230T-2 Project

Srajan Garg, Eddie Huang, Apeksha Jain, Dmitry Silantyev, Rachit Tripathi

October 2021

1 Introduction

In this report, we will attempt to replicate a paper regarding the dynamic hedging strategies using Reinforcement Learning methods. In specific, we will create and measure the effectiveness of RL models under varying input parameters and compare their performance to traditional Delta Hedging strategies. As suggested by the paper, we will see that the RL hedging strategy performs better than Delta hedging under high trading cost and lower risk aversion environments.

2 Data

In our analysis, we simulated stock prices under Geometric Brownian Motion to allow the aforementioned hedging strategies to build upon. We assume the contract which wil be hedged to be an at-the-money European Call Option. The initialized simulation details are:

- sigma = 0.01 (volatility)
- S0 = 100 (initial price)
- T = 10 (days to maturity) = 5 (periods per day)
- kappa = 0.1 (risk aversion)

3 Hedging

In many theoretical hedging practices, trading frictions are assumed to be negligible and continuous trading is possible. However, in practice, trading frictions are often material and only discrete trading frameworks exist. In order to balance an optimal hedging strategy that minimizes variance of profits with such trading frictions under a discrete framework, we set the reward function to be:

$$max(E[w_T] - k/2 * V[w_T])$$

where the final wealth wT is the sum of individual wealth increments delta*wT:

$$w_T = w_0 + \Sigma_t^T \delta w_t$$

With this optimization function, we allow the RL model to simultaneously maximize profits while minimizing variance in profits, subject to the agent's risk aversion.

In the random walk case, this leads to solving:

$$minpermissiblestrategies \Sigma_t^T (E[-\delta w_t] + k/2 * V[\delta w_t])$$

where

$$-\delta w_t = c_t$$

and ct is the total trading cost paid in period t (including commissions, bid-offer spread, market impact, and other sources of slippage.

Trading costs on trade size of n shares are defined as:

$$cost(n) = multiplier * TickSize * (abs(n) + 0.01n^2)$$

The quadratic term within trading cost is to model increasing market impact as trade size increases.

Finally, we choose the reward in each period to be:

$$R_t = \delta w_t - \kappa/2(\delta w_t)^2$$

4 Implementation

Our RL agent utilized a Neural Network that took previous output, simulated price, and time to maturity as input parameters while setting the negative reward as a loss function.

We trained a fully connected layer with 5 layers and 100 neurons. We utilized ReLU activations and adam optimization and adopted a learning rate schedule to decay by 0.1 after 200 episodes.

Expanding on the results of the paper by Kolm and Ritter, we trained the RL for 3 values of the risk aversion coefficient - 10,000, 100,000, and 1,000,000. We also incorporated 3 values of trading cost factors from 0.1, 1.0, and 10.0.

We run the strategies on 1000 simulations.

5 Results

First, we reference the results of the author which show that the RL hedging strategy exhibits lower total cost without a significant difference in volatility of total PL.

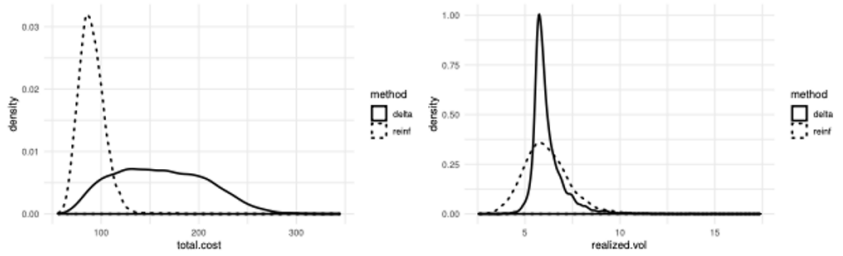


Figure 3: Kernel density estimates for total cost (left panel) and volatility of total P&L (right panel) from $N = 10,000$ out-of-sample simulations. The "reinf" policy achieves much lower cost (t-statistic $= -143.22$) with no significant difference in volatility of total P&L.

