95%EB%A5%A0_%EA%B3%BC%EC%A0%95]
(tensorflow)[https://www.tensorflow.org/guide/keras/train_and_evaluate?hl=ko]
(tensorflow)[https://www.tensorflow.org/api_docs/python/tf/keras/losses]
(경기중소기업여신지원센터)[https://m.blog.naver.com/PostView.naver?isHttpsRedirect=true&blogId=pjs667&logNo=80130308335]

Code

<Code 1>

```
import pandas as pd
import numpy as np
import requests
from scipy.stats import norm

### https://www.data.go.kr/iim/api/selectAPIAcountView.do#layer-api-guide
key =
'pM63raeWJv8s%2BtrbKRC9kS5gkPnGsekcRcCoTCdsT36azCw4PJDGb7QyJlhUWA64Bu%2BCetvU5jUCHFo7rjfQig%3D%3D'
url =
'https://api.odcloud.kr/api/GetStockSecuritiesInfoService/v1/getStockPriceInfo?numOfRows=10000&pageNo=1&resultType=json&beginBasDt=20220111&i&beginMrktTotAmt=30000000000000&serviceKey={api_key}'.format(api_key = key)

response = requests.get(url)
body = response.json()
body = body['response']['body']['items']['item']
df = pd.json_normalize(body)
df = df[['basDt', 'itmsNm','clpr']]
df['basDt'] = df['basDt'].astype('datetime64')
df['clpr'] = df['clpr'].astype('float64')
df = df.sort_values(by=['itmsNm', 'basDt'])

df = df[(df['itmsNm']!='LG 에너지솔루션')&(df['itmsNm']!='기아')]
StockInfo = pd.pivot_table(df, values='clpr', index='basDt',
columns='itmsNm')
```

<Code 2>

```
def bsCall(S, K, r, v, q, T):
    d1 = (np.log(S/K) + (r+q+0.5*v**2)*T)/(v*np.sqrt(T))
    d2 = (np.log(S/K) + (r-q+0.5*v**2)*T)/(v*np.sqrt(T))
    # d2 = d1 - v*np.sqrt(T)
    N1 = norm.cdf(d1)
    N2 = norm.cdf(d2)

    C = S*N1*np.exp(-q*T)-K*N2*np.exp(-r*T)
    return C

def bsPut(S, K, r, v, q, T):
    d1 = (np.log(S/K) + (r+q+0.5*v**2)*T)/(v*np.sqrt(T))
    d2 = (np.log(S/K) + (r-q+0.5*v**2)*T)/(v*np.sqrt(T))
    # d2 = d1 - v*np.sqrt(T)
    N1 = norm.cdf(-d1)
    N2 = norm.cdf(-d2)

    P = S*N2*np.exp(-r*T)-K*N1*np.exp(-q*T)
    return P

def Delta(S, K, r, v, q, T):
    d1 = (np.log(S/K) + (r-q+0.5*v**2)*T)/(v*np.sqrt(T))
    N1 = norm.cdf(d1)
    delta = np.exp(-q*T)*N1
    return delta

S = 78900.0
K = 78800.0
r = 0.0386
v = 0.0373
T = 45/365
N = 30
q = 0.0

PrcC = bsCall(S, K, r, v, q, T)
PrcP = bsPut(S, K, r, v, q, T)
Cdelta = Delta(S, K, r, v, q, T)
Pdelta = Cdelta -np.exp(-v*T)
print("Call Option Price :", PrcC)
print("Call delta :", Cdelta)
print("Put Option Price :", PrcP)
print("Put delta :", Pdelta)

<OUTPUT>
Call Option Price : 690.8274269911053
Call delta : 0.6796584229814875
Put Option Price : 281.1006568557859
Put delta : -0.3157535043915335
```

