# Project Proposal for TSRL Interview Question 3

Guoning Yu

Winter 2023

## 1 Introduction

This question is about reading the paper *Temporal Difference Learning for High-Dimensional PIDEs with Jumps, Liwei Lu, Hailong Guo, Xu Yang, Yi Zhu, 2023* [1] and replicate the reinforcement learning algorithm in it.

A *Partial Integro-Differential Equation (PIDE)* is a type of equation that combines features of both partial differential equations (PDEs) and integral equations. It has wide-spread applications in various scenarios, and particularly in finance. In a PIDE, the unknown function is defined over a domain and the equation involves both partial derivatives and integral operators acting on this function. More specifically, a PIDE typically takes the general form

$$\frac{\partial u}{\partial t} + \mathcal{L}u + \mathcal{I}u = f, \tag{1}$$

in which $u = u(t, \mathbf{x})$ is the unknown function with a time variable $t \in [0, T]$ and a spatial variable $\mathbf{x} \in \mathbb{R}^d$, $\mathcal{L}$ denotes a differential operator with respect to the spatial variable $\mathbf{x}$, and $\mathcal{I}$ represents an integral operator of $u$ with respect to $\mathbf{x}$.

*Jumps* refer to sudden, significant changes in the price of an asset. These can be caused by various events such as economic announcements, geopolitical events, or sudden market movements. Unlike the normal price fluctuations assumed in models like Black-Scholes, jumps represent discontinuities, where the asset price changes significantly in a very short period, often unpredictably.

PIDEs are particularly common in financial mathematics for modeling options pricing in markets with jumps or other non-continuous features, as they can capture both the local changes (via the differential part) and the global or aggregate effects (via the integral part), which addresses the limitations of models like Black-Scholes in markets with jumps.

In the context of PIDEs, the integral part, namely, the "non-local" term typically has the general form

$$\mathcal{I}(u)(t, x) = \int_\Omega K(x, y, t)u(y, t)dy,$$

where $\Omega$ is the domain of integration, and $K(x, y, t)$ is a kernel function that specifies how values of $u$ at different points $y$ contribute to the integral at the point $x$. The kernel often encapsulates the nature of the non-local interactions.

However, solving PIDEs is extremely hard in high-dimensional cases given a large number of underlying assets. There are previous work using deep neural networks (NN) to solve PDEs, while the results on solving PIDEs with NN method are not as extensive.

The paper is about solving PIDEs numerically. A group of Lévy-type forward-backward stochastic processes is introduced based on the target PIDE. It gives an algorithm that ... They used a method of Temporal Difference Learning under the Reinforcement Learning framework. The result presented in the paper is ... The significance of their algorithm is ...

## 2 Typos and Errors in Paper

The typos and errors in the paper are listed as follows.

1. In Equation (2.9), on the left side of the equation, the subscription is wrong: $X_{t_n+1}$ should be $X_{t_{n+1}}$.
2. In Fig. 1, the output $\mathcal{N}_2$ should be the integral of $\cdots \nu(dz)$, not $\cdots dz$
3. In conclusion section: ”the future work” should be ”future work”.

## 3 Unclear Parts

The parts that I had hard time understanding are:

1.

## 4 Replication Results

I used TensorFlow to replicate ... The project implementation is in the GitHub Repo.

In my replication, I used the following techniques.

The results of my replication are ...

Comparing with the paper's original results, ....

## 5 Notes for Important Concepts

### 5.1 Temporal Difference Learning

Temporal Difference (TD) Learning is a Reinforcement Learning (RL) method with form of bootstrapping. Bootstrapping is a resampling method used in model selection, assessment and estimation of statistical properties. To do bootstrapping, we draw a sample from your dataset with replacement repeatedly until we create a new sample of the same size (sometimes larger or smaller) as the original dataset. TD Learning is a method for learning the value function, which is a prediction of future rewards, based on the idea of updating estimates based partly on other learned estimates. This is where the bootstrapping aspect comes in.

The core of TD learning is the *TD error*, a difference between estimated values at successive time steps. In its simplest form TD(0), or Temporal Difference Learning with a look-ahead of zero, the TD error for state-value estimation is given by:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t), \tag{2}$$

where $R_{t+1}$ is the reward received at time $t+1$, $\gamma$ is the discount factor, $V(S_t)$ is the current estimate of the state value, and $V(S_{t+1})$ is the estimate of the next state's value.

