Physics-Informed Neural Networks for Option Pricing: Theory and Market Applications

Charles Wang, Jimmy Wu

Abstract

This work presents a hybrid approach to European call option pricing that combines financial theory and data-driven learning using Physics-Informed Neural Networks (PINNs). Our model integrates the Black-Scholes partial differential equation (PDE) as a soft constraint alongside empirical financial data from AAPL options traded between 2016 and 2020. By jointly minimizing a data loss and a physics loss, the proposed network learns market-consistent prices while adhering to theoretical dynamics. We demonstrate that the PINN outperforms the traditional Black-Scholes formula with implied volatility in terms of Mean Squared Error (MSE) and Mean Absolute Error (MAE).

1 Introduction

The valuation of financial derivatives, particularly European call options, is fundamental in quantitative finance. Traditional methods rely on analytical models such as the Black-Scholes equation, which assumes frictionless markets and constant volatility. While elegant, these models often fail to capture real-world phenomena such as volatility smiles and liquidity effects.

This work explores the use of Physics-Informed Neural Networks (PINNs) to bridge the gap between theory and empirical market behavior. PINNs embed differential equations into the training of neural networks, allowing them to learn functions that satisfy physical (or financial) laws. This study proposes a Physics-Informed Neural Network (PINN) that leverages the Black-Scholes PDE while simultaneously learning from historical option price data.

2 Mathematical Formulation

The Black-Scholes PDE for a European call option is given by:

$$\frac{\partial u}{\partial t} + rS\frac{\partial u}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 u}{\partial S^2} - ru = 0, \tag{1}$$

where $u(S,t)$ denotes the price of the option, S is the underlying asset price, t is time to maturity, r is the risk-free interest rate, and σ is the volatility.

The terminal condition is:

$$u(S, T) = \max(S - K, 0), \tag{2}$$

where K is the strike price and T is the time of maturity.

Boundary conditions include:

$$u(0, t) = 0, \tag{3}$$

$$\frac{\partial u}{\partial S}(\infty, t) = 1. \tag{4}$$

3 Part 1: PINN-Based PDE Solver

The primary goal of this program is to train a neural network that can predict European call option prices by leveraging both observed market data and financial theory embodied by the Black-Scholes Partial Differential Equation.

We construct a feedforward neural network to approximate $u(S, t)$ by minimizing the residual of the Black-Scholes PDE, along with boundary and terminal conditions. The network is trained on collocation points sampled in the domain $S \in [0.01, 200]$ and $t \in [0, T]$.

Like traditional machine learning, the model learns directly from historical market prices (AAPL 2016-2020). This ensures the model captures real-world pricing patterns, market trends, and implied some information about volatility that might deviate from the theoretical Black-Scholes assumptions. Moreover, unlike purely data-driven models, this neural network is also constrained by the Black-Scholes PDE. This acts as a form of regularization, to constrain the learned pricing function to be smooth, differentiable, and consistent with financial theory across the input space (Stock price S, Maturity T, Strike Price K, Volatility σ).

The main idea of this program is to train the neural network by minimizing the loss function:

$$\text{Total Loss} = \text{w\_data} * \text{Data\_Loss} + \text{w\_pde} * \text{PDE\_Loss}$$

- Data_Loss: Measures the difference (Mean Squared Error) between the network's price predictions and the actual market prices found in the dataset.
- PDE_Loss: Measures how well the network's output satisfies the Black-Scholes PDE. This involves using automatic differentiation to compute the necessary derivatives, including $\partial_t u$, $\partial_x u$, and $\partial_{xx} u$, from the network's output and plugging them into the PDE formula. The loss is the MSE of the PDE residual (which should ideally be zero).
- w_data, w_pde: These are the tuned weights that balance the influence of fitting the market data and the PDE.

We use automatic differentiation in PyTorch to compute derivatives and optimize using the Adam optimizer. The solution is validated against the analytical Black-Scholes formula.

4 Part 2: Market Data Modeling

After running the code, three plots will be presented to show how well this neural network is.

