Project Summary

Xiaoyu Jiang, Yuyang Lin and Ziming Gao

October 1, 2020

1 Rough Volatility Model and Simulation

1.1 Black-Scholes model

The model assume the underlying asset's price follows a general stochastic differential equation(SDE):

$$dS = \mu(S,t)dt + \sigma(S,t)dW(t) \tag{1.1}$$

The $\mu(S,t)dt$ represent the return rate of the stock. The $\sigma(S,t)dW(t)$ is a noise component. Then we could assume that $\mu(S,t) = \mu S$ and $\sigma(S,t) = \sigma S$. We could have

$$dS = \mu S dt + \sigma S dW(t) \tag{1.2}$$

Then we let $V(S,t)$ denote the value of an option. We could use the $Itô's$ Lemma, we have

$$dV(S,t) = (\frac{\partial V}{\partial t} + \mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2})dt + \sigma S\frac{\partial V}{\partial S}dW(t) \tag{1.3}$$

By the stochastic process, we could have

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{1.4}$$

where the r is the risk-free interest rate. This is the Black-Scholes partial differential equation. Therefore, we could have the Black-Scholes formula for

a European call option and put option.

$$V_C(S,t) = Se^{-q(T-t)}N(d_1) - Ke^{-r(T-t)}N(d_2) \quad (1.5)$$
$$V_P(S,t) = Ke^{-r(T-t)}N(-d_2) - Se^{-q(T-t)}N(-d_1) \quad (1.6)$$

where K is the strike price, q is the dividend-yield and

$$d_1 = \frac{ln(S/K) + (r - q + \frac{1}{2}\sigma^2)(T-t)}{\sigma\sqrt{T-t}} \quad (1.7)$$
$$d_2 = d_1 - \sigma\sqrt{T-t} \quad (1.8)$$

1.2 Delta Hedge portfolios

1.2.1 Delta

Delta is the rate of change of the option value with respect to changes in the underlying asset's price.

$$\Delta = \frac{\partial V}{\partial S} = iopt * N(iopt * d_1)exp(-qT) \quad (1.9)$$

where iopt is binary argument which takes value 1 for calls and -1 for puts.

1.2.2 Delta hedge portfolio

Delta-hedged portfolio is hedge against small changes in share price(called delta risk). The basic idea of a delta-hedged portfolio is to first create a zero-investment portfolio

$$\Pi = hS - c + B = 0 \quad (1.10)$$

which consist of h shares of long stock, one short call and borrowing B amount of cash. The objective of the delta portfolio is to offset the change in the stock value by the change in the option value.

1.3 Rough Volatility Model

1.3.1 Basic idea

From the Black-Scholes model, the basic assumption is the price of underlying asset satisfies $dS = \mu S dt + \sigma S dW(t)4$, where the volatility is constant. However, the historic volatilities of stocks shows that volatility could vary considerably in very short period of time. So it is reasonable to assume that volatility is changing in time. We will assume that the log-volatility, $\log \sigma_t$, will be given by a stochastic process $X = \log \sigma$ with dynamics

$$dX_t = \nu dY_t - \alpha(X_t - m)dt \tag{1.11}$$

with $m \in R, \nu, \alpha > 0$ and Y some stochastic process. The key point is to determine which process Y could be used to determine X. One reasonable idea is consider that Y is a Brownian motion. However, volatility process is rougher than the price process of the stock itself. Therefore, we will try to pick a process X that is rougher than a Brownian motion.

1.3.2 Rough Fractional Stochastic Volatility Model(RFSV)

One idea of Y is that we assume the Y is fractional Brownian Motion. Then we have the SDE for X:

$$dX_t = \nu dW_t^H - \alpha(X_t - m)dt \tag{1.12}$$

where the parameter H is assumed to be $H < \frac{1}{2}$ so that W_t^H is rougher than the Brownian motion. In [5], they found that the parameter H is approximately to 0.1, showing that volatility is rough.

