Project Proposal: Machine Learning for Quantitative Finance

Yiqi DENG[1]

yqdeng@connect.hku.hk

Abstract – The objective of this project is to apply machine learning techniques to solve problems in quantitative finance, specifically focusing on replicating the results presented in the paper by Lu et al. (2023) and applying their methods to option pricing applications. The implementation is conducted through TensorFlow. This proposal outlines our approach, including a literature review, method selection, preliminary results, testing plans, and answers to specific questions regarding the replication.

1 Background

Quantitative finance aims to analyze and optimize financial markets and instruments by developing mathematical models and computational techniques, with wide applications of option pricing (Merton, 1973; Cox et al., 1979), risk management (Covello & Mumpower, 1985; Aven, 2016), asset allocation (Brennan et al., 1997), algorithmic trading (Hendershott & Riordan, 2013; Treleaven et al., 2013), etc. As one of the classic problems in quantitative finance, option pricing (Liu et al., 2019; Horvath et al., 2021; Dempster et al., 2022) has spurred ongoing discussions and developments by many researchers. Options are financial instruments that provide the holder with the right, but not the obligation, to buy (call option) or sell (put option) an underlying asset at a predetermined price, i.e., strike price, before or at the expiration date. The payoff of the option is formulated as the difference between the strike price and a value that is dependent on the underlying asset's market price. With the goal of calculating the theoretical value of an option, option pricing facilitates market participants in enhancing their risk management, optimizing investment portfolios, and making informed investment decisions.

Two representative forms of option contacts among various options are European and American options. American options offer the flexibility to be exercised at any time before or on the expiration date, whereas European options can only be exercised at the expiration date. Both European and American options are categorized as path-independent, meaning their payoff is solely determined by the underlying asset's price at a specific moment, i.e., the expiration date for European options or the exercise date for American options. In contrast, there is another type of option known as the path-dependent option. These options have payoffs that are influenced by the underlying asset's entire historical price path, taking into account the asset's cumulative movement over time.

Compared to path-independent options, path-dependent options provide more practical meanings for investors and financial institutions with two advantages. First, path-dependent options provide effective risk hedging. Asian options, a subset of path-dependent options, make use of the underlying asset's average price over a predetermined time. The averaging helps smooth out rapid changes in the price of the underlying asset. Hence, when faced with extreme events or attempts to manipulate prices, this averaging feature can effectively reduce immediate responsiveness and mitigate its adverse effects on option payoffs, enhancing robustness and reducing variances compared with the path-independent counterparts. Second, path-dependent options facilitate flexible risk management. Another type of path-dependent option, Barrier option, is characterized by a predefined threshold. The payoff of barrier options depends on whether the price of the underlying asset reaches or exceeds a predetermined barrier level within the maturity date. By setting acceptable risk thresholds aligned with their preferences, these options provide investors with flexibility in managing their risk exposure.

2 Literature Review

Many financial models have been developed in option pricing, the most widely used model for option pricing is the Black-Scholes model (Black & Scholes, 1973). Under the assumption of constant volatility, the Black-Scholes model provides a closed-form solution to calculate the theoretical price of options. However, the dynamics of assets with variable volatility may not be adequately captured by the assumption of constant volatility. To alleviate the strict constraints of Black-Scholes model, local volatility models (Dupire et al., 1994; Berestycki et al., 2002) adjust volatility based on factors like price level and time, while not incorporating randomness. Stochastic volatility models (Heston, 1993), on the other hand, introduce randomness into the volatility process. In addition, jump-diffusion models (Runggaldier, 2003) that deal with large price jumps or occasional extreme price movements are also introduced as an extension of Black-Scholes model.

Closed-form solutions may not always be available for com-

plex options, such as path-dependent options, as the payoff distribution is unknown. In these cases, numerical methods are introduced and widely used to value these options. Different numerical methods have been developed to solve the option pricing partial differential equation (PDE) or partial integro-differential equation (PIDE) problems. Traditional methods such as the finite difference (Smith, 1985; Thomas, 2013) and finite element method (Johnson, 2012; Dziuk & Elliott, 2013) are proposed for PDE or PIDE in low-dimensional space. However, when faced with high-dimensional cases, the computational cost can increase exponentially with the dimensionality of the equation, which is called the curse of dimension. To mitigate the curse of dimensionality, several machine learning-based approaches have recently been developed. Among them, neural networks (NN) stand out as a promising tool and are widely adopted in current literature.