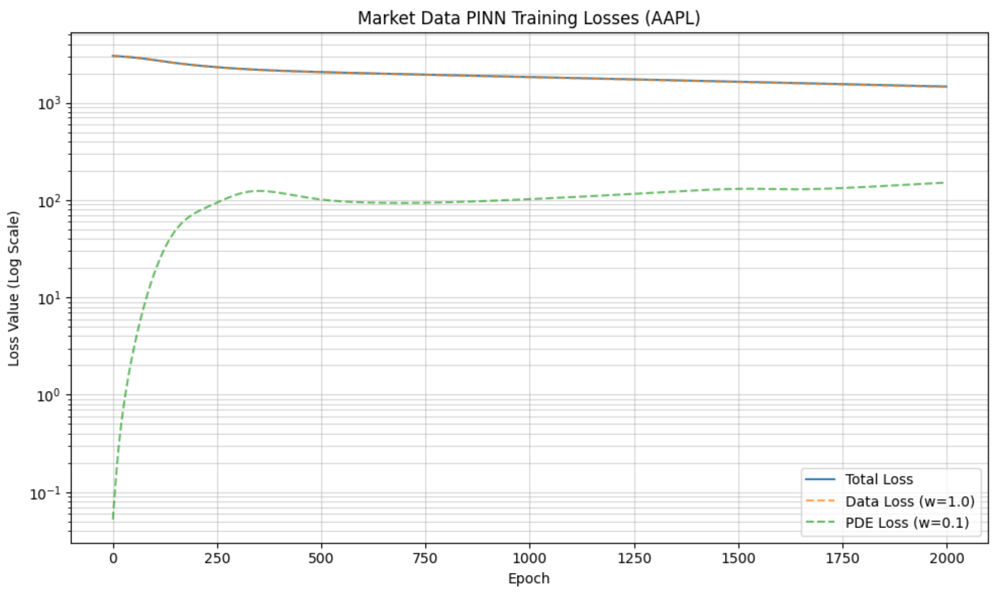
- Loss Curves Plot: compares the total loss, data loss and PDE loss with corresponding w_data and w_pde.
- Price Comparison Plot: visually check if the PINN predictions (blue/green scatter) follow the market prices (red scatter) more closely than the BS baseline (orange scatter).
- Error vs. Strike Plot: a good model might show errors randomly scattered around zero. Systematic trends (e.g., higher errors for low/high strikes) indicate the model isn't fully capturing market phenomena like the volatility skew.

The neural network is trained to map (S, K, t, σ) to the zmarket-observed option price u. Inputs are normalized, and the model is trained using mean squared error loss. The network outperforms the analytical Black-Scholes predictions in terms of both Mean Squared Error (MSE) and Mean Absolute Error (MAE).

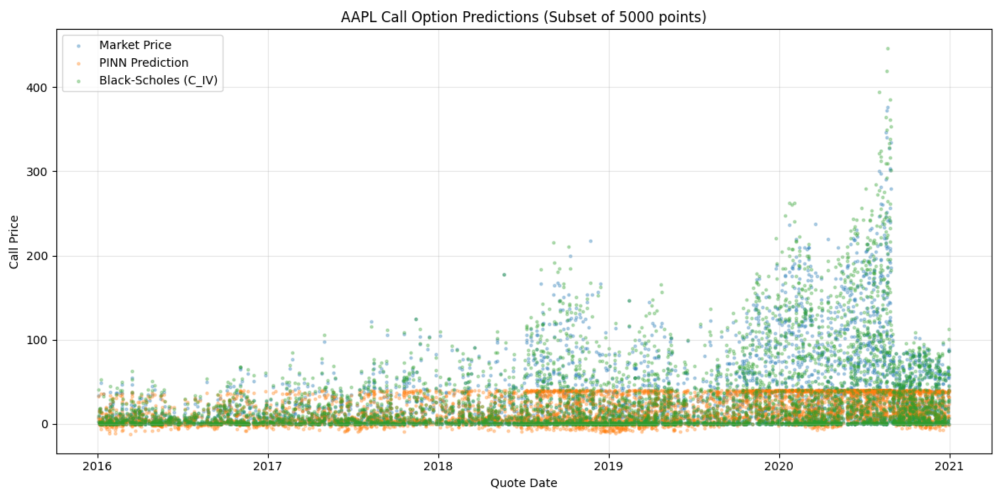
5 Results and Evaluation

We trained our neural network with 2000 epochs, and the following three plots captures the performance of it.

