Special Topics in Financial Engineering: Deep Hedging

Aadam Wiggers
5855063
A.A.G.Wiggers@student.tudelft.nl

Albin Jaldevik
5839408
R.A.Jaldevik@student.tudelft.nl

June 16, 2023

The purpose of this paper is to outline the main ideas of the Deep Hedging paper written by Buehler et al. As a result, we omit some proofs for less central theorems (i.e. universal approximation by neural networks). For details on these, please refer to the original paper.

Introduction

The paper "Deep Hedging" by Buehler et al. [2] constructs a framework in which one can price and manage risk contingent claims in incomplete markets with frictions. The authors model the trading decisions for hedging strategies with neural networks, containing varying sets of features. The motivation behind the paper stems from the fact that there is a lack of alternatives to complete market pricing. The complete market assumption breaks in reality, as in real markets, trading is subject to transaction costs, market impact, and liquidity constraints, among others. Due to the lack of efficient alternatives, complete market pricing is still widely used today, which this paper chose to break away from. Furthermore, the framework that the authors present is extremely flexible. One can add as many constraints, transaction costs, etc. as you would like, and requires no equivalent martingale measure (EMM) model.

To build up to the method presented, the idea is built on the hedging of a derivative portfolio under some convex risk measure (defined later on). With the universal approximation properties of Neural Networks (NN), the authors outline the theoretical arguments as to why the parametrization of hedging strategies with NN, in theory, works.

Discrete Time Markets with Frictions

The following assumptions are the ones we work with in this setting. As we can see below, the assumptions are relatively general and allow for a lot of flexibility.

- **Finite** time horizon T (*also maximum maturity of all instruments*).
- **Trading dates** $0 = t_0 < t_1 < \cdots < t_n = T$.
- **Market information** at t_k denoted by $I_k \in \mathbb{R}^r$. Process $I = (I_k)_{k=0,\dots,n}$ generates filtration $\mathcal{F} = (\mathcal{F}_k)_{k=0,\dots,n}$.
- d **hedging instruments. Mid-prices** $\mathbb{R}^d$-valued $\mathcal{F}$-adapted stochastic process $S = (S_k)_{k=0,\dots,n}$.

- **Portfolio of derivatives** Z is $\mathcal{F}_T$ measurable random variable.
- To **hedge** Z at T, one may trade in S using $\mathbb{R}^d$-valued $\mathcal{F}$-adapted stochastic process $\delta = (\delta_k)_{k=0,\dots,n-1}$.
- **Unconstrained trading strategies** denoted by $\mathcal{H}^u$.
- **Restricted trading strategies** by $\mathcal{H} := (H \circ \mathcal{H}^u) \subset \mathcal{H}^u$.
- **Trading is self financed**: Might need injection of additional cash p_0 into portfolio. Then, trading strategy in hedging outputs the following PnL: $(\delta \cdot S)_T := \sum_{k=0}^{n-1} \delta_k(S_{k+1} - S_k)$.
- **Transaction costs**: $C_T(\delta) := \sum_{k=0}^{n} c_k(\delta_k - \delta_{k-1})$, where c_k are costs.
- **Terminal PnL (w/ costs)**: $PL_T(Z, p_0, \delta) := -Z + p_0 + (\delta \cdot Z)_T - C_T(\delta)$.

Note that the hedging instruments S do not require an EMM under which S is a martingale. Thus, we can have that the hedging instruments are not simply primary assets, but also secondary assets, even those which have liquidity constraints. These constraints can be implemented as follows.

Example 1 (Constrained strategies)**.** Assume $\delta_k \in \mathcal{H}_k$, given by the image of a continuous $\mathcal{F}_k$-measurable map $H_k : \mathbb{R}^{d(k+1)} \to \mathbb{R}^d$, i.e.

$$\mathcal{H}_k := H_k(\mathbb{R}^{d(k+1)}), \text{ where } H_k(0) = 0.$$

Thus, one is able to, for example also, hedge with illiquid instruments.

An example of a transaction cost is proportional transaction costs.

Example 2 (Proportional transaction costs)**.** For c_k^i, define $c_k(n) := \sum_{i=1}^{d} c_k^i S_k^i |n^i|$, where S_k^i are the mid-prices of instrument i at time t_k and n^i the amount traded.

