Enhancing Delta Hedging with Neural Networks

Mark Bogorad, Joaquin Garay, Matthew Rubenstein, Thomas Shen

What Is Dynamic Hedging?

For any uncovered position in an investment portfolio, there is some variability in that position's return. Stock prices fluctuate, options depend on the behavior of their underlying assets, and fixed income positions face risks like credit or interest rate exposures. To hedge an uncovered position is to mitigate these risks by entering other offsetting investment positions. A "perfect hedge" eliminates risk entirely; however, perfect hedges are rarely feasible in the practical world, particularly when dealing with complex financial instruments.

When hedging, there is a distinction between static hedging and dynamic hedging. In a static hedge, an investor enters a hedging position and simply holds their position until a time of maturity. The hedge requires no further management. As an example, an oil producer who plans to sell oil in 3 months can sell a 3-month forward contract to eliminate their price risk. Their hedging process is over; they have implemented a static hedge by taking a short oil position that offsets their natural long position in oil. Dynamic hedging, on the other hand, requires continual rebalancing in order to maintain a desired risk profile. Often, this desired profile is risk neutrality. For stock options, this rebalancing often involves trading in the underlying asset. To illuminate this idea, we introduce the Black-Scholes model, which provides an analytical framework for option pricing and how that pricing depends on various underlying factors.

The Black-Scholes formula for a European call and put option is as follows:

For a call option:

$$C = S_0 e^{-qT}\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

For a put option:

$$P = -S_0 e^{-qT}\Phi(-d_1) + Ke^{-rT}\Phi(-d_2)$$

where:

$$d_1 = \frac{\ln\left(\frac{S_0}{K}\right) + \left(r - q + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}, \quad d_2 = d_1 - \sigma\sqrt{T}$$

Here, (S_0) is the current price of the underlying asset, (K) is the strike price, (T) is the time to maturity, (r) is the risk-free rate, (σ) is the volatility of the stock (assumed to be constant), (q) is the estimated continuous dividend yield of the stock, and $(\Phi(\cdot))$ is the cumulative distribution function of the standard normal distribution. The Greeks, derived as partial derivatives of the Black-Scholes formula, measure the sensitivities of an option's value to these inputs. Delta, which measures the sensitivity of an option's price to changes in its underlying asset's price $((\partial C/\partial S_0))$, is particularly significant in dynamic hedging, as it allows investors to hedge options with positions in underlying assets. This is often more practical than hedging options with other options, in large part due to the underlying asset's high level of liquidity.

One "delta-hedges" an option position by holding a negative-delta position in the underlying asset. This creates a delta-neutral portfolio of opposing positions in an option and its underlying asset, making the portfolio theoretically insensitive to movements in the underlying asset price.

Delta, however, is not a constant; it evolves over time. For example, as near-the-money options approach maturity, they become more sensitive to price changes. In contrast, deep in-the-money (ITM) or out-of-the-money (OTM) options exhibit less sensitivity when approaching maturity. Because of these evolving delta values, delta-hedging strategies must be dynamic.
The Black-Scholes model provides a foundation for how one might approach dynamic delta hedging. It gives us an understanding of how changes in underlying variables (like volatility, time to maturity, and underlying asset

price) affect an option’s price and its sensitivities. The goal of dynamic delta hedging is to maintain a delta-neutral portfolio, and by doing so, minimize exposure to directional price movements. By continuously adjusting the hedging position as delta evolves, traders can theoretically mitigate risk and adapt to market conditions. Further considerations for implementing this strategy will be explored in later sections.

How do People Treat Delta Hedging?

Delta hedging is widely regarded as a versatile risk management tool, with applications that have evolved significantly since the introduction of the Black-Scholes model in 1973. Originally designed to neutralize price sensitivity for individual options, it now is often used to hedge entire portfolios. Despite its usefulness however, in its original form, the basic Black-Scholes model left many problems unanswered, prompting the development of various model expansions, which are now used in practice.