Additionally, when accounted for trading costs, the RL hedging strategy exhibits a smaller amount of PL loss compared to the Delta Hedging benchmark:

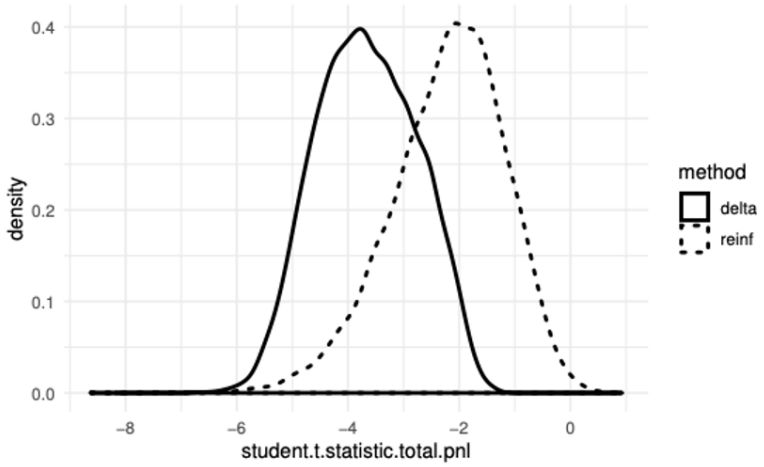
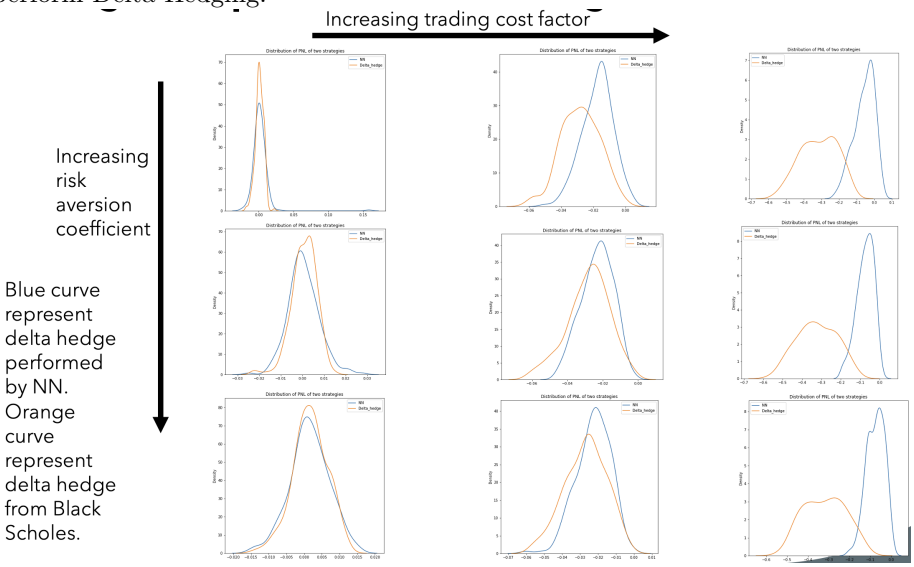


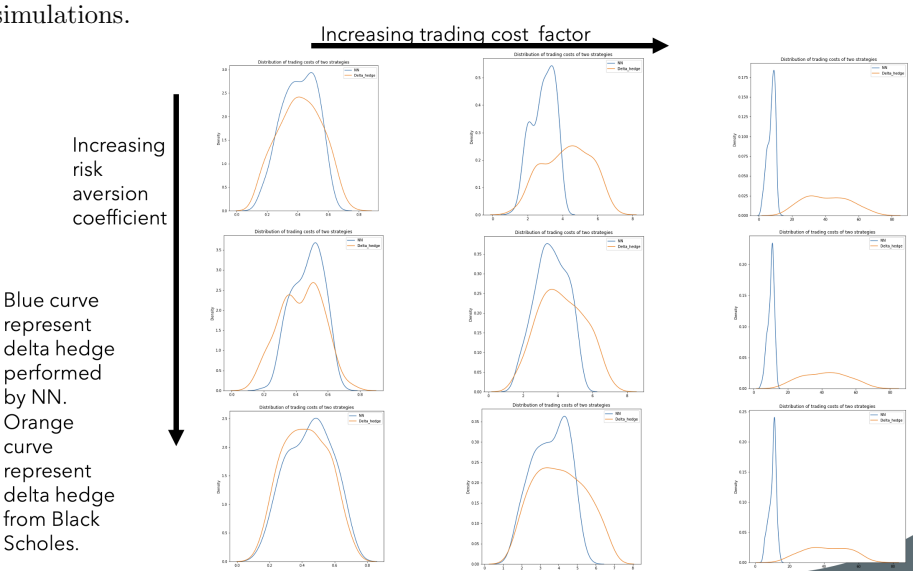
Figure 4: Kernel density estimates of the t-statistic of total P&L for each of our out-of-sample simulation runs, and for both policies represented above ("delta" and "reinf"). The "reinf" method is seen to outperform in the sense that the t-statistic is much more often close to zero and insignificant.