<Code 3>

```
S = 78900.0
K = 78800.0
r = 0.0386
v = 0.0373
T = 44/365
N = 29
q = 0.0

PrcC = bsCall(S, K, r, v, q, T)
PrcP = bsPut(S, K, r, v, q, T)
Cdelta = Delta(S, K, r, v, q, T)
Pdelta = Cdelta -np.exp(-v*T)
print("Call Option Price :", PrcC)
print("Call delta :", Cdelta)
print("Put Option Price :", PrcP)
print("Put delta :", Pdelta)

<OUTPUT>
Call Option Price : 681.0996814893806
Call delta : 0.678570393075141
Put Option Price : 279.88308118073473
Put delta : -0.3169432624131897
```

<Code 4>

```
def GeoAvg(array):
    multiple = 1
    a1 = array
    n = len(array)

    for i in a1:
        multiple = multiple * i

    avg = multiple ** (1/n)
    return avg

r = .0386
v = .0373
q = 0
T = 30/365
N = 30
dt = T/N

hedge_list = []

for i in range(9):
    price = StockInfo.iloc[:, i].values
    for k in range(3):
        price_2 = price[k:30+k]
        K = GeoAvg(price)
        cost = 0
        hedge = 0
        for j in range(30):
            time_rest = T - j*dt
            delta = GeometricAsianDelta(price_2[j], K, r, v, q, time_rest,
N)
            cost = cost + (delta - hedge) * S
            hedge = delta

        cost = cost + hedge * price[30]
        hedge_list.append(cost)

data_1 = pd.DataFrame(np.reshape(hedge_list, (9, 3)),
index=StockInfo.columns, columns=["0426 - 0610", "0427 - 0613", "0428 -
0614"]).T
```

<Code 5>

```
import plotly.express as px

px.histogram(data_1)
```

<Code 6>

```
import tensorflow as tf

my_input = []
premium = tf.keras.layers.Input(shape=(1, ), name='premium')
hedge_cost = tf.keras.layers.Input(shape=(1, ), name='hedge_cost')
price = tf.keras.layers.Input(shape=(1, ), name='price')
my_input = my_input + [premium] + [hedge_cost] + [price]

for j in range(8):
    delta = tf.keras.layers.Dense(32, activation='tanh')(price)
    delta = tf.keras.layers.Dense(1)(delta)

    new_price = tf.keras.layers.Input(shape=(1, ), name='S'+str(j+1))
    my_input = my_input + [new_price]

    price_inc = tf.keras.layers.Subtract(name="price_inc_"+str(j))([price,
new_price])
    cost = tf.keras.layers.Multiply(name="stock_"+str(j))([delta,
price_inc])
    hedge_cost = tf.keras.layers.Add(name='cost_'+str(j))([hedge_cost,
cost])
    price = new_price

payoff = tf.keras.layers.Lambda(lambda x : tf.math.maximum(x-K,0),
name='payoff')(price)
cum_cost = tf.keras.layers.Add(name="hedge_cost_plus_payoff")([hedge_cost,
payoff])
cum_cost = tf.keras.layers.Subtract(name="cum_cost-premium")([cum_cost,
premium])

model = tf.keras.Model(inputs=my_input, outputs=cum_cost)
```

<Code 7>

```
tf.keras.utils.plot_model(model)
```

<Code 8>

<Code 9>

```
# import plotly.express as px

hedge_data = pd.DataFrame(model.predict(x), columns=StockInfo.columns)
px.histogram(hedge_data)

premium = a[1] * np.ones([33, 1])
cost = np.zeros([33, 1])
SS = [StockInfo.iloc[:,i].values.reshape(33, 1) for i in range(9)]
x = [premium] + [cost] + [SS]
y = np.zeros([33, 1])

model.compile(loss='mse', optimizer='adam')
model.fit(x, y, batch_size=100, epochs=100, verbose=True,
validation_split=0.2)
```

```
MeanSquaredError = []

for i in range(9):
    line = hedge_data.iloc[:, i]
    d1 = 0
    for j in range(32):
        n1 = np.sqrt(line[j]**2)
        d1 += n1
    MeanSquaredError.append(d1)
```