This can be thought of as follows: Suppose you want to buy $n^1 = 100$ AAPL stock ($S_k^1 = \$177$ at the time of writing) at your local online broker. In order for the broker to make their business profitable, they have to incur a transaction cost on the orders that you execute with them. Suppose your broker charges a fee of 0.10% per price per stock traded ($c_k^1 = 0.001$). Then, the total transaction costs will be $0.001 \cdot 100 \cdot 177 = \$17.7$. One can easily imagine that if you are a bank with a much larger hedging book, these costs quickly become non-negligible.

Note that one can also model market impact, which will impact the asset distributions. An example outlined by the authors is as follows. Assume $I = S$, and we have a model of our market in the form of a conditional distribution $P(S_{k+1}|S_k)$. For a proportional impact parameter $\alpha > 0$, dynamics of S will be defined as $P(S_{k+1}|S_k(1 + \alpha(\delta_k - \delta_{k-1}))))$.

Pricing and Hedging under Convex Risk Measures

As mentioned before, the theory is built on finding optimality under convex risk measures, which are defined as follows.

Definition 1 (Convex Risk Measures)**.** Let $\mathcal{X}$ be the set of all real-valued random variables over Ω. Assume $X, X_1, X_2 \in \mathcal{X}$ represent asset positions. Call $\rho : \mathcal{X} \to \mathbb{R}$ a convex risk measure if:

1. Monotone decreasing: if $X_1 \geq X_2$, then $\rho(X_1) \leq \rho(X_2)$.
2. Convex: $\rho(\alpha X_1 + (1-\alpha)X_2) \leq \alpha\rho(X_1) + (1-\alpha)\rho(X_2)$ for $\alpha \in [0, 1]$.
3. Cash-Invariant: $\rho(X + c) = \rho(X) - c$ for $c \in \mathbb{R}$.

Let $\rho : \mathcal{X} \to \mathbb{R}$ be a convex risk measure, and for $X \in \mathcal{X}$. Then, the optimal strategy can be defined from the following optimisation problem:

$$\pi(X) := \inf_{\delta \in \mathcal{H}} \rho(X + (\delta \cdot S)_T - C_T(\delta).$$

The minimizer $\delta \in \mathcal{H}$ of the above is defined as an optimal hedging strategy. Now, we define the indifference price $p(Z)$ as the amount of cash that needs to be charged for the "agent" to be indifferent between the position $-Z$ and not doing so. Thus,

$$p(Z) := \pi(-Z) - \pi(0).$$

In a world without trading restrictions and costs, $p(Z)$ coincides with the price of a replicating portfolio.

This method accounts for previously studied frameworks for pricing in discrete time markets, for example, exponential utility indifference pricing [3] and optimized certainty equivalents [1]. These frameworks can be used to find convex risk measures, such as the following.

Example 3 (OCE $\to$ Conditional Value at Risk (CVaR))**.** Under the optimized certainty equivalents, we can define a convex risk measure ρ from a loss function $\ell : \mathbb{R} \to \mathbb{R}$, by setting

$$\rho(X) := \inf_{\omega \in \mathbb{R}} \{\omega + \mathbb{E}[\ell(-X - \omega)]\}, \quad X \in \mathcal{X}. \tag{1}$$

Let $\alpha \in (0, 1)$ and $\ell(x) := \frac{1}{1-\alpha} \max(x, 0)$. Inserting ℓ into the above definition for ρ, we get a the risk measure called CVaR.

Note that $-\rho$ is a convex risk measure.