In TD learning, the value function for a state is updated incrementally after each step. The update is done in the direction of the TD error. For example, the value function update rule in TD(0) is:

$$V(S_t) \leftarrow V(S_t) + \alpha\delta_t, \tag{3}$$

where $\alpha$ is the learning rate.

TD methods do not require waiting until the end of an episode to perform updates, making them more suitable for continuing environments. They can learn online after every step.

## 5.2 Combination of Loss Functions

The total loss of the model in the paper is combined with four loss functions at each time step.

**Loss 1:** TD error – the error from Equation 2.

**Loss 2:** Termination condition error – the cumulative reward process $Y_t$ needs to satisfy $Y_T = g(X_T)$, and so we calculate the error of such termination condition by taking the Mean Square Error (MSE) between $g(x_T^j)$ and the estimation of $Y_T$ which is $\mathcal{N}_1(T, x_T^j)$. Note that since we do not requite the simulation of entire trajectory, the terminal state $X_T$ is taken from the previous iteration in the calculation.

**Loss 3:** Termination condition gradients error – the gradient of $Y_T$ and $g(X_T)$ should also be equal, therefore, the MSE of the gradients of the two estimations in Loss 2 is taken as another loss.

**Loss 4:** Neural network approximation error –

## 5.3 Lévy-type Forward-backward Stochastic Processes

Lévy process is a stochastic process with independent, stationary increments. More specifically, it is a stochastic process $X = \{X(t) : t \geq 0\}$ that satisfies the following properties:

1. **Starts at Zero:** $X(0) = 0$ almost surely;

2. **Independent increments:** For any $0 \leq t_1 < t_2 < \cdots < t_n < \infty$, $X(t_2) - X(t_1), X(t_3) - X(t_2), \ldots, X(t_n) - X(t_{n-1})$ are mutually independent;

3. **Stationary increments:** For any $s < t$, $X(t) - X(s)$ is equal in distribution to $X(t-s)$;

4. **Continuous in probability:** For any $\varepsilon > 0$ and $t \geq 0$, $\lim_{h\to 0} P\left(|X(t+h) - X(t)| > \varepsilon\right) = 0$.

The most well-known Lévy processes are the Brownian motion (Wiener process), which is a process without jumps, and the Poisson process, which can include jumps. A Lévy process $X = \{X(t) : t \geq 0\}$ defined as above is a Brownian motion if for any $t \geq 0$,

$$X(t) \sim \mathrm{N}(\mu, \sigma^2),$$

where $\mu = 0$ and $\sigma = t$. A Lévy process $X = \{X(t) : t \geq 0\}$ defined as above is a Poisson process if for any $t \geq 0$,

$$X(t) \sim \mathrm{Pois}(\Lambda),$$

where $\Lambda = \lambda(t - s)$, and $\lambda$ is called rate or intensity.

The Itô's formula is ...

By applying the Lévy type stochastic integrals and the Itô's formula to equation (2.6), we can derive equation (2.8). (needs to check) The system of (2.6) and (2.8) are referred to as the *forward-backward stochastic differential equation (FBSDE) system*. Equation (2.6) is called the "forward" equation since it describes the evolution of the state variable $X_t$ in a forward manner in time, starting from an initial condition at $t = 0$ and moving towards $t = T$. The dynamics of $X_t$ are influenced by a deterministic term ($b(X_t)$), stochastic term (Brownian motion $dW_t$), and a jump term involving the Lévy process $\int_{\mathbb{R}^d} G(X_t, z)\tilde{N}(dt, dz)$. Equation (2.8) describes the dynamics of the processes $Y_t$, $Z_t$, and $U_t$, which are connected to the value function $u(t, X_t)$ and its derivatives. Unlike the forward equation, it is called a "backward" equation as it starts with a terminal condition at time $T$ (i.e., $Y_T = g(X_T)$) and then moves backward in time. This backward structure is intrinsic to many problems in stochastic control and finance, where the final payoff or condition is known, and the goal is to determine the optimal strategy or value function backward in time.

The processes of $(X_t, Y_t, Z_t, U_t)$ satisfying equation (2.8) provides a foundation for assessing the accuracy of the approximate solution $u(t, x)$ to the PIDE (2.4), which is the key idea of the whole paper. The discrete version stated by equation (2.9) and (2.10) can be applied to simulate for numerical approximation.

## 6 Further Questions

There is an important reference [2] that also uses such approach to solve high-dimensional parabolic PDEs.

## References

[1] L. Lu, H. Guo, X. Yang, and Y. Zhu, Temporal difference learning for high-dimensional pides with jumps, 2023.

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