For one, the Black-Scholes model assumes that any underlying asset maintains a constant volatility. From this assumption, the Black-Scholes model derives a lognormal distribution for any underlying asset, which does not reflect real world market behavior. In real market conditions, stock prices often exhibit "fat tails," with more frequent extreme events than a lognormal distribution would suggest. This mismatch can lead to significant errors in pricing and hedging. To account for this, stochastic volatility models were introduced, which adapt the Black-Scholes framework to incorporate volatility movements. Other extensions attempt to capture jumps in the market (resulting from news events, policy changes, etc.). These are called jump-diffusion models. Furthermore, statistical techniques are used to adjust the Black-Scholes model in order to align it with real-market data. Perhaps most notable is the nonlinear least squares (NLS) method. Lastly, there exists a wide set of numerical tools that can be adjust or eschew aspects of the Black-Scholes model as need. These include Monte Carlo methods, lattice methods and finite difference methods.

The Limitations of Non-ML Dynamic Hedging Approaches:

The Black-Scholes Model (BSM) and its expansions face several limitations when applied to dynamic hedging, stemming from both their analytical frameworks and the practical realities of their implementation.

One significant issue comes from the flaw of time-discretization. When Black-Scholes delta-hedge ratios don’t adapt continuously, they essentially function as first-order Taylor approximations for how the option price changes with respect to the underlying asset price. Since continuous rebalancing is impossible in practice, these approximation errors become inevitable; they become particularly significant when large stock price movements occur between rebalancing periods. As the time between rebalancing shortens, the accuracy of the approximation improves.

In practice, the operational costs of rebalancing further complicate dynamic hedging strategies. Frequent rebalancing incurs substantial transaction costs, including bid-ask spreads and fees, making near-continuous hedging impractical. Because of these costs, investors are incentivized to rebalance *less* frequently, but this exacerbates the approximation errors introduced by time-discretization. These competing constraints highlight the practical trade-offs between reducing risk and managing transaction costs.

Delta-hedging also leaves risk exposure with respect to other sensitivities, including higher order sensitivities like gamma. In the case of non-constant volatility, there is exposure to volatility movements, described by an option’s vega. To handle this particular problem, practitioners often adopt techniques like delta-vega hedging, which involves trading multiple options (as well as the underlying asset) in order to offset sensitivity to both delta and vega. This strategy, while theoretically sound, is operationally constrained by low liquidity and slippage, which can significantly increase transaction costs. The same is true of other advanced hedging strategies such as delta-gamma or delta-vega-gamma hedging. As a result, these methods ultimately remain expensive and highly dependent on market conditions.

As mentioned earlier, numerous models—such as NLS, stochastic volatility models, path-dependent lattice and finite difference methods, and jump-diffusion models—have been developed to address the limitations of the Black-Scholes model, particularly its assumption of constant volatility. While these models enhance pricing accuracy, they can introduce inconsistent pricing biases over time, leading to larger replication errors for delta-neutral portfolios. Ultimately, despite their advancements, these models still fall short of fully capturing market dynamics and the true behavior of asset prices over time. For example, Nonlinear Least Squares (NLS) minimizes the residuals between the optimal and calculated delta-hedge ratios by fitting an explicitly defined nonlinear model that minimizes the sum of squared residuals (SSR). However, NLS struggles in scaling well to high-dimensional data, introducing significant computational costs and susceptibility to overfitting. This paper introduces a dynamic hedging model similar to NLS, using a forward-neural-network (instead of a nonlinear function) that minimizes (and predicts) the residual between the model's and optimal delta hedge ratio. Neural networks, when combined with regularization techniques and parallel computing can scale better than nonlinear functions used by NLS.

In summary, the limitations of traditional dynamic hedging strategies are rooted in the simplifying assumptions of the models used and the operational constraints of real-world implementation. As markets evolve and become increasingly complex, machine learning offers a promising alternative for developing dynamic hedging strategies that can better adapt to the intricacies of financial markets. These methods will be explored in the following section.

