Reinforcement Learning in Limit Order Book for Cryptocurrency Trading

Jean-Sébastien Delineau & Yanruiqi Yang

Abstract

In a model-based stock market environment, we train multiple Reinforcement Learning (RL) agents with PPO and StableBaselines3 to perform in highly volatile environments like Bitcoin cryptocurrency. This project uses `mbt_gym`[2], a module that provides a suite of gym environments for training RL agents. We demonstrate that the agent can have a promising ability to learn even in a mildly stochastic environment.

Background

An order book is a list of orders that a trading exchange uses to record the interests of buyers and sellers for a particular financial instrument. The highest bid (best-bid) and lowest ask (best-ask) are defined as the top of the book.

A limit order book (LOB) is a bookkeeping system that displays all buy and sell limit orders for a given instrument. The LOB is composed of the bid and ask sides. The bid side contains all buy limit orders, arranged from the highest to the lowest price. Conversely, the ask side contains all sell limit orders, arranged from the lowest to the highest price. LOB orders are filled based on price and time priority, meaning the highest bid order and the highest ask order that arrived first are filled first.

A market maker provides liquidity in the stock market by quoting bid and ask prices at which they are willing to buy and sell specific quantities of assets. By doing so, the market maker faces two types of risk: inventory risk, which is inherent to stock price uncertainty, and asymmetric information risk, which arises from dealing with better-informed traders (Bias et al., 2004; Stoll, 2003). This project only considers inventory risks.

Model

We train the agent to learn how to quote a bid price p^b and an ask price p^a around the 'true' price s, which is defined as the mid-price, where the mid-price is the average sum of the best-bid and the best-ask as we can see on the middle lower plot of Figure 1. [2] Has shown that in simple environments, the agent learns the optimal closed analytic solution derived in [1] for a market maker.

Figure 1. Performance $\mathbf{BM}_{\sigma=2}$

The framework of the environment is composed of a stochastic process for the mid-price S_t, Hawkes Process for the arrival process H_t, and an Exponential Fill process for the probability of filling an order λ_t, all embedded in a Limit Order Book Dynamics. The reward function is implemented with the running inventory penalty[2] function, and not the Profit and Loss (PnL), explaining the divergence between the PnL and the cumulative reward on Figure 6.
The state space is of 6 dimensions, the inventory Q_t (number of assets held), the cash C_t (amount of cash held), the time steps, the stock price S_t, the Hawkes Process (the number of orders as a cluster), and the Price Impact Process I_t (Impact execution price by the current trading activity).

$$\mathbf{PnL}_t = S_t \cdot Q_t + C_t - (S_0 \cdot Q_0 + C_0).$$

The action space is a continuous two-dimensional space, $[\delta_t^b, \delta_t^a]$, representing the distances between the mid-price and the bid price, and the mid-price and the ask price.

$$\delta_t^b := s_t - p_t^b \quad \delta_t^a := p_t^a - s_t.$$

Training

Proximal Policy Optimization (PPO) Algorithm

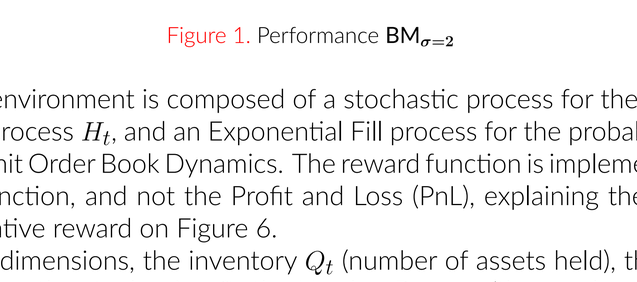


Figure 2. PPO Algorithm

Advantage Estimation: $\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \quad \text{where} \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

Clipping Mechanism: $L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$

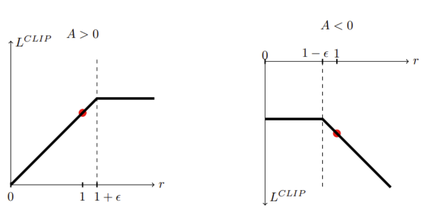


Figure 3. Clipping Mechanism

Numerical Simulation

Performance in the Original Training Environment:

After training, we simulated 1000 trajectories to assess the agent's performance. Figure 1 shows one example, while Figure 6 depicts the average performance over all trajectories. Despite the mean cumulative reward being negative, the PnL remains positive due to the stricter dynamics of the running inventory penalty. The middle plot reveals that both the PnL and end cumulative reward are highly skewed with a fat tail for losses.

