Pricing and Hedging European Options in Incomplete Markets with Neural Networks

Nicolas Baradel*

March 11, 2025

1 Introduction

We explore an approach to pricing and hedging European options in an incomplete financial market using machine learning techniques. We consider two tradable assets: the underlying and a liquid call option on the underlying, where the latter is designed to capture the stochastic volatility associated with the underlying.

Our methodology builds on the self-financing condition of a portfolio, initialized with a premium and constructed to cover a payoff at maturity. This direction has been explored in prior work leveraging machine learning. For instance, [2] employs a Q-learning reinforcement learning approach, focusing solely on the stock price within a self-financing framework. In [1] the authors incorporate transaction costs, deriving prices and strategies via convex risk measures using deep reinforcement learning methods.

In our approach, we utilize a single neural network to simultaneously determine the option price and its hedging strategy. In a complete market, a self-financing portfolio can perfectly hedge a European option, with the hedging strategy derived as the gradient of the price function. Here, we extend this concept to an incomplete market by training a neural network to represent the price, where its gradient serves as the hedging strategy. We optimize this network by minimizing a loss function based on the self-financing condition, incorporating both the price and the hedge. To evaluate the method's effectiveness, we compare the resulting Profit and Loss (P&L) distribution against that of a baseline Black-Scholes hedge for various standard options.

The paper is structured as follows. Section 2 introduces the market model, featuring an underlying asset with stochastic volatility and a stochastic correlation with its volatility, alongside two tradable assets, rendering the market incomplete. Section 3 details the neural network architecture, which provides an option price and a hedging strategy via its gradient, and explains the estimation process using the self-financing condition. Section 4 demonstrates that surpassing a simple Black-Scholes strategy reliant solely on the underlying in an idealized market (i.e., without transaction costs

*Inria, CMAP, CNRS, École polytechnique, Institut Polytechnique de Paris, 91200 Palaiseau, nicolas.baradel@polytechnique.edu.

or liquidity constraints) is challenging. We then propose enhancements, including an endogenous terminal condition and exploitation of payoff-specific properties, yielding results that significantly outperform the Black-Scholes benchmark.

2 The mathematical framework

Let $\Omega := C([0,T],\mathbb{R}^3)$ represent the space of continuous functions from $[0,T]$ to $\mathbb{R}^3$, where $T>0$ and functions start at value 0 at time 0. We define the canonical process by $W(\omega)=\omega$, and let $\mathbb{P}$ denote the Wiener measure on the Borel sets of Ω. Consequently, $W=(W^i)_{1\le i\le 3}$ consists of three independent Brownian motions.

2.1 The market model

We introduce a market model defined by the following triplet of stochastic processes.

Definition 2.1. *Let $\mu\in\mathbb{R}, a>0, \sigma_\circ>0, \xi>0, \gamma\in[0.5,1], b>0, p_\circ\in\mathbb{R}, \chi>0$. For initial conditions $(x,\sigma,p)\in(\mathbb{R}_+)^2\times\mathbb{R}$ at time $t\in[0,T]$, and for $s\in[t,T]$, the processes are given by:*

$$\begin{aligned} X_s^{t,x} &= x+\int_t^s \mu X_u^{t,x}du+\int_t^s \Sigma_u^{t,\sigma}X_u^{t,x}dW_u^1,\\ \Sigma_s^{t,\sigma} &= \sigma+\int_t^s -a(\Sigma_u^{t,\sigma}-\sigma_\circ)du+\int_t^s \xi(\Sigma_u^{t,\sigma})^\gamma d(\rho_u^{t,p}W_u^1+\sqrt{1-(\rho_u^{t,p})^2}W_u^2), \qquad (1)\\ P_s^{t,p} &= p+\int_t^s -b(P_u^{t,p}-p_\circ)du+\int_t^s \chi P_u^{t,p}dW_u^3, \end{aligned}$$

where the correlation process is defined as $\rho^{t,p}:=\tanh(P^{t,p})$.