The Evolution of Dynamic Hedging: How Machine Learning Has Surpassed Limitations

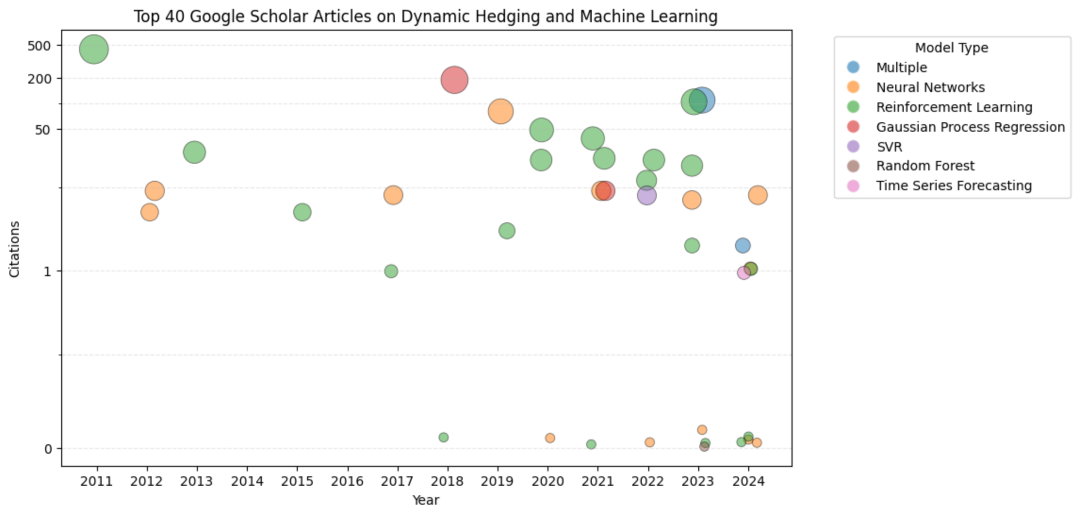
Dynamic hedging has evolved significantly since the emergence of the Black-Scholes model in 1973. During the 1980s, the derivatives market grew significantly, with static hedging strategies relying heavily on the Black-Scholes framework. These methods remained largely unchanged until the late 1990s, when the field of quantitative finance introduced mathematical solutions such as lattice methods, Monte Carlo simulations, and finite difference techniques (Broadie & Glasserman, 1997). These approaches marked the popularization of dynamic hedging by allowing strategies to better adapt to evolving market conditions. However, as discussed above, they were limited by their reliance on rigid assumptions, human discretionary inputs, and operational inflexibility, which hindered their performance in real-world scenarios.

The early 2000s brought the emergence of high-frequency trading (HFT), revolutionizing dynamic hedging strategies. HFT allowed for instantaneous and automated adjustments to hedging positions, reducing human error and enabling traders to account for nuances like market microstructure and slippage (Gerig, SEC paper). By operating at smaller time intervals, HFT enhanced responsiveness to market conditions. However, this approach introduced new risks, as demonstrated by events like the 2010 Flash Crash, where algorithmic trading caused extreme mispricing. Despite its advancements, the limitations of HFT at that time highlighted the need for more robust methods.

The adoption of machine learning in the early 2010s marked a pivotal shift in dynamic hedging. Support Vector Regression (SVR) enabled the modeling of non-linear relationships, improving forecasting accuracy for implied volatility and delta adjustments (Basak et al., 2011). Decision trees and random forests brought interpretability and robustness, allowing traders to segment data into volatility regimes and reduce overfitting in noisy environments (Breiman, 1984; Bali et al., 2009). These innovations enhanced the predictive power of dynamic delta hedging strategies, particularly in volatile markets, and laid the foundation for deeper learning approaches.

By 2015, neural networks emerged as transformative tools in dynamic hedging. Advances in computational power, data availability, and training techniques enabled architectures like feedforward networks, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) to model non-linear relationships and temporal dependencies in market data (LeCun et al., 2015). These methods excelled at managing path-dependent options, such as Asian and barrier options, by leveraging features like hierarchical data extraction and improved robustness through techniques like dropout (Hinton et al., 2012) and batch normalization (Ioffe & Szegedy, 2015). Neural networks significantly enhanced the accuracy of delta prediction, facilitating more efficient recalibration of hedging positions. However, their inability to directly incorporate transaction costs remained a limitation, later addressed by reinforcement learning (RL).

Reinforcement learning has become a cornerstone of modern dynamic hedging, with deep reinforcement learning (DRL) methods revitalizing its application. DRL integrates the decision-making capabilities of traditional RL with the feature extraction power of deep learning, enabling the development of adaptive, data-driven hedging strategies (Mnih et al., 2015). Models like Deep Q-Networks (DQN) and Policy Gradient Methods explicitly account for transaction costs and liquidity constraints, balancing risk and reward (Kolm & Ritter, 2020). DRL frameworks also leverage simulated environments to train on historical and synthetic data, ensuring robustness across diverse market conditions, including stress events. This capability mitigates the risks associated with relying solely on historical data, which may not fully represent future market behavior.