NN methods such as physics-informed neural networks (PINN) (Raissi et al., 2019; Yu et al., 2022) and deep galerkin method (DGM) (Sirignano & Spiliopoulos, 2018) directly train an NN function on the original PIDE for its solutions. Additionally, NN techniques such as the forward-backward stochastic neural network (FBSNN) (Raissi, 2018) and Deep BSDE (Han et al., 2017, 2018; Gnoatto et al., 2022) are also proposed as PIDE solutions by training an NN function on forward-backward stochastic differential equations reformulated from the original PIDE. However, Deep BSDE and FBSNN can only update the NN parameter after the completion of an entire trajectory simulation, which may incur high computational costs as the number of trajectories increases. To ameliorate this, reinforcement learning technique is adopted in Zeng et al. (2022) and Lu et al. (2023). Specifically, they utilize temporal difference learning to enable parameter updates without the need to wait to complete the entire trajectory, which further lowers computational costs. However, Zeng et al. (2022) is only for the PDE equation, whereas Lu et al. (2023) is extended to PIDE equations involving jump and lévy processes. The integral operators in PIDE help capture non-local behaviors. Meanwhile, the jump and lévy processes allow to model assets that are subject to large, discontinuous changes.

3 Method Selection

In this project, we aim to implement the results in Lu et al. (2023). Furthermore, we intend to extend their results to path-dependent options such as Asian options, Barrier options, and up-and-out put options. In Lu et al. (2023), they demonstrate a diagram of solving high-dimensional PIDEs with jumps by temporal difference method. Specifically, they introduce a set of Lévy processes and construct a corresponding reinforcement learning model. To simulate the entire process, they use deep neural networks to represent the PIDE solutions and non-local terms of equations. The networks are later trained with a loss function incorporating temporal difference error, termination condition, and non-local term.

Lu et al. (2023) offer a feasible solution for path-dependent options pricing. We notice that another paper, Gnoatto et al. (2022), can also be leveraged to solve this problem. While Lu et al. (2023) use temporal difference learning and update parameters at each time step, the Deep BSDE method, Gnoatto et al. (2022), updates NN parameters only after a trajectory is completed. To summarize, we select Lu et al. (2023) based on the following considerations:

- Financial markets exhibit jumps due to unexpected events, news releases, or other factors. Options are often influenced by sudden, discontinuous movements in the underlying asset's price. PIDEs with jumps help capture this realistic market behavior and these sudden events, which may further produce more accurate valuations.
- Path-dependent options including Asian and Barrier options, often involve complex features like barriers, and averaging mechanisms that depend on historical information. We want to capture these complex features and model their effects during the pricing process through non-local terms in PIDEs. Their loss function that incorporates termination conditions and the properties of non-local terms is beneficial for this goal.
- Lu et al. (2023) achieves a high-precision approximation of PIDE solutions. Their methods show fast convergence with the error fluctuating within a relatively small range ($O(10^{-4})$). Meanwhile, it reduces computational costs through temporal difference learning, allowing parameter updates without the completion of the entire trajectory. Results in Lu et al. (2023) show great potential in sharp, precise, and rapid option pricing practice.

4 Testing Plans

To ensure the correctness and validity of our implementation, we will follow the testing plans below.

1) Implement PIDE solver in Lu et al. (2023).
   - Accomplish the residual network that matches the architecture of the residual network shown in Figure 1, for example, d+1-dimension input (t, x) and two outputs, residual blocks with residual connections.
   - Perform temporal difference learning algorithm, including reward in Equation (2.17, 2.18), the updates on the value function $u(t, x)$ in Equation (2.19), and the definition of TD_{error} in Equation (2.20, 2.21), and ensure it greatly matches contents in Section 2.3.
   - Replicate four losses, i.e. Loss 4 in Equation (2.24), Loss 2, and Loss 3 based on the previous terminal state and its gradient in Equation (2.22, 2.23). Ensure that the four loss functions reasonably decrease as the number of iterations increases.