In this model, $X^{t,x}$ represents the underlying price with stochastic volatility $\Sigma^{t,\sigma}$, which reverts to a long-term mean $\sigma_\circ$ at rate a. The volatility $\Sigma^{t,\sigma}$ is correlated with $X^{t,x}$ via a stochastic correlation $\rho^{t,p}$, driven by the process $P^{t,p}$, which itself exhibits mean reversion to $p_\circ$. The term $\rho^{t,p}W_u^1+\sqrt{1-(\rho^{t,p})^2}W_u^2$ ensures a unit-variance Brownian motion with correlation $\rho^{t,p}$ to W^1.

Remark 2.2. *The process $(X^{t,x},\Sigma^{t,\sigma},P^{t,p})$ in Definition 2.1 admits a unique strong solution under the specified conditions.*

For notational simplicity, we henceforth denote (X,Σ,P) as $(X^{t,x},\Sigma^{t,\sigma},P^{t,p})$.

2.2 The tradable assets

Of the processes introduced in Definition 2.1, not all are tradable assets. We now define those that are.

Definition 2.3 (Tradable assets)**.** *The market includes two tradable assets:*

- *The underlying X , as defined in* (1)*.*
- *A European call option $C(K)$ with strike $K>0$ where $K>0$ and maturity $T>0$.*

The price of the call option $C(K)$ at time $s \in [t, T]$ is modeled as the Black-Scholes price with instantaneous volatility Σ_s:

$$C_s(K) := BS(T - s, X_s, \Sigma_s, r, K),$$

where $BS(u, x, \sigma, r, K)$ denotes the Black-Scholes price of a European call option with time to maturity $u \in [0, T]$, underlying price $x > 0$, volatility $\sigma > 0$, interest rate $r \in \mathbb{R}$, and strike $K > 0$.

This framework establishes an incomplete market, as there are two tradable assets but three independent sources of randomness (the Brownian motions W^1, W^2, W^3). Consequently, perfect hedging of a derivative is unattainable, though our goal is to achieve an optimal hedge.

2.3 Objective

From the perspective of a seller of a European option, this study aims to determine:

- a price,
- a hedging strategy,

for an option with payoff $g(X_T, P)$ where P denotes a parameter (e.g. a strike price or a barrier level). To address this, we employ a neural network to simultaneously derive both the price and the hedging strategy.

3 The neural network framework

We seek to determine the price and hedging strategy for a European option with payoff $g(X_T, P)$ where $P \in \mathbb{R}^d$ for $d \geq 1$ represents the option's parameters (e.g., strike price or barrier level). To this end, we introduce a neural network:

$$(t, x, c, K, P) \mapsto N_\theta(t, x, c, K, P),$$

where θ denotes the trainable parameters, t is the time to maturity (with $t = 0$ at maturity), x is the asset price, c is the call option price, and $K > 0$ is the strike price of the hedging call option $C(K)$.

Rather than employing separate networks for the price and hedging strategy, we combine both within a single network. Given that the hedging strategy can be derived from the price function's derivatives in a complete market, we propose:

- N_θ represents the option price,
- $\partial_x N_\theta$ and $\partial_c N_\theta$ provide the hedging strategies for the underlying asset and the call option, respectively.

To implement this, we define a self-financing portfolio:

Definition 3.1 (Self-financing condition)**.** *Let V denote the value of a self-financing portfolio, with Δ^x and Δ^c representing the holdings in the underlying asset and the call option $C(K)$, respectively. For $t \leq s \leq T$, the self-financing condition is:*

$$V_s = V_t + \int_t^s r\left(V_u - \Delta_u^x X_u - \Delta_u^c C_u\right) du + \int_t^s \Delta_u^x dX_u + \int_t^s \Delta_u^c dC_u.$$

In discrete form, assuming constant holdings over $[t, s]$, this becomes:

$$V_s = e^{r(s-t)}\left(V_t - \Delta_t^x X_t - \Delta_t^c C_t\right) + \Delta_t^x (X_s - X_t) + \Delta_t^c (C_s - C_t).$$

We set the portfolio value at time t as $V_t := N_\theta(T-t, X_t, C_t, K, P)$. If the price is accurate and hedging is perfect, with $T > 0$ as maturity, we define:

$$\begin{aligned} \Delta_t^x &:= \partial_x N_\theta(T-t, X_t, C_t, K, P), \\ \Delta_t^c &:= \partial_c N_\theta(T-t, X_t, C_t, K, P), \\ N_\theta(0, X_T, C_T, K, P) &= g(X_T, P). \end{aligned} \tag{2}$$