While neural networks and RL share transformative potential, their strengths and limitations differ. Neural networks excel at capturing complex, high-dimensional relationships and are particularly effective for predicting option delta and managing non-linear dependencies. However, they require frequent retraining to remain effective in dynamic environments. RL, in contrast, is inherently adaptive, refining optimal hedge ratios in response to evolving market conditions. Its ability to explicitly incorporate transaction costs makes it well-suited for real-world scenarios.

The evolution from static hedging to machine learning-driven strategies highlights the field's trajectory toward more adaptive and robust approaches. By integrating advanced computational techniques, practitioners can better navigate the intricacies of dynamic hedging, overcoming many of the limitations of traditional models while aligning strategies with real-world constraints.

Choice of Application and ML Model: Predicting Delta with Neural Networks

Many difficulties in the realm of dynamic delta hedging stem from operational problems, such as identifying optimal rebalancing times. Solutions to these operational problems are often firm-specific, pertaining to features like transaction costs, liquidity constraints, and capital requirements. This data is inaccessible to us, and for this reason we opt for another application: predicting optimal delta-hedge ratios

For this application, we chose to use a neural network, and there are several reasons that this decision was made. First, why would one use a neural network over reinforcement learning, which is the other foremost used machine learning technique in dynamic hedging? Reinforcement learning is particularly good at handling some of the operational problems described above; however, when it comes to predicting delta hedge ratios, there are some advantages to neural networks. Namely, neural networks are trained on a static set of data, and through gradient descent, they can converge upon a function that corresponds to a local minimum, if not an absolute minimum, in a specified loss function over that dataset. Reinforcement learning, on the other hand, uses an agent that explores, interacts with and alters its environment, making the training process dynamic and somewhat uncertain. As a result, reinforcement learning algorithms face more difficulty in converging upon stable decision-making strategies (which in this case would predict a delta-hedge ratio) than neural networks. In addition, reinforcement learning algorithms tend to be much more computationally expensive compared to neural networks.

Now, why are neural networks preferred in dynamic hedging applications over other less popular approaches like SVR and decision tree models? In brief, SVRs can model non-linear relationships with kernels. However, the necessity of pre-specifying kernels makes these models still less flexible than neural networks, which excel for their ability to find non-linear patterns in data. Additionally, the computational expense of an SVR grows very rapidly with the size and dimensionality of its dataset, making it impractical for applications beyond a certain scale. In contrast, neural networks scale well to large, high-dimensional data. Decision tree models, on the other hand, face their own limitations. For one, they are piece-wise functions; in this application, however, there are many reasons to believe that delta is continuous over significant areas of the relevant feature space. This makes the continuity of neural network outputs attractive in comparison. Furthermore, because trees split on one feature at a time, they are somewhat more constrained than neural networks in their ability to model feature interactions. Lastly, decision trees, being greedy algorithms, struggle to maintain performance at high dimensionalities of data, as the number of possible splits grows incredibly quickly with the number of features in a dataset. While ensemble methods can lessen the effect of some of these drawbacks, they do not eliminate the drawbacks completely. Furthermore, these ensemble tree methods come with significant computational expense at scale.

Specific Neural Network Application:

Drawing inspiration from "*Enhancing Black-Scholes Delta Hedging via Deep Learning*" by Chunhui Qiao and Xiangwei Wan, we use a supervised neural network approach to augment a Black-Scholes Delta Hedging strategy. By using neural networks to predict delta over discrete time intervals, our approach can potentially overcome two large limitations in traditional hedging strategies. First, successful predictions could result in better hedging errors than those produced by the truncation errors inherent when time-discretizing Black-Scholes. Second, our approach uses real market data, which helps to overcome the market assumptions implicit in many of the traditional hedging frameworks.

Our data, described in detail later, includes features across various SPX call options over many time steps (each trading day for 4 months). For each option, at each time step t, we compute the optimal delta hedge ratio, i.e., the amount that eliminates the hedging error at time t+1. This is calculated as:

$$\delta_{\text{optimal, t}} = \frac{\Delta C_{t+1}}{\Delta S_{t+1}} = \frac{C_{t+1} - C_t}{S_{t+1} \cdot e^{-q \cdot \Delta t} - S_t}$$

where C is the call price, S is the underlying asset price, and q is the estimated continuous dividend yield. We note that the variable q is not observable at time t and is a small assumption of our model.