As previously mentioned, we omit some of the universal approximation theorems and jump straight into the main ideas behind how we can numerically calculate optimal hedging strategies using neural networks. Let $\mathcal{NN}^{\sigma}_{\infty,d_0,d_1}$ be the set of neural networks that map from $\mathbb{R}^{d_0} \to \mathbb{R}^{d_1}$ with the activation function σ. Now, let $\{\mathcal{NN}_{M,d_0,d_1}\}_{M \in \mathbb{N}}$ be a sequence of subsets of $\mathcal{NN}^{\sigma}_{\infty,d_0,d_1}$ where

- $\mathcal{NN}_{M,d_0,d_1} \subset \mathcal{NN}_{M+1,d_0,d_1}$ for all $M \in \mathbb{N}$.
- $\bigcup_{M \in \mathbb{N}} \mathcal{NN}_{M,d_0,d_1} = \mathcal{NN}^{\sigma}_{\infty,d_0,d_1}$.
- For any $M \in \mathbb{N}$, one has $\mathcal{NN}_{M,d_0,d_1} = \{F^{\theta} : \theta \in \Theta_{M,d_0,d_1}\}$ with $\Theta_{M,d_0,d_1} \subset \mathbb{R}^q$ for some $q \in \mathbb{N}$ (depending on M).

Recall information available in our market at t_k is described by $I_0, \ldots, I_k$. Thus, we get the following semi-recurrent deep neural network structure for unconstrained trading strategies

$$\mathcal{H}_M := \{(\delta^{\theta}_k)_{k=0,\ldots,n-1} \in \mathcal{H}^u : \delta^{\theta}_k = F^{\theta_k}(I_0, \ldots, I_k, \delta_{k-1}), \theta_k \in \Theta_{M,r(k+1)+d,d}\}.$$

Skipping some steps outlined in the paper, the above ultimately leads to the following optimisation problem

$$\pi^M(X) := \inf_{\delta \in \mathcal{H}_M} \rho(X + (H \circ \delta \cdot S)_T - C_T(H \circ \delta) \tag{2}$$

$$= \inf_{\theta \in \Theta_M} \rho(X + (H \circ \delta^{\theta} \cdot S)_T - C_T(H \circ \delta^{\theta})), \tag{3}$$

where $\Theta_M = \prod_{k=0}^{n-1} \Theta_{M,r(k+1)+d,d}$. Thus, the infinite-dimensional problem of finding an optimal hedging strategy is reduced to the finite-dimensional constraint problem of finding optimal parameters for the neural network. The authors go further in-depth to show that strategies in $\mathcal{H}$ are approximated arbitrarily well by strategies in $\mathcal{H}_M$ [2].

Now the questions boils down to: how can we calculate an optimal parameter $\theta \in \Theta_M$? The authors focus on the case where ρ is an OCE risk measure, with no trading costs, and we will do the same to illustrate the idea better. The general case can be found in their paper [2].

Inserting equation (1) into equation (3), the optimization problem can be written as follows

$$\pi^M(-Z) = \inf_{\bar{\theta} \in \Theta_M} \inf_{\omega \in \mathbb{R}} \left\{ \omega + \mathbb{E}[\ell(Z - (\delta^{\bar{\theta}} \cdot S)_T + C_T(\delta^{\bar{\theta}}) - \omega)] \right\} = \inf_{\theta \in \Theta} J(\theta),$$

where $\Theta = \mathbb{R} \times \Theta_M$ and for $\theta = (\omega, \bar{\theta}) \in \Theta$,

$$J(\theta) := \omega + \mathbb{E}[\ell(Z - (\delta^{\bar{\theta}} \cdot S)_T + C_T(\delta^{\bar{\theta}}) - \omega)].$$

To find a minimum of J, one can use some variant of a gradient descent algorithm, and the authors consider Adam in their applications (we do too) [4].

Combining the theory, the general algorithm for Deep Hedging is as follows.

Algorithm 1 General Algorithm Deep Hedging

1: Simulate N paths of S + add any information I
2: Transform I into features (can include S)
3: Train NN to find $\pi^M(Z)$ where loss function is derived from a convex risk measure ρ

The actual architecture of the network is kept the same at 3 hidden layers, hidden dimensions equal to 15, with $ReLU(\max\{x, 0\})$ activation functions. The varying parameter are the features that the network takes in, with examples illustrated below. These depend on the use case of the hedger, i.e. FFN would be more suitable for path-dependent options like barrier options, as the agent sees the smallest or largest value in the hedge path's history.