1.4 Simulation rough volatility model

The rough Bergomi model [1] assumed that the volatility is satisfied a fractional Brownian motion. The asset price process and variance process is defined by following equations:

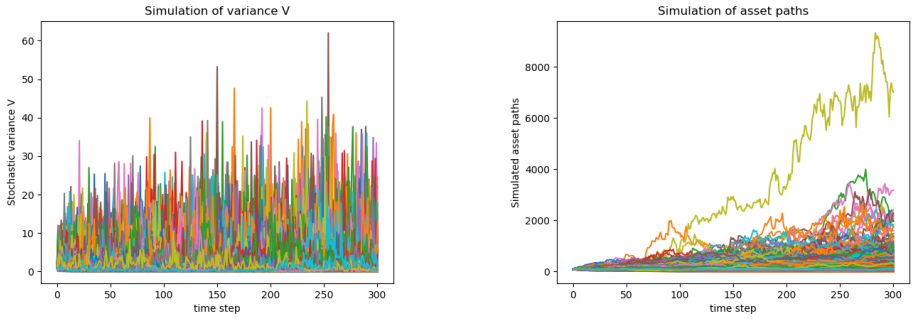
$$S_t = exp(\int_0^t \sqrt{V_u} dB_u - \frac{1}{2}\int_0^t V_u du) \tag{1.13}$$

$$B_u = \rho W_u^1 + \sqrt{1-\rho^2} W_u^2 \tag{1.14}$$

$$V_t = \xi exp(\eta Y_t^\alpha - \frac{\eta^2}{2} t^{2\alpha+1}) \tag{1.15}$$

$$Y_t^a = \sqrt{2a+1} \int_0^t (t-u)^a dW_u^1 \tag{1.16}$$

where W^1, W^2 are Brownian motion, $\rho \in [-1, 1]$ and $\eta > 0$.
We use the hybrid scheme[2] to simulate the Volterra process Y_t^α. Then construct the variance process V_t and construct the price process S_t finally. We obtain the simulation of price process with different paths. The result of variance process and price process are shown in figures. 1
Then the strike price K and asset price S_T is used to compute the payoff



(a) variance (b) price

Figure 1: Simulation of variance process and price process

then compute the mean value of payoff of different paths to obtain the call option. $C = E[max(S_T - K, 0)]$ [4]. The simulation of option price is shown in figure.2. We then simulate the delta of the option value. We consider to use the pathwise method to simulate the option greek. Based on the following equation [3].

$$\delta = \frac{dC}{dS} = e^{rT} I_{\{S_T > K\}} \frac{S_T}{S} \tag{1.17}$$

where $l_{\{\}}$ is the indicator.

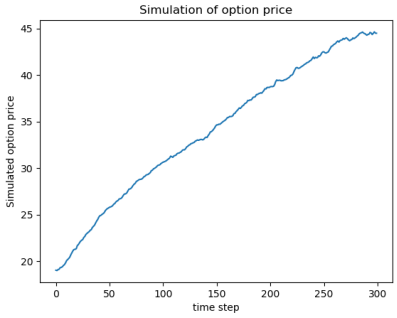


Figure 2: Simulation of option price

2 Deep Reinforcement Learning

2.1 Review

The theory of reinforcement learning originated from psychology and neuroscience that how animal or human learning in the environment[8]. The process of reinforcement learning is training how the agent map a situation to a proper action and maximize one reward function, which is different from traditional supervised learning and unsupervised learning. The agent takes raw inputs of data, returns an action to the environment and attain a reward from the environment. It aims to maximize the mean value of reward received from environment. In the deep reinforcement learning, the artificial agent is a neural network in the deep learning model. The artificial agent has already performed well in many different areas such as different games.

2.2 Deep Q-Network(DQN)

In [7], a deep Q-network design for combining the Q-learning and the deep neural networks. The artificial agent using artificial neural network could learn concepts directly from the raw data. The detail structure of the deep neural networks could be varied for different problem. With the process of the interaction of environment and agent, the deep neural network tries to attain the optimal solution of the action-value function as follow.

$$Q^*(s,a) = \mathbf{max}_\pi \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi] \tag{2.1}$$

where the π is a behavior policy. And the whole algorithm is as follow.