Using a neural network to predict the optimal hedge ratio might seem intuitive, but as options near maturity, this ratio becomes discontinuous around the strike price, making it an unideal target for gradient descent algorithms.

To address this, we instead introduce a benchmark Black-Scholes Delta Hedging Ratio δ_{BS}, and take the residual between this value and the Optimal Delta Hedging Ratio δ_{optimal}:

$$target_i = \delta^{(i)}_{\text{optimal}} - \delta^{(i)}_{BS}$$

This residual provides a much smoother target for a neural network to predict on.

Finally, we handle our data cross-sectionally, training our neural network on a subset of our observation set, using a subset of our feature set.

Data

For our case study, we retrieved Bloomberg data for 12 different SPX Call Option Products over a four-month time window (8/5/2024 - 12/3/2024):

ID	Ticker	Type	Strike Price	Maturity Date	Underlying
1	SPX US 3/21/25 C5700 Index	Call	5700	2025-03-21	SPX
2	SPX US 3/21/25 C5800 Index	Call	5800	2025-03-21	SPX
3	SPX US 3/21/25 C5900 Index	Call	5900	2025-03-21	SPX
4	SPX US 3/21/25 C6000 Index	Call	6000	2025-03-21	SPX
5	SPX US 3/21/25 C6100 Index	Call	6100	2025-03-21	SPX
6	SPX US 3/21/25 C6200 Index	Call	6200	2025-03-21	SPX
7	SPX US 12/20/24 C5700 Index	Call	5700	2024-01-20	SPX
8	SPX US 12/20/24 C5800 Index	Call	5800	2024-01-20	SPX
9	SPX US 12/20/24 C5900 Index	Call	5900	2024-01-20	SPX
10	SPX US 12/20/24 C6000 Index	Call	6000	2024-01-20	SPX
11	SPX US 12/20/24 C6100 Index	Call	6100	2024-01-20	SPX
12	SPX US 12/20/24 C6200 Index	Call	6200	2024-01-20	SPX

The features for which we collected data include: (i) Underlying Asset Price, (ii) Underlying Asset Estimated Continuous Dividend Yield, (iii) Underlying Asset Risk-Free Rate (based on mid price), (iv) Option Mid Quote Option Implied Volatility (based on mid price), (v) Option Delta (based on mid price), (vi) Option Time to Maturity, and (vii) Option Moneyness.

From these features, we calculated the following features: (i) Black Scholes Option Price, (ii) Black Scholes Option Delta, (iii) Empirically Optimal Hedge Ratio, and (iv) Residual Between Empirically Optimal Hedge Ratio and Black Scholes Option Delta.

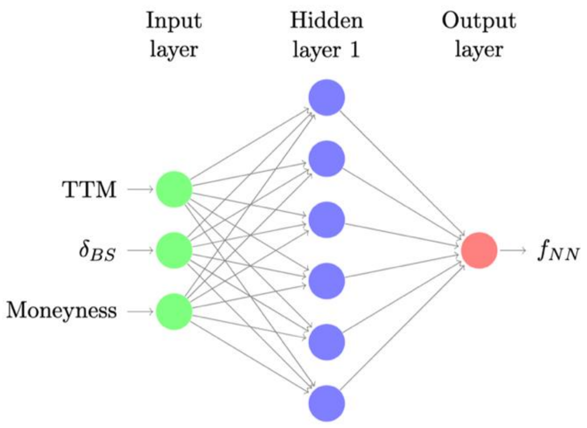
Hands On Application

Let $\mathbf{X}^{[0]} \in \mathbb{R}^{m\times d}$ be the matrix of input features, where m represents the batch size and d represents the number of input features, and $\mathbf{X}^{[2]} \in \mathbb{R}^{m\times 1}$ be the final output layer. For a one-hidden-layer FNN, let $\mathbf{X}^{[1]} \in \mathbb{R}^{m\times h}$ be the hidden layer, where h represents the number of neurons on the hidden layer. All layers are fully connected, with hidden layer weights $\mathbf{W}^{[1]} \in \mathbb{R}^{d\times h}$ and bias $\mathbf{b}^{[1]} \in \mathbb{R}^{1\times h}$, output layer weights $\mathbf{W}^{[2]} \in \mathbb{R}^{h\times 1}$ and $\mathbf{b}^{[2]} \in \mathbb{R}$. Using a sigmoid activation function g, the input-to-output mapping is calculated as follows:

$$\mathbf{X}^{[1]} = \sigma\underbrace{\left(\mathbf{X}^{[0]}\mathbf{W}^{[1]} + \mathbf{1}_m\mathbf{b}^{[1]}\right)}_{\mathbf{Z}^{[1]}}$$

$$f_{NN} \; = \; \mathbf{X}^{\lfloor 2 \rfloor} = \mathbf{X}^{\lfloor 1 \rfloor}\mathbf{W}^{\lfloor 2 \rfloor} + \mathbf{1}_m \mathbf{b}^{\lfloor 2 \rfloor}$$

In this hands-on example, we selected 3 features, namely time-to-maturity (TTM), implied BSM delta δ_{BS}, and Moneyness, and set 6 neurons for the fully connected hidden layer with sigmoid activation. Finally, the output layer consists of only one neuron without activation.

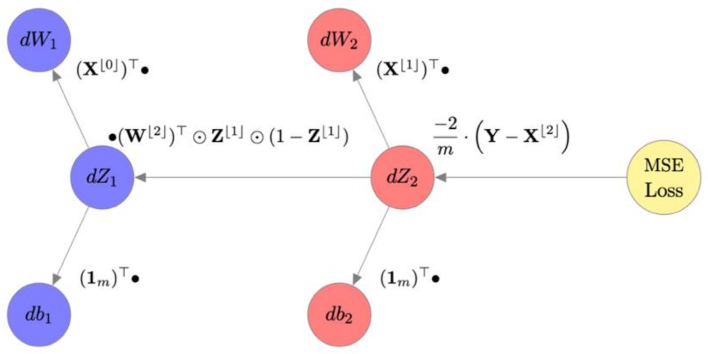


Back Propagation of a MLP

Let's consider the following MSE loss function,

$$\mathcal{L} = \frac{1}{m}\sum_i [y_i - f_{NN}(x_i)]^2 = \frac{1}{m}\left(\mathbf{Y} - \mathbf{X}^{\lfloor 2 \rfloor}\right)^\top \left(\mathbf{Y} - \mathbf{X}^{\lfloor 2 \rfloor}\right),$$

if we also consider $\odot$ as element-wise matrix multiplication operator, then we can describe the backpropagation process for our FNN as the algorithm in the figure below.


dW_1
$(\mathbf{X}^{\lfloor 0 \rfloor})^\top \bullet$
dW_2
$(\mathbf{X}^{\lfloor 1 \rfloor})^\top \bullet$
$\bullet(\mathbf{W}^{\lfloor 2 \rfloor})^\top \odot \mathbf{Z}^{\lfloor 1 \rfloor} \odot (1 - \mathbf{Z}^{\lfloor 1 \rfloor})$
$\frac{-2}{m} \cdot \left(\mathbf{Y} - \mathbf{X}^{\lfloor 2 \rfloor}\right)$
dZ_1
dZ_2
MSE Loss
$(\mathbf{1}_m)^\top \bullet$
$(\mathbf{1}_m)^\top \bullet$
db_1
db_2

Once the gradients are computed, we update the parameters with a simple gradient descent. For instance,

$$\mathbf{W_{i+1}}^{\lfloor 1 \rfloor} \; = \; \mathbf{W_i}^{\lfloor 1 \rfloor} \; - \; \alpha \cdot \nabla \mathbf{W_i}^{\lfloor 1 \rfloor}.$$

Results

We selected 30 samples for training with no validation. We initialized the values of $\mathbf{W}^{\lfloor 1 \rfloor}$, $\mathbf{b}^{\lfloor 1 \rfloor}$, $\mathbf{W}^{\lfloor 2 \rfloor}$, $\mathbf{b}^{\lfloor 2 \rfloor}$ as random number sampled from a uniform distribution $\mathcal{U}\left(-\sqrt{k}, k\right)$ where $k = (features\ in)^{-1}$ and set the learning rate $\alpha = 0.1$.