Example 4 (Example networks)**.** Let H_t be the price of an equity at t, then potential networks could be:

Name	Feature Vector
Simple	$f_t(H_t)$
Recurrent	$f_t(H_t, \delta_{t-1})$
FFN	$f_t(H_t, \min_{t'<t} H_{t'}, \max_{t'<t} H_{t'})$

Experiments

The authors have figures in their paper to illustrate the workings of the algorithm. We wanted to try and replicate some of these plots and play around with the algorithm ourselves. As a result, to gain a deeper understanding and appreciation of the framework, we implemented our own codebase from scratch that allows one to test the algorithm themselves.

Note that in some of our experiments, we use slightly different settings from the authors, predominantly because of computational constraints.

Agents

- **NakedAgent:** Performs 0 trades (baseline).
- **DeltaAgent:** Always rebalance to the delta of the contingent claim derived via Black-Scholes model.

- **SimpleAgent:** Trades via neural network outlined in Training. Features are current market prices of hedging instruments.
- **RecurrentAgent:** Trades via neural network outlined in Training. Features are current market prices of hedging instruments and the current positions.

Training

- **Paths:** 2 000 000
- **Epochs:** 200
- **Optimizer:** Adam with 0.005 learning rate [4]
- **Network:** 2 hidden layers with 15 neurons each. ReLU activation function

Setting 1 (Black-Scholes World Baseline)**.** To validate whether or not Deep Hedging works, we first look at the baseline case of delta hedging in the Black-Scholes world.

Claim: European Call ($z_T = (S_T - K)^+$, $K = S_0$)

Hedge: Underlying stock (GBM($S_0 = 1, \sigma^2 = 0.1, r = 0$))

T: 1 month with daily rebalancing

ρ**:** Worst-Case: inf X

Costs: 0

We first examine why we want to hedge in the first place. To do so, we plot P&L distribution of the DeltaAgent against our NakedAgent.

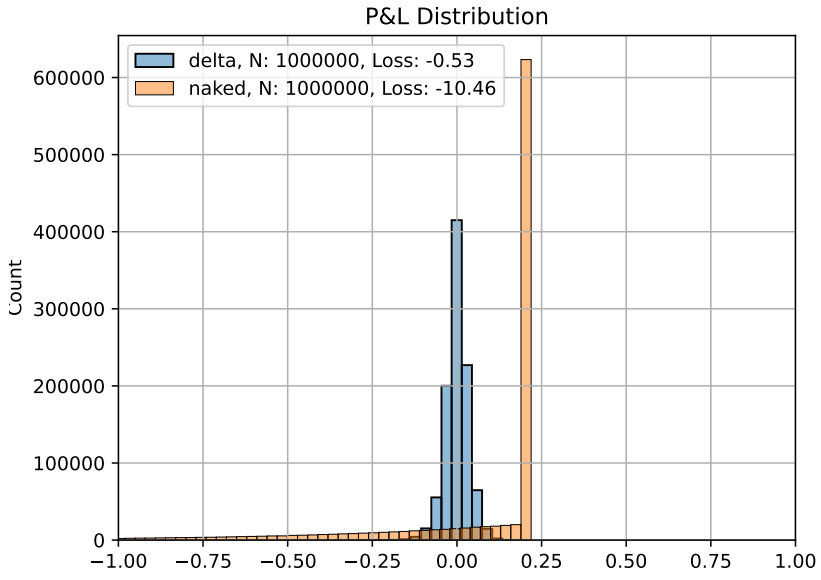

Figure 1: Setting 1 - DeltaAgent vs NakedAgent (why do we hedge in the first place?)

As we can see, despite the fact that most of the time the NakedAgent makes more money, it has a much longer left tail. This contrasts the stable distribution of the DeltaAgent centered around 0, with no visible large left tails. This nasty left tail is the reason why many funds have gone out of business, and why properly hedging is an essential part of a financial institution.

Now that we've seen why we hedge, we would like to compare how well the deep hedging networks do against delta hedging. We also would like to see how different architectures compare, so, we plot the P&L distributions for the SimpleAgent against the RecurrentAgent. For sake of brevity, we only plot the RecurrentAgent against the DeltaAgent, as we will explain later that the RecurrentAgent performs better in general than the SimpleAgent.

