Option Pricing with Time-Stepped FBSDE and Deep Learning

Jeremy Lu, Chengyang Gu

NYU Courant

June 2, 2022

Introduction

- "Introduction to Solving Quant Finance Problems with Time-Stepped FBSDE and Deep Learning" (Hientzsch 2019)

The Problem

- Asset pricing is one of the central problems in financial economics.
- A typical option pricing problem may be expressed as finding

$$Y_t = E[Y(X_T)|X_t = x] \tag{1}$$

  given the dynamics of the underlying asset X_t.
- Generally speaking, the value of the contract may depend on a number of different variables (called risk factors) in addition to the underlying asset.
- We formulate this as a forward-backward stochastic differential equation (FBSDE) problem.

Risk Neutral Pricing

- In risk-neutral pricing, we find

$$Y_t = E_{\mathbb{Q}}\left[e^{-\int_t^T r_s ds} Y_T | \mathcal{F}_t\right] \quad (2)$$

given the stock price following the dynamics

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t \quad (3)$$

- The Black-Scholes model is a specialization of risk-neutral pricing, with a constant risk-free rate, drift and variance of the stock price.
- The traditional approach is to first write out the PDE about the value function, and then solve the PDE with analytical or numerical methods.

Dynamic Replication

- Risk neutrality can be implemented by tracking the cash flow of a portfolio.
- For example, in Black-Scholes, we have

$$dY_t = r_t Y_t dt + \Delta(t, X_t)\sigma_t dW_t \tag{4}$$

- If we can know the position of stock $\Delta(t, X_t)$, this equivalently solves the option pricing problem.
- This is reminiscent of a stochastic control problem, where $\Delta(t, X_t)$ is the dynamic programming principal.
- The Hamilton-Jacobi-Bellman (HJB) equation gives a necessary and sufficient condition for $\Delta(t, X_t)$, but it's usually not easy to solve.

Digression: Reinforcement Learning

- In the context of reinforcement learning, an agent interacts with the environment and takes sequential actions (called policy) to optimize a goal function.
- The policy in the example of European call options is $\Delta(t, X_t)$.
- The goal function is the tracking error, which we hope to minimize.
- Many successful algorithms have been posed in the realm of reinforcement learning, but our approach is fundamentally different.

Forward and Backward SDEs

- The forward stochastic differential equation (FSDE) is an Ito process

$$dX_t = \mu(t, X_t)dt + \sigma(t, X_t)dW_t \tag{5}$$

- The backward SDE (BSDE) describes contract values that take future values of risk factors into account

$$dY_t = f_Z(t, X_t, Y_t, Z_t)dt + Z_t dW_t \tag{6}$$

  where Z_t is the policy.
- In the example of European call options, it can be simply

$$dY_t = rY_t dt + \Delta(t, X_t)\sigma dW_t \tag{7}$$

Forward Approach

- Given the initial stock price X_0, the objective is to replicate the final payoff. In other words, we minimize

$$loss(X_T, Y_T) = E[(Y_T - Y(X_T))^2] \quad (8)$$

- For European call options, we have

$$Y(X_T) = \max\{X_T - K, 0\} \quad (9)$$

- The loss function can be easily generalized to price American options, exotic options (e.g. barrier options) and etc.

The Monte Carlo Method

- Since the loss function involves expectation, we can apply Monte Carlo simulation to estimate it.
- Suppose we know the policy $\Delta(t, X_t)$. With the forward and backward SDEs, we can easily construct the whole sample paths of X_t and Y_t.

Utilizing Neural Networks

- So far, we haven't touched on the estimation of the policy $\Delta(t, X_t)$.
- To use Monte Carlo in the way suggested above, an iterative algorithm can be handy.
- We use a neural network to estimate $\Delta(t, X_t)$.
- The value of the contract Y_0 (or as a function $Y_0(X_0)$) is also trainable and can be jointly optimized with any stochastic gradient descent algorithm (e.g. the Adam optimizer).