$$\mathbf{W_0}^{[1]} = \begin{bmatrix} 0.441 & 0.479 & -0.135 \\ 0.530 & -0.126 & 0.117 \\ -0.281 & 0.339 & 0.509 \\ -0.424 & 0.502 & 0.108 \\ 0.427 & 0.078 & 0.278 \\ -0.082 & 0.445 & 0.085 \end{bmatrix}^{T}, \mathbf{b_0}^{[1]} = \begin{bmatrix} -0.270 \\ 0.147 \\ -0.266 \\ -0.068 \\ -0.234 \\ 0.383 \end{bmatrix}^{T}, \mathbf{W_0}^{[2]} = \begin{bmatrix} -0.322 \\ -0.188 \\ -0.115 \\ -0.245 \\ 0.039 \\ -0.403 \end{bmatrix}, \mathbf{b_0}^{[2]} = [0.369].$$

After the first epoch, we updated the parameters to

$$\mathbf{W_1}^{[1]} = \begin{bmatrix} 0.444 & 0.484 & -0.132 \\ 0.532 & -0.124 & 0.118 \\ -0.280 & 0.341 & 0.510 \\ -0.421 & 0.506 & 0.110 \\ 0.426 & 0.078 & 0.278 \\ -0.078 & 0.450 & 0.089 \end{bmatrix}^{T}, \mathbf{b_1}^{[1]} = \begin{bmatrix} -0.266 \\ 0.149 \\ -0.265 \\ -0.065 \\ -0.235 \\ 0.387 \end{bmatrix}^{T}, \mathbf{W_1}^{[2]} = \begin{bmatrix} -0.354 \\ -0.219 \\ -0.144 \\ -0.274 \\ 0.009 \\ -0.438 \end{bmatrix}, \mathbf{b_1}^{[2]} = [0.320],$$

and finally

$$\mathbf{W_2}^{[1]} = \begin{bmatrix} 0.446 & 0.488 & -0.130 \\ 0.533 & -0.121 & 0.120 \\ -0.279 & 0.343 & 0.511 \\ -0.420 & 0.509 & 0.112 \\ 0.426 & 0.077 & 0.278 \\ -0.076 & 0.455 & 0.091 \end{bmatrix}^{T}, \mathbf{b_2}^{[1]} = \begin{bmatrix} -0.265 \\ 0.150 \\ -0.264 \\ -0.064 \\ -0.235 \\ 0.388 \end{bmatrix}^{T}, \mathbf{W_2}^{[2]} = \begin{bmatrix} -0.369 \\ -0.230 \\ -0.157 \\ -0.286 \\ -0.004 \\ -0.454 \end{bmatrix}, \mathbf{b_2}^{[2]} = [0.303]$$

Regarding the losses, they steadily decreased

$$Loss_0 = 13.5854 \rightarrow Loss_1 = 13.5274 \rightarrow Loss_2 = 13.5166$$

These results were successfully replicated in Python using Torch; the parameter values and losses obtained at each iteration were the same with a precision on the worst case of 8.0×10^{-8}.

Future Extension

To extend this machine learning application beyond the scope of our current demonstration, several changes are necessary.

First, the feature set should be expanded to incorporate additional information. This expansion can include the other features in our dataset beyond the three used in the demonstration above. Additionally, the paper by Chunhui Qiao and Xiangwei Wan includes other Black-Scholes variables, such as Vega, Gamma, and Theta, which could provide deeper insights for the model to train on.

Second, the scope of the dataset must grow significantly. A robust approach would include options across a wider range of strike prices and, more critically, span years rather than months. Such extended dataset, capturing full market dynamics, would enable the model to learn patterns across varying market conditions. To put this into perspective, our dataset comprises 1,008 unique observations, while the Qiao and Wan study uses 2.073 million.

Lastly, the loss function should directly target the hedging problem. Currently, we minimize the Mean Squared Error (MSE) of the residual, but the ideal approach would minimize the actual hedging error, defined for each observation as:

$$\frac{1}{m} \cdot \sum_{i}^{m} \left(\Delta C_i - \left(\delta_{BS}^{(i)} + f_{NN}^{(i)}(x) \right) \Delta S_i \right)^2$$

This custom loss function would better align the neural network's optimization with the practical goal of reducing hedging errors.