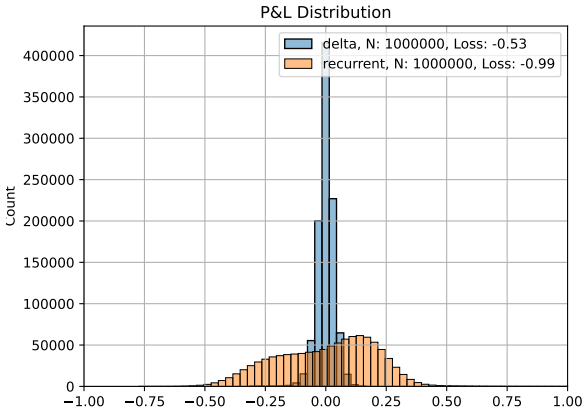

Figure 2: DeltaAgent vs RecurrentAgent.

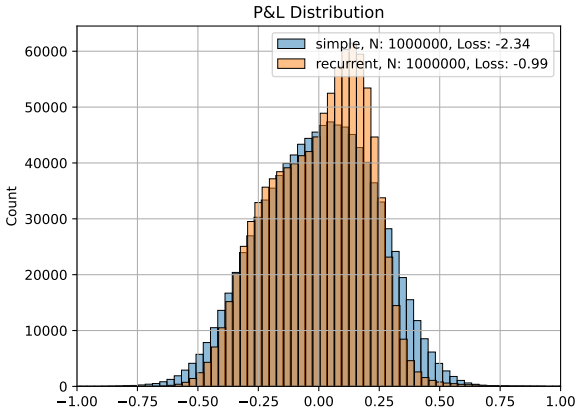

Figure 3: SimpleAgent vs RecurrentAgent.

In the plot above, we observe that the RecurrentAgent does, in general, learn how to hedge. While it may not have as sharp of a distribution as the DeltaAgent (i.e. it has more variance), we can safely assume that the mean is roughly centred around 0. If we were to train this for longer, with more paths, this distribution will look more and more like the one of the DeltaAgent.

As expected, we see that the RecurrentAgent performs better than the SimpleAgent in the sense that the P&L distribution of the RecurrentAgent has a lower standard deviation and a higher peak around 0. This is due to the fact that the RecurrentAgent has access to its positions in certain assets, giving it more information to train and act upon.

To get an even finer understanding of how the network agents operate, we plot a sample path of the SimpleAgent and one of the RecurrentAgent.

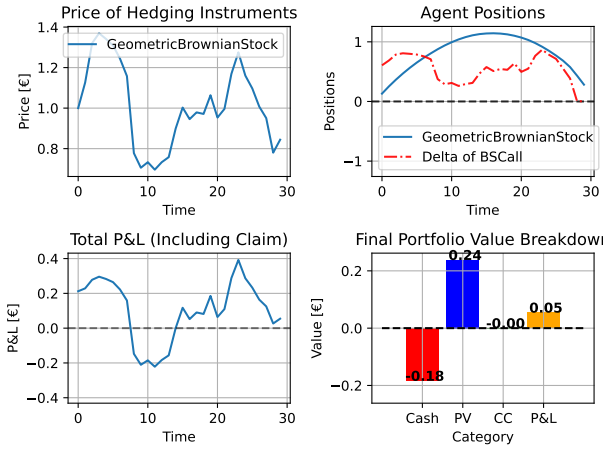

Figure 4: Sample path of SimpleAgent.

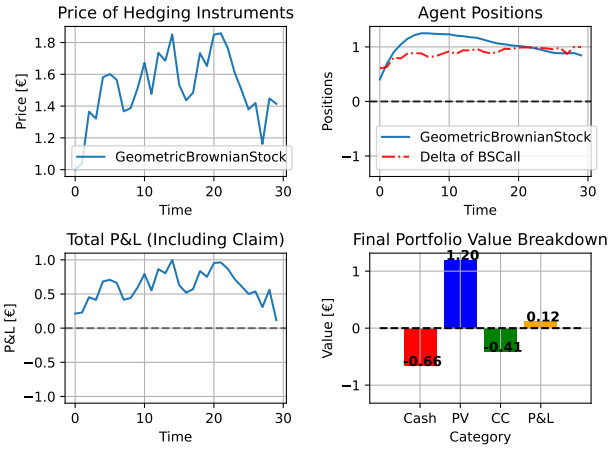

Figure 5: Sample path of RecurrentAgent.