Computation Graph

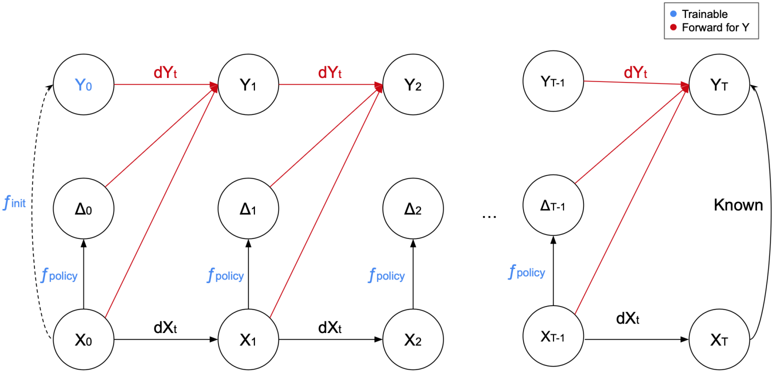

Figure 1: Computation Graph of Forward Approach

Backward Approach

- Given the full sequence of stock prices, the goal is to find a least variant approximation of Y_0.
- If X_0 is fixed, this simply reduces to minimizing the variance of Y_0.
- If X_0 is not fixed, the posterior mean of Y_0 is replaced with another function $Y_0(X_0)$, which can be jointly optimized with the policy $\Delta(t, X_t)$.

Computation Graph

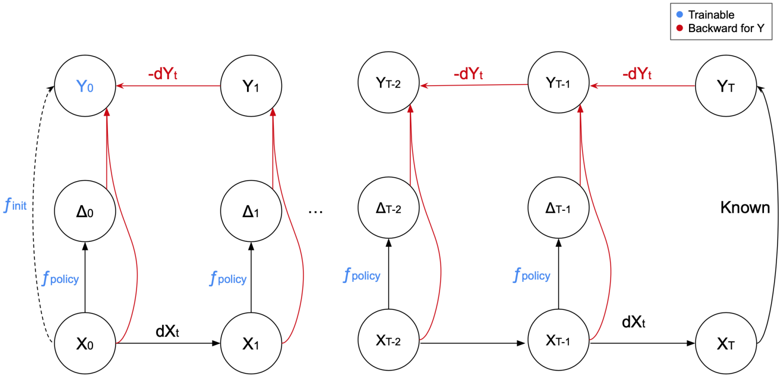

Figure 2: Computation Graph of Backward Approach Using Backward Policy

Computation Graph

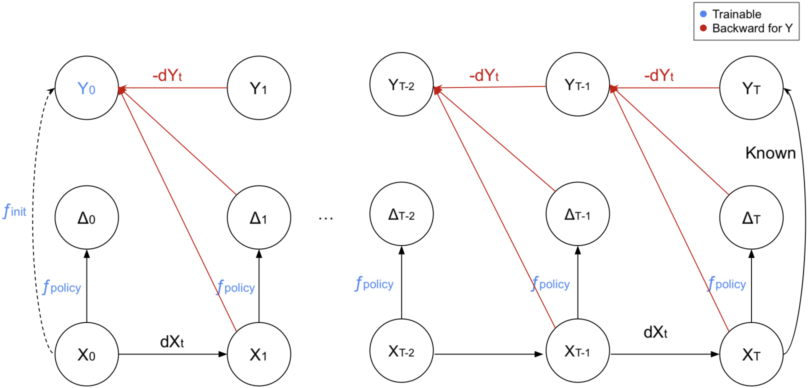

Figure 3: Computation Graph of Backward Approach Using Forward Policy

Pricing European Options

- A combination of a long call at 120 and two short calls at 150.
- Constant drift (=risk-free rate) 6% and volatility 20%. Maturity is 6 months, discretized to 50 steps.
- A 4-layer fully-connected neural network for policy with layer sizes 2,11,11,1. Batch size is 512.
- The initial stock price X_0 can be either fixed or not. In the latter case, simply use another neural network to represent the initial portfolio value as a function of X_0, and optimize the two networks jointly.
- The solution $Y_0(X_0)$ exists in this setup and can be computed with the Black-Scholes model.