In practice, due to the incomplete market and discrete hedging, N_θ satisfies approximately, for $\Delta t > 0$:

$$\begin{aligned} N_\theta(T-(t+\Delta)t, X_s, C_s, K) \approx\ & e^{r\Delta t}\left(N_\theta(T-t, X_t, C_t, K) - \Delta_t^x X_t - \Delta_t^c C_t\right) \\ & + \Delta_t^x (X_{t+\Delta t} - X_t) + \Delta_t^c (C_{t+\Delta t} - C_t), \\ N_\theta(0, X_T, C_T, K) \approx\ & g(X_T, P). \end{aligned} \tag{3}$$

This framework enables the network to jointly estimate the price and hedging strategy.

3.1 The loss function and estimation

In practice, we simulate trajectories of the processes defined in (1) as well as strike prices of the hedging option and the parameters of the option hedge. This yields n trajectories with a discretization over $m+1$ points.

To train the network, we simulate trajectories of the processes in (1), along with strike prices K and parameters P for the option. We generate n paths discretized at $m+1$ points: $0 = t_0, t_1, \ldots, t_m = T$, yielding:

$$\left(t_j, X_{t_j}^i, C_{t_j}^i(K_j^i), C_{t_{j+1}}^i(K_j^i), K_j^i, P_j^i\right)_{0 \leq j \leq m}^{1 \leq i \leq n}. \tag{4}$$

Here, varies across each (i, j), so $Ct_{j+1}{}^i(K_j^i)$ uses the same strike as $C_{t_j}(K_j^i)$.

Based on (3), we define two loss components: the terminal condition loss,

$$\ell_T(\theta) = \sum_{i=1}^n \left(N_\theta(0, \cdot) - g(X_T^i, P_T^i)\right)^2.$$

and the path loss,

$$\begin{aligned} \ell_{\texttt{path}}(\theta) := \sum_{i=1}^n \sum_{j=0}^{m-1} \Big(\Big(& N_\theta(T-t_j, \cdot) - \Delta_{t_j}^x X_{t_j}^i - \Delta_{t_j}^c C_{t_j}^i \Big) e^{r(t_{j+1}-t_j)} \\ & + \Delta_{t_j}^x (X_{t_{j+1}}^i - X_{t_j}^i) + \Delta_{t_j}^c (C_{t_{j+1}}^i - C_{t_j}^i)(K_j^i) - N_\theta(T-t_{j+1}, \cdot) \Big)^2. \end{aligned}$$

where $\Delta^x_{t_j} = \partial_x N_\theta(T - t_j, X^i_{t_j}, C^i_{t_j}, K^i_j, P^i_j)$ and similarly for $\Delta^c_{t_j}$. The total loss is:

$$\ell(\theta) := \ell_{\texttt{path}}(\theta) + \lambda_T \ell_T(\theta), \tag{5}$$

with $\lambda_T > 0$ as a weighting factor.

We estimate θ via gradient descent, using mini-batches to reduce computational cost. From $n \times m$ data points, we select a subset $\kappa \ll n \times m$, denoted $\Lambda(k) \subset \{1, \ldots, n\} \times \{0, \ldots, m-1\}$, for each iteration k. The algorithm is:

Definition 3.2 (Algorithm for estimating θ)**.** *Starting with $\theta_{0,0}$, for $k = 1, ..., M_0$:*

1. *Simulate $\Lambda(k)$ to define the mini-batch.*
2. *For $i = 1, ..., M_1$:*
   (a) *Compute $\ell(\theta_{k,i}, \Lambda(k))$ and its gradient,*
   (b) *Update $\theta_{k,i}$ via gradient descent.*