We see from the above plots that the agents in general learn to buy the underlying as it sells a call. The RecurrentAgent seems to predict the current delta (red dotted line) of the call a bit better than the SimpleAgent. All in all, we think it is safe to conclude that the agents do actually learn to delta hedge to a certain extent.

Setting 2 (BS World with Transaction Costs)**.** Now we venture into the realm of incomplete markets, and add proportional transaction costs to trading. To begin our investigation, we start with a relatively low proportional cost of 2%.

Claim: European Call ($z_T = (S_T - K)^+$, $K = S_0$)

Hedge: Underlying stock (GBM($S_0 = 1, \sigma^2 = 0.1, r = 0$))

T: 1 month with daily rebalancing

ρ**:** Worst-Case: inf X

Costs: ProportionalCost(0.02) (2 %)

Again, we compare the networks against delta hedging and against themselves. Like in setting 1, we plot the DeltaAgent against RecurrentAgent as well as SimpleAgent against RecurrentAgent.

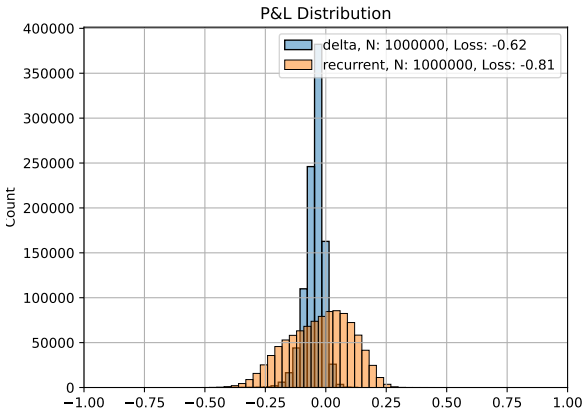

Figure 6: DeltaAgent vs RecurrentAgent.

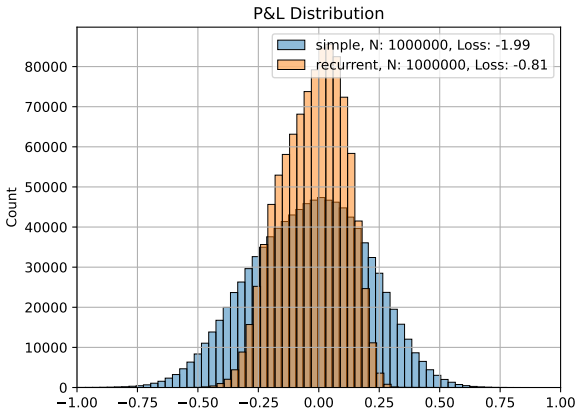

Figure 7: SimpleAgent vs RecurrentAgent.

We get similar results here to the previous setting. However, the distributions are obviously shifted due to these costs. We now see that the mean of the DeltaAgent is shifted more to the left than the one of the RecurrentAgent. But, what we see here predominantly is that the RecurrentAgent performs much better than the SimpleAgent. It has a tighter distribution around its mean and a lot less heavy on the tails.

Again, we illustrate a sample path of the RecurrentAgent below.