Implementing these extensions—expanding features, scaling the dataset, and refining the loss function—would enhance the robustness and effectiveness of this application, making it better suited for real-world dynamic hedging challenges.

Conclusion

Dynamic hedging has advanced significantly with the integration of machine learning techniques, particularly neural networks, which address many limitations of traditional Black-Scholes-based strategies. These models excel at capturing complex, non-linear relationships and improving delta estimation. However, challenges remain, including a dependency on massive datasets, reduced interpretability, and the continued need for human discretion in modeling. In our hands-on implementation, we demonstrated how a feedforward neural network effectively models deviations between the Black-Scholes delta and observed hedging errors using features like time-to-maturity, implied delta, and moneyness. The results showed steady loss reduction, underscoring the practical potential of these approaches while highlighting opportunities for further refinement and application. As markets continue to evolve, machine learning-driven strategies will play an increasingly critical role in achieving robust and adaptive hedging solutions.

Member Contributions

The work done for this project was highly collaborative with each member contributing in many different ways. Below is a reduced summary of each member's work, noting the topics for which each member led the effort. Mark Bogorad: [Evolution of Dynamic Hedging, Conclusion]; Joaquin Garay: [Data Extraction and Hands-On manual implementation], Matthew Rubenstein: [What is Dynamic Hedging?, Bubble Chart, Choice of Application and Model, Specific Neural Network Application], Thomas Shen: [Limitations of Non-ML Dynamic Hedging Approaches, code implementation]

Citations

Basak, D., Pal, S., & Patranabis, D. C. (2007). *Support vector regression*. *Neural Information Processing: Letters and Reviews, 11*(10), 203–224. https://www.researchgate.net/publication/228537532_Support_Vector_Regression

Bloomberg. (2020, January 22). *The development of the Black-Scholes formula: Theory, research, and practice*. https://www.bloomberg.com/professional/insights/data/the-development-of-the-black-scholes-formula-theory-research-and-practice/

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees*. Chapman & Hall/CRC. https://api.pageplace.de/preview/DT0400.9781351460491_A31471984/preview-9781351460491_A31471984.pdf

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. Kluwer Academic Publishers. https://doi.org/10.1023/A:1010933404324

Broadie, M., & Glasserman, P. (1997). Monte Carlo simulation of option pricing and hedging. *In D. J. Bricker (Ed.), Monte Carlo: Methodologies and applications for pricing and risk management* (pp. 13–66). Springer.

FasterCapital. (2024, June 24). *Delta hedging: Minimizing risk in Black-Scholes option trading*. https://fastercapital.com/content/Delta-Hedging--Minimizing-Risk-in-Black-Scholes-Option-Trading.html#Limitations-of-Delta-Hedging

Gerig, A. (2012). *High-frequency trading synchronizes prices in financial markets*. U.S. Securities and Exchange Commission. https://www.sec.gov/files/dera-wp-hft-synchronizes.pdf

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*. https://arxiv.org/abs/1207.0580

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv*. https://arxiv.org/abs/1502.03167

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*, 436–444. https://doi.org/10.1038/nature14539

Nandi, S., & Waggoner, D. F. (2000). *Issues in hedging options positions*. *Federal Reserve Bank of Atlanta Economic Review, 85*(1), 24–39. https://www.atlantafed.org/-/media/documents/research/publications/economic-review/2000/vol85no1_nandi-waggoner.pdf

Mishra, N. (2018). Building bridges: International trade law, internet governance, and the regulation of data flows. *Vanderbilt Journal of Transnational Law* (forthcoming). NUS Centre for International Law Research Paper No. 19/09. SSRN. https://ssrn.com/abstract=3263271

Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*, 529–533. https://doi.org/10.1038/nature14236

Qiao, C., & Wan, X. (2024). Enhancing Black-Scholes delta hedging via deep learning. *arXiv*. https://arxiv.org/abs/2407.19367

Ruf, J., & Wang, W. (2019). Neural networks for option pricing and hedging: A literature review. *Journal of Computational Finance*. SSRN. https://ssrn.com/abstract=3486363

Zhang, Z., Zohren, S., & Roberts, S. (2020). Deep learning for portfolio optimisation. SSRN. https://ssrn.com/abstract=3613600