Algorithm 1 deep Q-learning algorithm

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
for $Episode = 1, 2, .., M$ **do**
    Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    for $t = 1, .., T$ **do**
        if $random < \epsilon$ **then**
            $a_t = random \quad action$
        else
            $a_t = argmax_a Q(\phi(s_t), a; \theta)$
        end if
        Take action a_t and attain reward r_t and image x_{t+1}
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and $\phi_{t+1} = \phi(s_{t+1})$
        Store the transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
        Sample random small batch of transitions from D
        Set $y_j = r_j + \gamma max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-})$
        Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to parameter θ
        Reset $\hat{Q} = Q$ for every C steps
    end for
end for

3 Rough Delta Hedging using DQN

3.1 Environment

The model of DQN is constructed for hedging in the stock market based on current condition of market. So the state provided by environment should only include the information that agent could observe at that time. So we set the input as a set contains six elements: expiry time $T-t$, volatility σ_t, asset price S_t, number of stock N_t, option price C_t and one Greek letter Δ_t. And since we assume agent hedges 100 call options so $\Delta_t = 100 * \frac{\partial C_t}{\partial S_t}$
In this model, the action a_t is the number of shares at time $t+1$. Since the action in DQN is discrete and finite, we set the range of action is from 0 to 100.
Assume we do not consider the transaction cost, The delta-hedge portfolio could be

$$\Pi_t = N_t S_t + B_t e^{r\Delta t} - 100 C_t \tag{3.1}$$

And if it is a self-financing condition and no transaction cost, then we have

$$B_{t+1} = B_t e^{r\Delta t} - (N_{t+1} - N_t) S_t \tag{3.2}$$

Since the bank account of cash could be uniquely determined by Equation.3.2 and stock price list, so the state do not need to contain B_t.
And we also consider the influence of the variance of the portfolio. So we have the reward function as follow [6].

$$r_t = \begin{cases} \gamma^{-t}[\Delta r_t - \frac{\kappa}{2}\Delta v_t], & (0 \le t < T) \\ \gamma^{-T}[(100C_0 - N_0 S_0)e^{rT} - 100C_0], & (t = T) \end{cases} \tag{3.3}$$

where κ balances the reward for return and risk and

$$\Delta r_t = (N_{t+1}S_{t+1} - N_t S_t) - (N_{t+1} - N_t) S_t e^{r(T-t)} - 100(C_{t+1} - C_t) \tag{3.4}$$

$$\Delta v_t = [(N_{t+1} - D_t)\sigma S_t]^2 \Delta t \tag{3.5}$$

3.2 Agent

As introduced previously, we use deep neural network as the artificial agent and code the agent using the TensorFlow2.0. The detail structure of the

Layer	output shape	activate function
dense	16	relu
dense	128	relu
dense	101	linear

Table 1: agent

neural network are shown as follow. We input a small batch with 32 states each time for training. The states contains six elements. The shape of output is 101-dimension vector, which corresponding to 101 kinds of actions. We choose the action with highest output value.

References

[1] Christian Bayer, Peter Friz, and Jim Gatheral. Pricing under rough volatility. *Quantitative Finance*, 16(6):887–904, 2016.

[2] Mikkel Bennedsen, Asger Lunde, and Mikko S Pakkanen. Hybrid scheme for brownian semistationary processes. *Finance and Stochastics*, 21(4):931–965, 2017.

[3] Mark Broadie and Paul Glasserman. Estimating security price derivatives using simulation. *Management science*, 42(2):269–285, 1996.

[4] Amir Dar and N. Anuradha. Option pricing using monte carlo simulation. *British Journal of Economics, Finance and Management Sciences*, 13:53–81, 03 2017.

[5] Jim Gatheral, Thibault Jaisson, and Mathieu Rosenbaum. Volatility is rough. *Quantitative Finance*, 18(6):933–949, 2018.

[6] Igor Halperin. Qlbs: Q-learner in the black-scholes (-merton) worlds. *The Journal of Derivatives*, 2020.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[8] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.