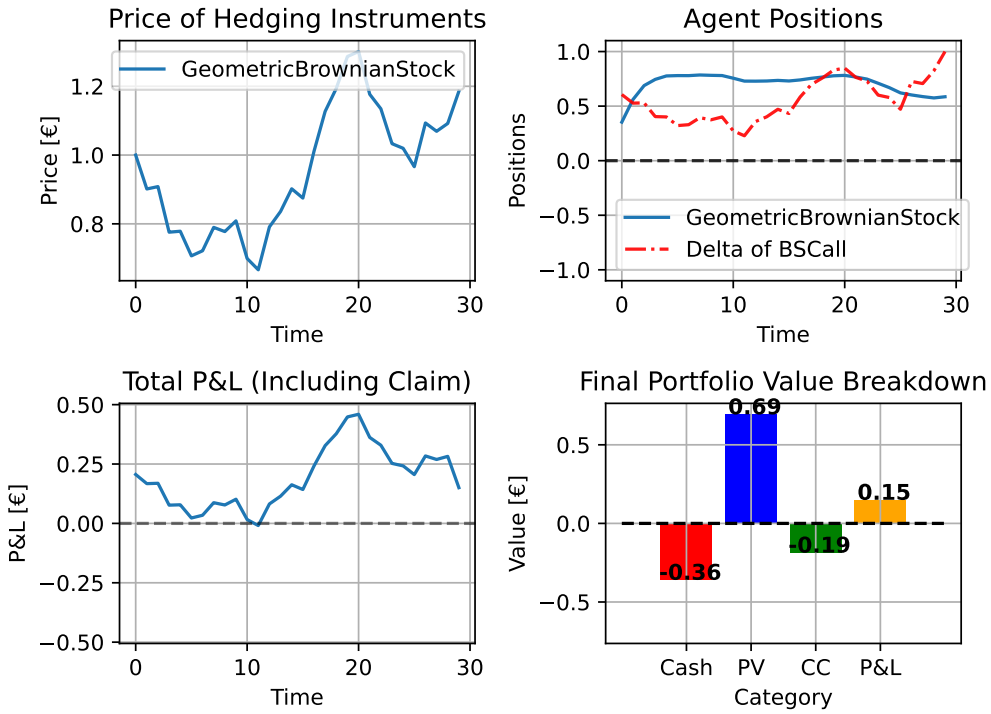

Figure 8: Sample path of RecurrentAgent.

Again, the RecurrentAgent learns to delta hedge, and we see that it learns to take a positive delta position (long the underlying) as it sells a call option. The agent's positions are flatter, indicating that it recognises

that the transaction costs can eat away at its PnL.

Setting 3 (BS World with Extreme Costs)**.** In the previous setting, we mainly saw the differences between the networks. To really see if the network agents get an edge over daily delta hedging, we bump up the transaction costs to extreme values, namely 25% proportional costs.

Claim: European Call ($z_T = (S_T - K)^+$, $K = S_0$)

Hedge: Underlying stock (GBM($S_0 = 1, \sigma^2 = 0.1, r = 0$))

T: 1 month with daily rebalancing

ρ**:** Worst-Case: inf X

Costs: ProportionalCost(0.25) (25 %)

Again, we plot the PnL distributions of the same agents.

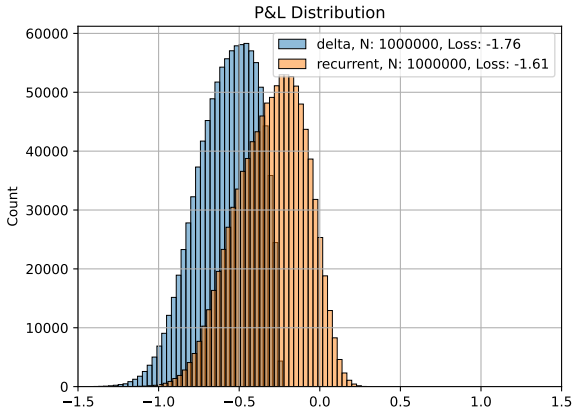

Figure 9: DeltaAgent vs RecurrentAgent.

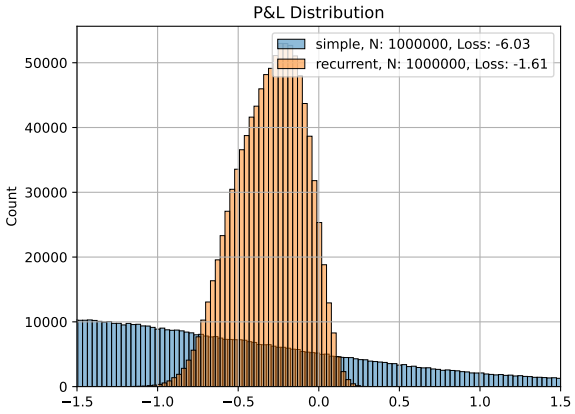

Figure 10: SimpleAgent vs RecurrentAgent.

Unlike before, there is a very clear edge for the RecurrentAgent against the DeltaAgent. Its PnL distribution is shifted more towards positive values than the DeltaAgent, while maintaining roughly the same standard deviations. For sake of completeness, we also plot the RecurrentAgent against the SimpleAgent, but we see a very wide distribution of the SimpleAgent, indicating that it has not managed to learn how to properly delta hedge in this setting.

