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* 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. 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 $u = u(\mathbf{x}, t)$ is the unknown function, $\mathcal{L}u$ denotes a differential operator, which is often a linear or nonlinear combination of spatial derivatives of u, and $\mathcal{I}u$ represents an integral operator, involving an integral of u with respect to space, possibly weighted by a kernel function.

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 Ω 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 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 $t+1$, γ 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 α 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