2) Replicate results of 'One-Dimensional Pure Jump'.

   - Test whether the solutions of PIDE in Equation (3.2) generate the same results as in Lu et al. (2023).
   - The exact value $Y_0 = 1$ in 'One-dimensional pure jump process'. Check if the final 5000 updates of training are close to Y_0, with a related error smaller than 0.001.

3) Define PIDEs for different options.

   - Match the state update $(t_n,\ x_n^j)$ to $(t_{n+1},\ x_{n+1}^j)$ following Equation (2.9).
   - Correspond the process of jumps simulation to Fig 3.
   - Correctly define and set the target function and parameters for PIDEs of Asian Options, Barrier Options, etc.

4) Obtain results for different options.

5 Replication of Lu (23) in TensorFlow

When implementing Lu et al. (2023), we identified certain ambiguities and errors that might impact the replication process. After thoroughly reviewing the provided equations and results, we discovered the following ambiguities and errors.

5.1 Typos and Errors

1) Figure 1 shows the architecture of the residual network. In Figure 1, the input 'Input $(t, x_1, x_2, .., x_n)$' is not correct, which should be 'Input $(t, x_1, x_2, .., x_d)$'.

2) In Figure 1, the equation of output '$\mathcal{N}_2$' is not correct, which should be '$\mathcal{N}_2 = \int_{\mathbb{R}^d}(u(t, x + G(x, z)) - u(t, x))\nu(\mathrm{d}z)$'.

3) Figure 2 displays the proposed diagram of solving high-dimensional PIDEs in this paper. In Figure 2, the equation of output '$\mathcal{N}_2$' is not correct, which should be '$\mathcal{N}_2 = \int_{\mathbb{R}^d}(u(t, x + G(x, z)) - u(t, x))\nu(\mathrm{d}z)$'.

5.2 Unclear Parts

1) In Equation (2.1) – (2.3) of Page 3, and the sentence 'For any lévy process L_t,...', the definition of 't' is not clear. The meaning of the letter 'ω' is also not clear.

2) In Equation (2.1), definitions of 'card' and 'S-' are not clear.

3) In Section 2.1, the definition of 'T' should be clarified.

4) In the previous literature (Zeng et al., 2022), the original formula of 'One-Step Temporal Difference Learning' is:

$$u\left(s_n\right) \leftarrow u\left(s_n\right) + \alpha\left(R_{s_n s_{n+1}} + u\left(s_{n+1}\right) - u\left(s_n\right)\right)$$

   However, Equation (2.19) shows a different format. This is because the reward in formula (2.17) contains a negative sign, making the formula of 'One-Step Temporal Difference Learning' slightly different from the original one.

5) In the setting of neural networks in Figure 1, the settings for hidden dimensions in each linear layer are unknown.

6) In Figure 1, the definition of 'Residual connection' is ambiguous and confusing.

7) The authors clarify that "The training consists of 400 iterations, with 50 parameter updates in each iteration, resulting in a cumulative total of 20,000 parameter updates." in the first paragraph on page 11. However, in Table 1, there are 20,000 iterations corresponding to the results of 400 iterations in Figure 4 (a), showing an inconsistency that may confuse the readers.

5.3 Explanatory Note

In addition to replicating results, we provide a short note to account for certain unclear points in Sec 5.2 based on our observations.

- The t in lévy process L_t can refer to the time point t within a given time period T.
- For network settings of Figure 1, the dimension of each linear layer should have the same dimension of residual blocks for the 'residual connection'. According to He et al. (2016), this 'connection' should refer to '$F(x) + x$', meaning adding the original inputs x to the output from the previous layer F.
- For the training iterations, we maintain that there consists of 20,000 iterations for a detailed comparison of Table 1 in Lu et al. (2023).