The path loss over a mini-batch Λ is:

$$\ell_{\texttt{path}}(\Lambda, \theta) := \sum_{(i,j)\in\Lambda} \Big(\Big(N_\theta(T - t_j, \cdot) - \Delta^x_{t_j} X^i_{t_j} - \Delta^c_{t_j} C^i_{t_j} \Big) e^{r(t_{j+1} - t_j)} + \Delta^x_{t_j}(X^i_{t_{j+1}} - X^i_{t_j}) + \Delta^c_{t_j}(C^i_{t_{j+1}} - C^i_{t_j})(K^i_j) - N_\theta(T - t_{j+1}, \cdot) \Big)^2.$$

Then:

$$\ell(\theta) := \ell_{\texttt{path}}(\theta) + \lambda_T \ell_T(\theta).$$

3.2 Evaluation criterion

To evaluate the neural network's performance, we consider an initial state at $t = 0$ (time to maturity T), with underlying asset price x, call option price c, strike K, and option parameter P. The option price is given by $N_\theta(T, x, c, K, P)$. Over a discretized trajectory $(t_j)_{0\le j\le m}$ with $t_0 = 0$ and $t_m = T$, we apply the hedging strategy:

$$\Delta^x_{t_j} := \partial_x N_\theta(T - t_j, X_{t_j}, C_{t_j}, K, P), \quad \Delta^c_{t_j} := \partial_c N_\theta(T - t_j, X_{t_j}, C_{t_j}, K, P),$$

and at maturity, the terminal payoff $g(X_T, P)$ is delivered.

The Profit and Loss (P&L) for a single trajectory i is defined as the initial price plus hedging gains, discounted to present value, minus the discounted payoff. Assuming continuous compounding with interest rate r, it is:

$$P\&L_i := N_\theta(T, x, c, K, P) + \sum_{j=0}^{m-1} \Big(\Delta^x_{t_j}(X_{t_{j+1}} - X_{t_j}) + \Delta^c_{t_j}(C_{t_{j+1}} - C_{t_j}) \Big) e^{-rt_j} - g(X_T, P) e^{-rT}.$$

Alternatively, the P&L could be computed at maturity by multiplying by e^{rT}.

For n simulated trajectories $(X^i_{t_j}, C^i_{t_j})_{0\le j\le m, 1\le i\le n}$ from the model in (1), we obtain $(P\&L_i)_{1\le i\le n}$. The empirical distribution of these P&L values is then analyzed to assess the hedging effectiveness.

4 Applications and improvements

We begin by testing a European call option with terminal payoff $g(X_T, P) := (X_T - P)^+$ where $X_T > 0$ is the asset price at maturity and $P > 0$ is the strike price. We simulate trajectories using the stochastic volatility model from Definition 2.1, with parameters specified in Table 4.

Parameters	Value
μ	0
a	5
$\sigma_\circ$	0.2
ξ	0.5
γ	0.7
b	5
$p_\circ$	-0.3
χ	0.5

Table 1: Parameters for the model in Definition 2.1.

4.1 Initial results on the vanilla option

We employ a neural network with three layers, each containing 32 neurons, and the hyperbolic tangent (tanh) activation function. The parameters θ are optimized using the Adam algorithm [3], implemented in Python with PyTorch [4]. To evaluate the hedging strategy's effectiveness, we compare its Profit and Loss (P&L) distribution to that of the Black-Scholes model.

