Optimal Hedged Monte Carlo

Model Inventors: Marc Potters, Jean-Philippe Bouchaud, Dragan Sestovic

Author: Jerry Xia

Date: 2018/06/19

Note: The advanced Marckdown features such as math expression may not be compatible in GitHub, please see README.pdf instead if you want more details

1 Introduction

This is a Python Notebook about optimal hedged Monte Carlo (OHMC) in option pricing. We invoke this method to price European options and make comparison with regular Monte Carlo and Black-Scholes formula.

1.1 Facts

- The option price is not simply the average value of the discounted future pay-off over the objective (or historical) probability distribution
- The requirement of absence of arbitrage opportunities is equivalent to the existence of "risk-neutral measure", such that the price is indeed its average discounted future pay-off.
- Risk in option trading cannot be eliminated

1.2 Objective

- It would be satisfactory to have an option theory where the objective stochastic process of the underlying is used to calculate the option price, the hedge strategy and the *residual risk*.

1.3 Advantages

- It is a versatile methods to price complicated path-dependent options.
- Considerable variance reduction scheme for Monte Carlo
- It provide not only a numerical estimate of the option price, but also of the optimal hedge strategy and of the residual risk.
- This method does not rely on the notion of risk-neutral measure, and can be used to any model of the true dynamics of the underlying

2 Underlying Dynamics

Black-Scholes Model

$$dS = rSdt + \sigma SdW_t$$

$$logS_{t+1} = logS_t + (r - \frac{\sigma^2}{2})\Delta t + \sigma\sqrt{\Delta t}\epsilon$$

where

$$\epsilon \sim N(0, 1)$$

3 Methodology

3.1 Simbol Definition

Option price always requires to work backward. That is because the option price is known exactly at the maturity. As with other schemes, we determine the option price step by step from the maturity $t = K\tau = T$ to the present time $t = 0$. The unit of time being τ, for example, one day. We simulate N trajectories. In trajectory i, the price of the underlying asset at time $k\tau$ is denoted as $S_k^{(i)}$. The price of the derivative at time $k\tau$ is denoted as C_k, and the hedge function is H_k. We define an optimal hedged portfolio as

$$W_k^{(i)} = C_k(S_k^{(i)}) + H_k(S_k^{(i)})S_k^{(i)}$$

The one-step change of our portfolio is

$$\Delta W_k^{(i)} = df(k, k+1)C_{k+1}(S_{k+1}^{(i)}) - C_k(S_k^{(i)}) + H_k(S_k^{(i)})(df(k, k+1)S_{k+1}^{(i)} - S_k^{(i)})$$

Where $df(k, k+1)$ is the discounted factor from time $k\tau$ to $(k+1)\tau$

3.2 Objective

The optimal hedged algorithm can be interpreted as the following optimal problem

$$\begin{array}{ll} \text{minimize} & Var[\Delta W_k] \\ \text{subject to} & E[\Delta W_k] = 0 \end{array}$$

It means we should try to minimize the realized volatility of hedged portfolio while maintaining the expected value of portfolio unchanged.

3.3 Basis Functions

The original optimization is very difficult to solve. Thus we assume a set of basis function and solved it in such subspace. We use N_C and N_H to denote the number of basis functions for price and hedge.

$$C_k(\cdot) = \sum_{i=0}^{N_C} a_{k,i} A_i(\cdot)$$

$$H_k(\cdot) = \sum_{i=0}^{N_H} b_{k,i} B_i(\cdot)$$

The basis functions A_i and B_i are priori determined and need not to be identical. The coefficients a_i and b_i can be calibrated by solving the optimal problem.

3.4 Numerical Solution

$$\begin{aligned} &\text{minimize} \qquad && \frac{1}{N}\sum_{i=1}^{N} \Delta W_k^{(i)2} \\ &\text{subject to} \qquad && \frac{1}{N}\sum_{i=1}^{N} \Delta W_k^{(i)} = 0 \end{aligned}$$

Denote the discounted forward underlying price change at time $k\tau$ as

$$\Delta S_k = df(k, k+1)S_{k+1} - S_k$$

Define

$$Q_k = \begin{bmatrix} -A_{k,1}(S_k^{(1)}) & \cdots & -A_{k,N_C}(S_k^{(1)}) & B_{k,1}(S_k^{(1)})\Delta S_k^{(1)} & \cdots & B_{k,N_H}(S_k^{(1)})\Delta S_k^{(1)} \\ -A_{k,1}(S_k^{(2)}) & \cdots & -A_{k,N_C}(S_k^{(2)}) & B_{k,1}(S_k^{(2)})\Delta S_k^{(2)} & \cdots & B_{k,N_H}(S_k^{(1)})\Delta S_k^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -A_{k,1}(S_k^{(N)}) & \cdots & -A_{k,N_C}(S_k^{(N)}) & B_{k,1}(S_k^{(N)})\Delta S_k^{(N)} & \cdots & B_{k,N_H}(S_k^{(N)})\Delta S_k^{(N)} \end{bmatrix}$$

$$c_k = (a_{k,1}, \cdots a_{k,N_C}, b_{k,1}, \cdots, b_{k,N_H})^T$$

$$v_k = df(k, k+1)C_{k+1}(S_{k+1})$$

As for v_k, note that we know the exact value at maturity, which means there is no need to approximate price in terms of basis functions, that is

$$v_k = \begin{cases} df(N-1,N)\, payoff(S_N), & k = N-1 \\ df(k,k+1)\ \sum_{i=1}^{N_C} a_{k+1,i} A_i(S_{k+1}), & 0 < k < N-1 \\ df(0,1)\, C_1(S_1), & k = 0 \end{cases}$$

Then, the optimization problem can be expressed as

$$\begin{aligned} \arg\min_{c_k} \quad & (v_k + Q_k c_k)^T (v_k + Q_k c_k) \\ \text{subject to} \quad & 1^T_{[N\times 1]}(v_k + Q_k c_k) = 0 \end{aligned}$$

In step k, since we already know the information (v_k) in step k+1. By canceling the constant term, the optimal problem can be simplified as the following

$$\begin{aligned} \arg\min_{c_k} \quad & 2v_k^T Q_k c_k + c_k^T Q_k^T Q_k c_k \\ \text{subject to} \quad & 1^T_{[N\times 1]} v_k + 1^T_{[N\times 1]} Q_k c_k = 0 \end{aligned}$$