Forward Approach

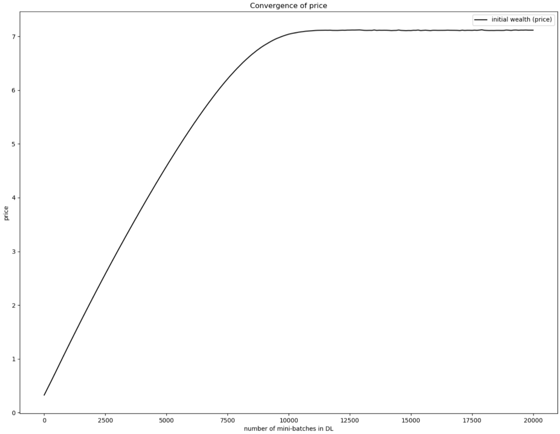

Figure 4: Convergence of Price

Forward Approach

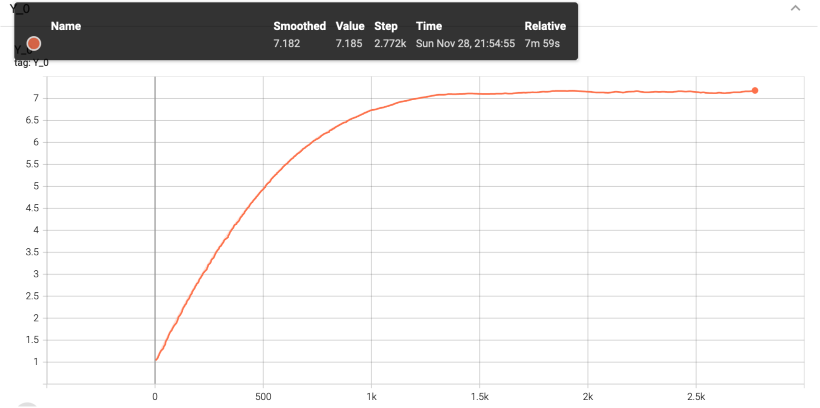

Figure 5: Convergence of Price - Another Experiment (Truth≈7.14)

Forward Approach

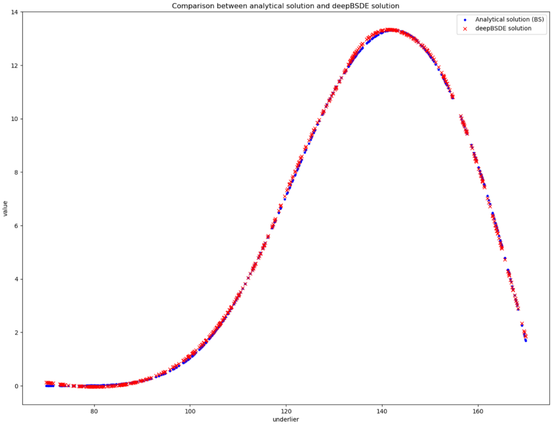

Figure 6: Convergence of Price Function

Backward Approach

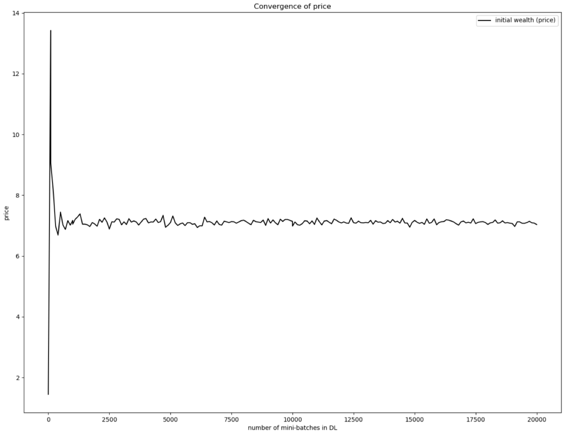

Figure 7: Convergence of Price

Backward Approach

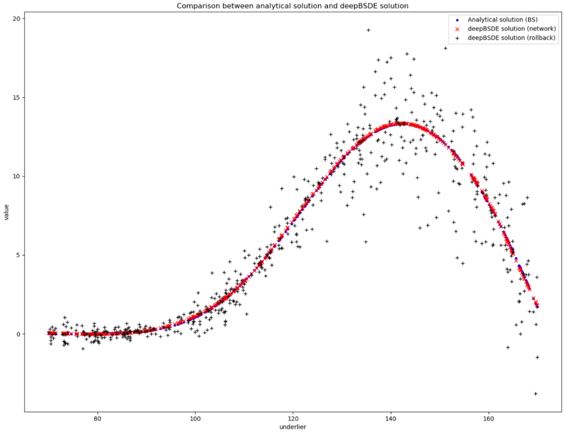

Figure 8: Convergence of Price Function

Advantages

- Easy to extend to more risk factors and other dynamics besides geometric Brownian motion. Only a few lines of code need adjustment.
- Faster and more accurate compared with pure Monte Carlo simulation.
- Can be explained.

Disadvantage

- More computationally intense and less accurate than traditional PDE approaches, if the problem can be solved using the latter.

Q & A