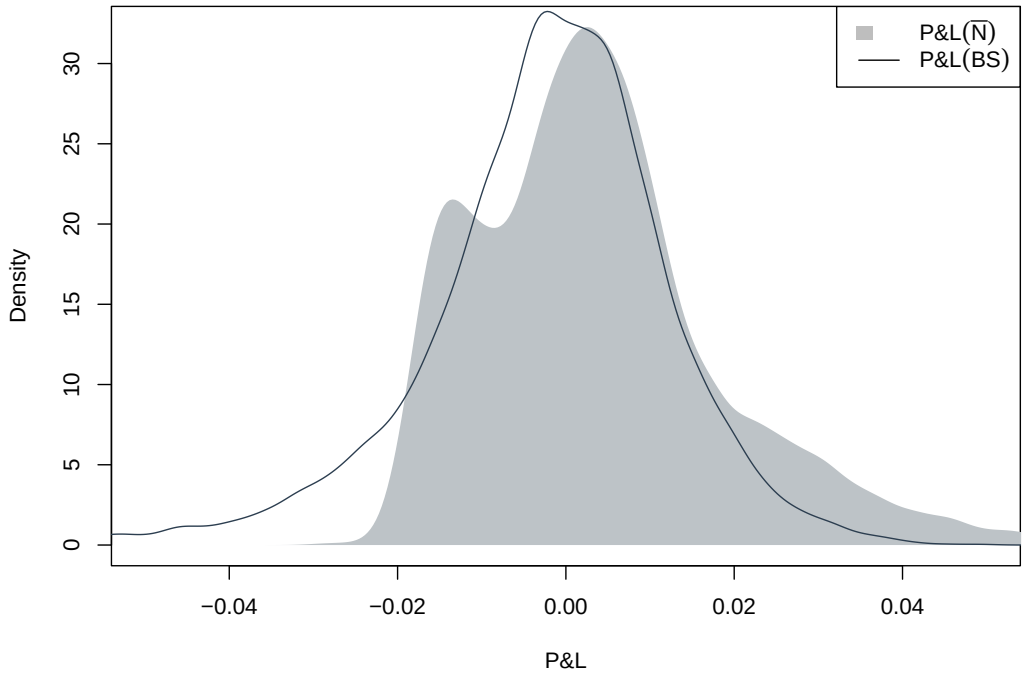

Figure 1: Comparison of empirical P&L distributions for the Black-Scholes hedging strategy and the neural network N.

The neural network's hedging strategy yields modest results. As shown in Figure 1, it mitigates some extreme negative P&L values but exhibits a dispersion similar to that of the Black-Scholes strategy, offering no substantial improvement in overall performance. To address this, we propose enhancements tailored to the specific payoff $g(X_T, P)$, sacrificing the general applicability of the approach for improved accuracy on this option type.

4.2 Making the terminal condition endogenous

For a vanilla call option, the terminal payoff $g(X_T, P) = (X_T - P)^+$ is non-differentiable with respect to the strike P at $X_T = P$. While the neural network N_θ is smooth everywhere, more complex options may exhibit even greater irregularity, such as discontinuities. To address this, we incorporate the terminal condition endogenously.

We introduce a smooth function $f(t, x, P)$, defined on $[0, T] \times (\mathbb{R}_+)^4$ and satisfying $f(0, x, P) = g(x, P)$ The modified price function is:

$$\overline{N}_\theta(T - s, x, c, K, P) := \frac{s}{T} f(T - s, x, P) + N_\theta(T - s, x, c, K, P), \tag{6}$$

For a vanilla call, a natural choice for f is the Black-Scholes formula with constant volatility $\sigma_\circ$, from Table 1, i.e. $f(u, x, P) = BS(u, x, \sigma_0, r, P)$ approximating the option price and matching $(x - P)^+$ at $u = 0$.

Thus, $\overline{N}_\theta$ represents the option price, with hedging strategies derived from $\partial_x \overline{N}_\theta$ and $\partial_c \overline{N}_\theta$. The weighting $\frac{s}{T}$ ensures:

- At maturity, $\overline{N}_\theta(0, x, c, K, P) := g(x, P)$ satisfied when $N_\theta = 0$ simplifying the network's task.
- As t is away of the maturity, f's influence diminishes; if f is a good approximation, $N_\theta = 0$ yields a reasonable price but risks trapping the optimization in a local minimum.

4.3 Incorporating general arbitrage bounds on the price

No-arbitrage principles impose bounds on option prices, independent of the underlying model. For a vanilla call option with interest rate $r \geq 0$, the price must satisfy $\overline{N}_\theta(T - t, X_t, C_t, K, P) \geq \left(X_t - Pe^{-r(T-t)}\right)^+$. We enforce this constraint within the neural network.