Our results from our Implementation step generally follow with the results

from Kolm and Ritter. On average, the NN hedging strategy exhibited higher PL than the traditional delta hedging strategy. At the lowest trading cost factor, the differences between PL histograms between the two hedging strategies was negligible but as we increased the trading cost factor, we saw NN begin to outperform Delta Hedging:

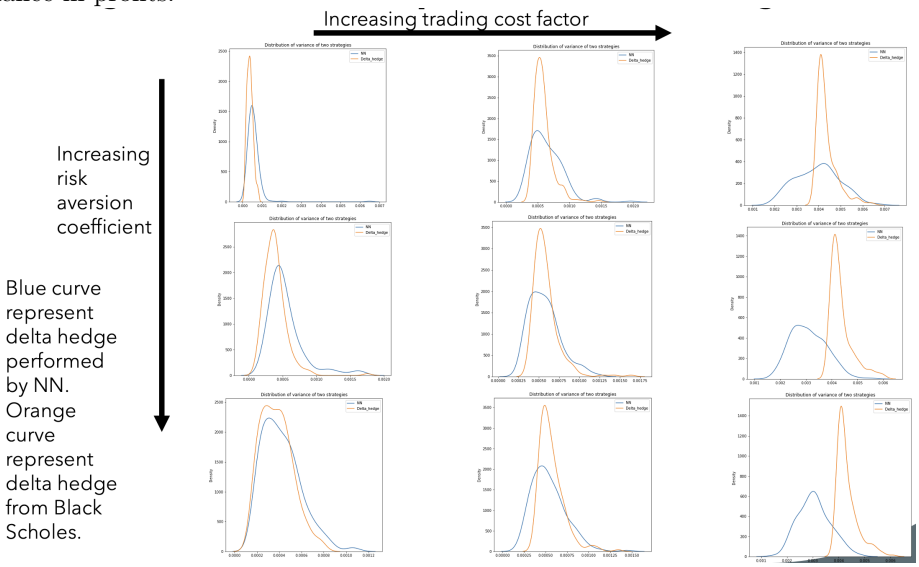


In terms of total trading costs, NN exhibited a smaller and more concentrated distribution, on average. While the effect was more muted for low trading cost factors, as we increased this parameters, we begain to see NN outperform Delta Hedging via keep trading costs low and without too much variation across all simulations.



Finally, on variance of profits, we see slightly mixed results. For low trading

cost factors, Delta Hedging has marginally lower variance. As trading cost factor increases, NN begins to outperform once again by exhibiting materially lower variance in profits.



In each of these graphs, we see that the risk aversion coefficient does not have significant impact in the performance of the RL strategy. The RL strategy outperforms in illiquid markets and even in liquid markets, the RL strategy behaves very similar to the Delta hedge strategy. This performance is with the disadvantage that RL does not know about the derivative payoff and trading cost structure and could be easily applied for complex derivative securities due to this.

6 Takeaways

We see that in the RL agent matches or outperforms Delta Hedging in nearly all scenarios when varying risk aversion parameter and trading costs. Illiquid markets tend to be where the RL hedging strategy outperforms, the lack of dependence on knowing security payoff structure allows this strategy to be transferred to many securities more complicated than vanilla Europen options.

The assumptions going into this strategy are a limiting factor, however, as geometric brownian motion may not be viable in the observed world and tail risk is more prevalent.

We would like to expand on the assumptions within the simulations in future research to see if our results still hold.

You can find our code at: (insert github repo here)