Project 3: Deep Hedging

(First discussion: Nov 6; Last questions: Nov 20; Deadline: Nov 27)

Responsible: Gabriele Visentin

The goal of this project is to implement the deep hedging model introduced in [Buehler et al., 2019]. You must implement your model from scratch, either in PyTorch (see `demo.ipynb` for a quick demo) or in TensorFlow. You are not allowed to use third-party repositories with ready-made implementations of deep hedging. Submit your solution using the template provided in `template3.ipynb`

1. Consider the Black–Scholes model in which the risky asset S follows a risk-neutral dynamics given by:

$$dS_t = rS_t dt + \sigma S_t dW_t, \quad S_0 = s_0 \in \mathbb{R}_+, \tag{1}$$

where W is a Brownian motion under the unique risk-neutral measure $\mathbb{Q}$, σ is the annualized volatility, and where we assume that the risk-free interest rate r is zero[1].

Given an option with payoff $g(S_T)$ and maturity T, the hedging problem consists in finding a self-financing trading strategy H with initial value equal to the risk-neutral price of the option and such that its value at maturity is exactly equal to the option payoff.

In a complete market model, such as the Black–Scholes model, every option admits a hedging strategy, which can therefore be represented as the solution of the following optimization problem:

$$\inf_{H \in \mathcal{H}} \mathbb{E}\left[\left(g(S_T) - p - \int_0^T H_u dS_u\right)^2\right],$$

where $\mathcal{H}$ is the set of all predictable processes and p is the risk-neutral option price.

We can solve this problem numerically on a uniform time grid $0 = t_0 < t_1 < \ldots < t_N = T$ by approximating the Itô integral with the discrete stochastic integral $\sum_{j=0}^{N-1} H_{t_j} \cdot \left(S_{t_{j+1}} - S_{t_j}\right)$, where $H_{t_0}, \ldots, H_{t_{N-1}}$ are N neural networks jointly trained by minimizing the following empirical loss

$$\frac{1}{m}\sum_{i=1}^{m}\left(g\left(s_T^{(i)}\right) - p - \sum_{j=0}^{N-1} H_{t_j} \cdot \left(s_{t_{j+1}}^{(i)} - s_{t_j}^{(i)}\right)\right)^2 \tag{2}$$

on a training set $D = \left((s_{t_0}^{(i)}, s_{t_1}^{(i)}, \ldots, s_{t_N}^{(i)}), 0 \le i \le m\right)$ of m simulated paths of S.

Implement and test the model following the steps below:

(a) Use Itô's formula to check that $S_t = s_0 \exp\left(\sigma W_t - \frac{1}{2}\sigma^2 t\right)$ is a solution for the SDE (1).

[1] If the risk-free interest rate r is non-zero, one can reduce to the zero interest rate case by working with discounted prices.

(b) Simulate a training set of 10^5 paths and a test set of 10^4 paths for the asset S with parameters $N = 30$, $S_{t_0} = s_0 = 1$, $T = 1$ month $= 30/365$, $\sigma = 0.5$.

The process S can be simulated exactly on a finite grid by setting

$$S_{t_{j+1}} = S_{t_j} \exp\left(-\frac{\sigma^2}{2}\frac{T}{N} + \sigma\sqrt{\frac{T}{N}} Z_{j+1} \right),$$

where $Z_1, \ldots, Z_N$ are N iid standard Gaussian random variables.

(c) Implement the model by defining each H_{t_j} as a neural network with input $\log(S_{t_j})$. Start by choosing a simple architecture (e.g. one hidden layer with 32 neurons).

(d) Let us assume we sell a European call option, i.e. an option with payoff $g(S_T) := (S_T - K)^+$, with strike $K = 1$ and maturity $T = 1$ month $= 30/365$. Train the deep hedging model for this option by minimizing the loss (2) on the training set.

To compute the risk neutral price p, recall that in the Black–Scholes model, the value of a European call option at time t, denoted by $C(S_t, t)$, can be computed explicitly and is a function of the value of the risky asset S_t and of time t:

$$C(S_t, t) = \Phi(d_+)S_t - \Phi(d_-)Ke^{r(T-t)}, \tag{3}$$

where

$$d_+ = \frac{1}{\sigma\sqrt{T-t}}\left(\log\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t) \right),$$

Φ is the standard Gaussian cumulative distribution function, and $d_- = d_+ - \sigma\sqrt{T-t}$.

You can compute the risk-neutral price $p := C(S_0, 0)$ using Equantion (3).

(e) Evaluate the hedging portfolio losses at maturity, i.e. $g(S_T) - p - \sum_{j=0}^{N-1} H_{t_j} \cdot \left(S_{t_{j+1}} - S_{t_j}\right)$, on the test set. Plot their histogram and print their empirical mean and standard deviation.

2. In the Black-Scholes model, the hedging problem in continuous time admits an analytical solution given by

$$H_t^{\mathrm{BS}}(s) = \frac{\partial C(s,t)}{\partial s}.$$

(a) Derive a closed-form formula for $H_t^{\mathrm{BS}}(s)$ by computing the partial derivative above.

(b) Evaluate the hedging portfolio losses on the test set when using the analytical hedging strategy H^{BS} to rebalance the hedging portfolio at the trading dates $t_0, t_1, \ldots, t_{N-1}$. Plot their histogram and print their empirical mean and standard deviation.

(c) Compare the histograms of hedging portfolio losses computed in Ex.1(e) and Ex.2(b).

(Hint: your neural network model should perform at least as well as the analytical model. Experiment with different architectures, activation functions, batch sizes, and learning rates.)

(d) For each $j \in \{0, 10, 20, 29\}$, create a plot in which you draw the neural network function $s \mapsto H_{t_j}(s)$ and the analytical solution $s \mapsto H_{t_j}^{\mathrm{BS}}(s)$ for $s \in [0.5, 1.5]$. For what times t_j are the two functions most similar? Why?

3. Instead of training N separate neural networks as in Exercise 2, one can compute the hedging strategy using a single neural network by adding time as an additional feature.

   (a) Implement a new deep hedging model by setting $H_{t_j} = F_\theta(\sqrt{T - t_j}, \log(S_{t_j}))$ for all timesteps j, where F_θ is a feedforward neural network. Start by choosing a simple architecture (e.g. two hidden layers with 32 neurons each).

   (b) Train the deep hedging model for our call option by minimizing the loss (2) on the training set.

   (c) Evaluate the hedging portfolio losses at maturity, i.e. $g\left(S_T\right) - p - \sum_{j=0}^{N-1} H_{t_j} \cdot \left(S_{t_{j+1}} - S_{t_j}\right)$, on the test set. Plot their histogram and print their empirical mean and standard deviation.

   (d) Compare the two deep hedging models in Ex.1 and Ex.3 in terms of their runtime, performance (e.g. hedging portfolio losses distribution) and overall number of parameters.

References

[Buehler et al., 2019] Buehler, H., Gonon, L., Teichmann, J., and Wood, B. (2019). Deep hedging. *Quantitative Finance*, 19(8):1271–1291.