Given a lower bound $v(t, x, P)$ for the payoff $g(x, P)$, we define a violation loss over a mini-batch Λ:

$$\ell_{\texttt{VI}}(\theta, \Lambda) := \sum_{(i,j)\in\Lambda} \left(\left(\overline{N}_\theta(T - t_j, \cdot) - v(t_j, \cdot)\right)^- \right)^2,$$

where $x^- := \max(0, -x)$ penalizes values below the bound. The total loss becomes:

$$\ell(\theta, \Lambda) := \ell_{\texttt{path}}(\theta, \Lambda) + +\lambda_{\texttt{VI}}\ell_{\texttt{VI}}(\theta, \Lambda) + \lambda_T \ell_T(\theta). \tag{7}$$

with $\lambda_{\texttt{VI}} > 0$ weighting the arbitrage constraint. For the European call, we set $v(t, x, P) := \left(x - Pe^{-r(T-t)}\right)^+$.

4.4 Better results on the vanilla option

We employ the same neural network as before, incorporating two proposed enhancements: making the terminal condition endogenous and integrating the intrinsic value barrier. Specifically, f is the Black-Scholes formula with volatility $\sigma_\circ$ from Table 1, i.e. $f(u, x, P) = BS(u, x, \sigma_0, r, P)$, and $v(t, x, P) = \left(x - Pe^{-r(T-t)}\right)^+$.

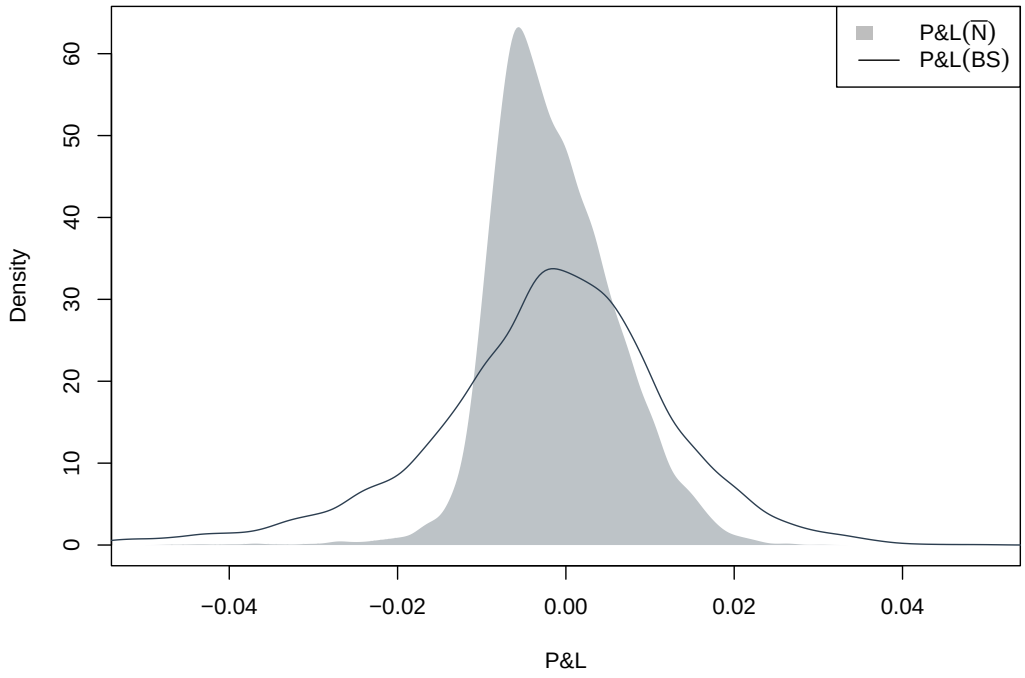

Figure 2: Comparison of empirical P&L distributions for the Black-Scholes hedging strategy and the neural network $\overline{N}$.

With these modifications, the hedging strategy derived from $\overline{N}_\theta$ outperforms the Black-Scholes hedging strategy, as evidenced by the P&L distributions in Figure 2. Note that perfect hedging is unattainable due to both the incomplete market structure and the discrete hedging with a daily time step (consistent with m intervals over $[0, T]$).

Acknowledgments

Nicolas Baradel acknowledges the financial support provided by the *Fondation Natixis*.

References

[1] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[2] Igor Halperin. Qlbs: Q-learner in the black-scholes(-merton) worlds. *The Journal of Derivatives*, 28(1):99–122, 2020.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[4] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *NIPS 2017 Autodiff Workshop*, 2017. Presented at the NIPS 2017 Workshop on Automatic Differentiation.