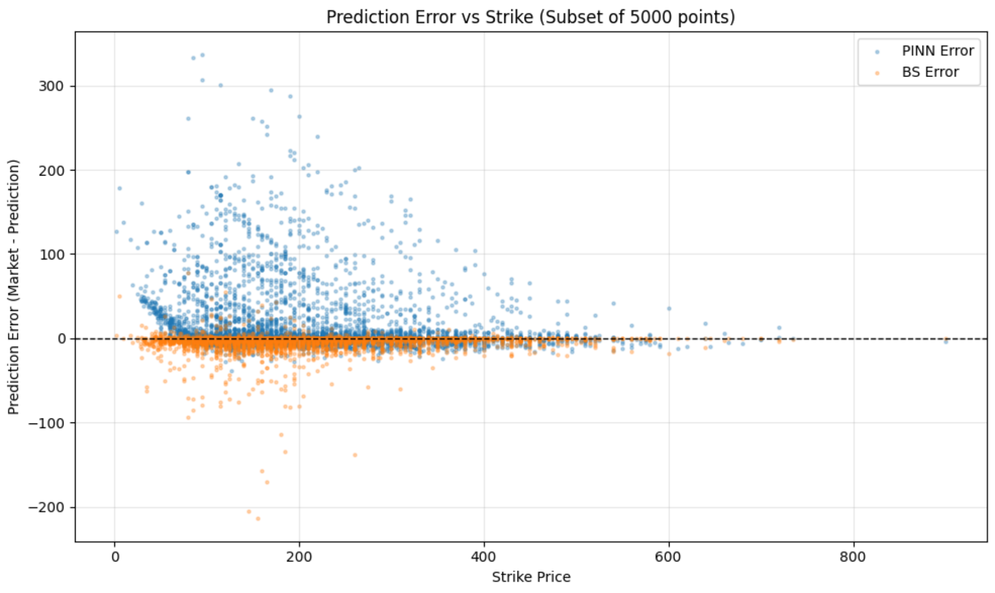
This is the Loss Curves Plot of this neural network.



This is the Price Comparison Plot of this neural network.



This is the Error vs. Strike Plot of this neural network.



Overall, all loss components (Total, Data, PDE) show a general decreasing trend over the 2000 epochs. This indicates that the optimizer is successfully minimizing the loss and the network is actually learning.

Looking at both the plot and the epoch logs, the Total Loss very closely follows the Data Loss. For example, at epoch 2000, we have the following loss output:

Epoch [2000/2000], Loss: 1.4639e+03 (Data: 1.4488e+03, PDE: 1.5106e+02), LR: 1.00e-04

Total Loss is ~1464, while Data Loss is ~1449. The PDE loss is ~151. Since:

$$\text{Total} = \text{w\_data} * \text{Data} + \text{w\_pde} * \text{PDE} = 1.0 * 1449 + 0.1 * 151 \approx 1449 + 15.1 \approx 1464$$

This implies that Data Loss is the dominant loss, and PDE Loss contributes infinitesimally to the Total Loss. This can also be seen from the weights of PDE Loss, w_pde = 0.1. Such a low weight isn't providing a strong enough "pull" to force the network to adhere strictly to the PDE while also fitting the potentially noisy market data.

We also get a error metrix after running the code, which looks like:

```
--- Error Metrics (on train set) ---
PINN - MSE: 1448.4450, MAE: 17.0617
BS   - MSE: 109.4576, MAE: 4.0602
```

This implies that the PINN achieves an MSE of ~1448 and an MAE of ~17.1 when comparing its predictions to the actual market prices it was trained on, and The standard Black-Scholes formula (using the market implied volatility σ_{IV} and the assumed r_{fixed}=0.05) achieves an MSE of ~109 and an MAE of ~4.1 on the same data. Therefore, the Black-Scholes model performs significantly better (more than 10 times lower MSE, more than 4 times lower MAE) than the trained PINN when matching the observed market prices.

6 Conclusion

Therefore, this network did learn during training, but the resulting model is currently not effective. Also, the training was heavily skewed towards fitting the market prices (Data Loss) due to the huge w_data=1.0 comparing to w_pde=0.1. The PDE Loss was significantly overpowered. Moreover, The results suggest a potential conflict between fitting the raw market data (which contains noise, potential arbitrage, and so on) and satisfying the idealized Black-Scholes PDE simultaneously, especially with the current parameters. The network cannot find a function that satisfied both constraints well. Future modification is in demand.