For other difficult points that may be unclear to readers unfamiliar with the subject. Lu et al. (2023) incorporates background knowledge from the fields of mathematics and computer science. It is advantageous for readers to understand PIDEs. As lévy process plays a role as the standard framework for option pricing, it is beneficial to know about Levy motion, and how the PIDE equation will be formulated after incorporating a Lévy process. Additionally, Lu et al. (2023) mentions 'temporal-difference learning', which is a reinforcement learning method with a form of bootstrapping. Readers should acknowledge its main idea. Furthermore, to apply Lu et al. (2023) in Asian options and Barrier options, etc, knowing definitions and characteristics of different types of options is favourable in constructing appropriate PIDE equations for these options.

References

Aven, T. (2016). Risk assessment and risk management: Review of recent advances on their foundation. *European Journal of Operational Research*, *253*(1), 1–13.

Berestycki, H., Florent, I., et al. (2002). Asymptotics and calibration of local volatility models. *Quantitative finance*, *2*(1), 61.

Black, F., & Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, *81*(3), 637–654.

Brennan, M. J., Schwartz, E. S., & Lagnado, R. (1997). Strategic asset allocation. *Journal of Economic dynamics and Control*, *21*(8-9), 1377–1403.

Covello, V. T., & Mumpower, J. (1985). Risk analysis and risk management: an historical perspective. *Risk analysis*, *5*(2), 103–120.

Cox, J. C., Ross, S. A., & Rubinstein, M. (1979). Option pricing: A simplified approach. *Journal of financial Economics*, *7*(3), 229–263.

Dempster, M., Medova, E., & Tang, K. (2022). Long-term spread option valuation and hedging. In *Commodities* (pp. 125–146). Chapman and Hall/CRC.

Dupire, B., et al. (1994). Pricing with a smile. *Risk*, *7*(1), 18–20.

Dziuk, G., & Elliott, C. M. (2013). Finite element methods for surface pdes. *Acta Numerica*, *22*, 289–396.

Gnoatto, A., Patacca, M., & Picarelli, A. (2022). A deep solver for bsdes with jumps. *arXiv preprint arXiv:2211.04349*.

Han, J., Jentzen, A., & E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, *115*(34), 8505–8510.

Han, J., Jentzen, A., et al. (2017). Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in mathematics and statistics*, *5*(4), 349–380.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hendershott, T., & Riordan, R. (2013). Algorithmic trading and the market for liquidity. *Journal of Financial and Quantitative Analysis*, *48*(4), 1001–1024.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, *6*(2), 327–343.

Horvath, B., Muguruza, A., & Tomas, M. (2021). Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quantitative Finance*, *21*(1), 11–27.

Johnson, C. (2012). *Numerical solution of partial differential equations by the finite element method*. Courier Corporation.

Liu, S., Oosterlee, C. W., & Bohte, S. M. (2019). Pricing options and computing implied volatilities using neural networks. *Risks*, *7*(1), 16.

Lu, L., Guo, H., Yang, X., & Zhu, Y. (2023). Temporal difference learning for high-dimensional pides with jumps. *arXiv preprint arXiv:2307.02766*.

Merton, R. C. (1973). Theory of rational option pricing. *The Bell Journal of economics and management science*, 141–183.

Raissi, M. (2018). Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *arXiv e-prints*, arXiv–1804.

Raissi, M., Perdikaris, P., & Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, *378*, 686–707.

Runggaldier, W. J. (2003). Jump-diffusion models. In *Handbook of heavy tailed distributions in finance* (pp. 169–209). Elsevier.

Sirignano, J., & Spiliopoulos, K. (2018). Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, *375*, 1339–1364.

Smith, G. D. (1985). *Numerical solution of partial differential equations: finite difference methods*. Oxford university press.

Thomas, J. W. (2013). *Numerical partial differential equations: finite difference methods* (Vol. 22). Springer Science & Business Media.

Treleaven, P., Galas, M., & Lalchand, V. (2013). Algorithmic trading review. *Communications of the ACM*, *56*(11), 76–85.

Yu, J., Lu, L., Meng, X., & Karniadakis, G. E. (2022). Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, *393*, 114823.

Zeng, S., Cai, Y., & Zou, Q. (2022). Deep neural networks based temporal-difference methods for high-dimensional parabolic partial differential equations. *Journal of Computational Physics*, *468*, 111503.