To see behaviour, we plot a sample path of the RecurrentAgent.

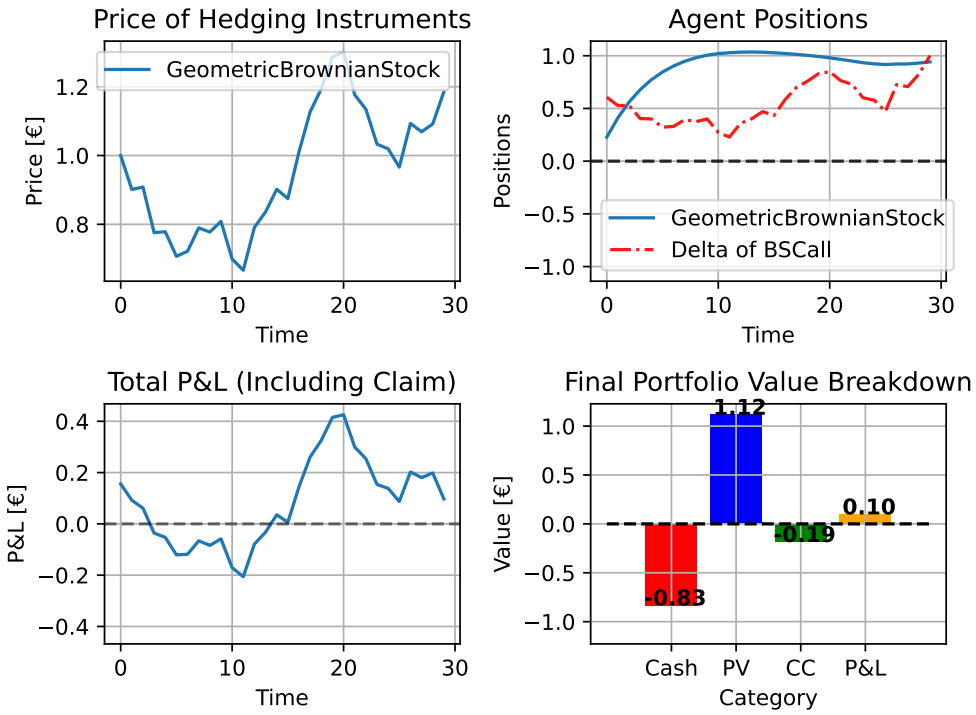

Figure 11: Sample path of RecurrentAgent.

Similar to the previous setting, the RecurrentAgent learns not to change its position too much, preferring to stay flat, even more so than before.

Conclusion

The deep hedging framework developed by Buehler et al. demonstrates remarkable capabilities, such as learning to effectively delta hedge and manage transaction costs in a more efficient manner in a GBM world. However, it's important to acknowledge certain limitations inherent in this approach. Notably, the agent requires retraining each time a parameter is adjusted, which could be a resource-intensive process. In addition, the process of creating a robust neural network tends to be time-consuming, and the choice of network architecture significantly influences performance outcomes. Looking ahead, there are promising avenues to explore in future iterations of this deep hedging approach. Incorporating varying market dynamics into the learning process could offer more adaptive and resilient hedging strategies. The potential for enhancements in this field of financial machine learning appears to be vast, underscoring the exciting future prospects of deep hedging. Such advancements have already been made, and we hope to be able to keep further going down this rabbit hole.

Please take a look at our GitHub page for the full custom implementation:
github.com/appie-mathematics/Deep-Hedging

References

[1] Aharon Ben-Tal and Marc Teboulle. "An old-new concept of convex risk measures: the optimized certainty equivalent". In: *Mathematical Finance* 17.3 (2007), pp. 449–476.

[2] Hans Buehler et al. "Deep hedging". In: *Quantitative Finance* 19.8 (2019), pp. 1271–1291.

[3] A. Neuberger Hodges S. "Optimal replication of contingent claims under transaction costs". In: *Review Futures Market* 8 (1989), pp. 222–239. URL: https://cir.nii.ac.jp/crid/1573668925038107776.

[4] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].