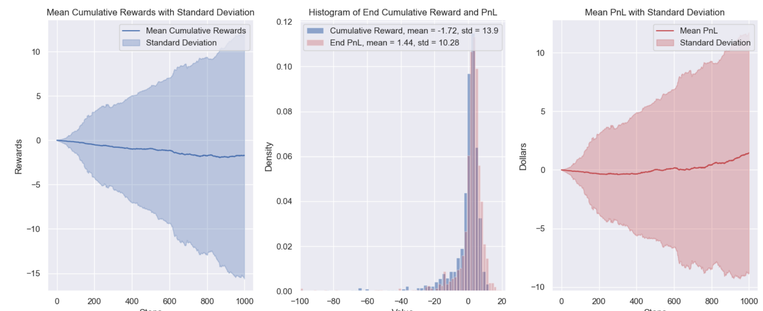


Figure 4. $\mathbf{BM}_{\sigma=0.1}^{J=1}$ Mean performance over 1000 trajectories

Performance in the Backtest Environment:

We can observe a total opposite behavior, $\mathbf{GBM}_{\boldsymbol{\sigma=0.1}}$ shorts quantities of more than 200 in stock, whereas $\mathbf{BM}_{\boldsymbol{\sigma=0.1}}^{\boldsymbol{J=1}}$ never holds or shorts more than three units of assets and has more frequency.

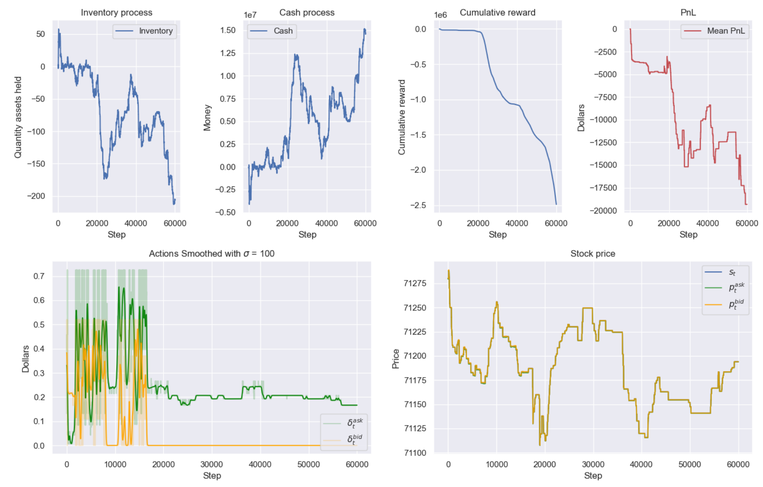


Figure 5. $\mathbf{GBM}_{\boldsymbol{\sigma=0.1}}$ Backtest Performance 10ms

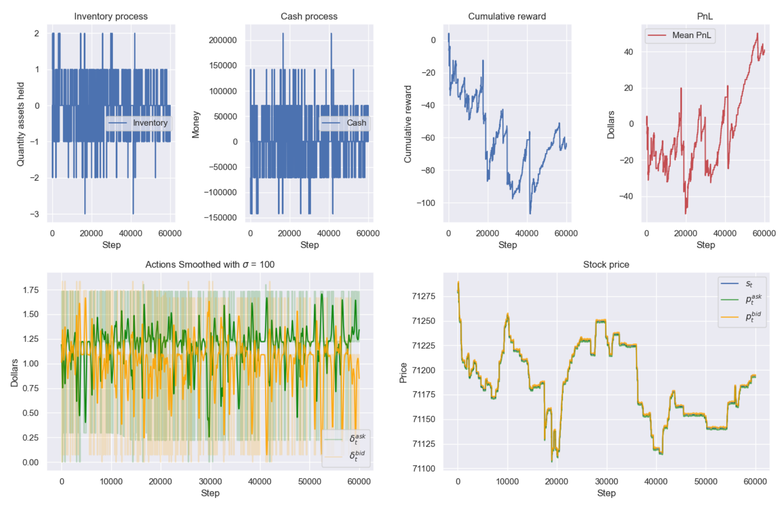
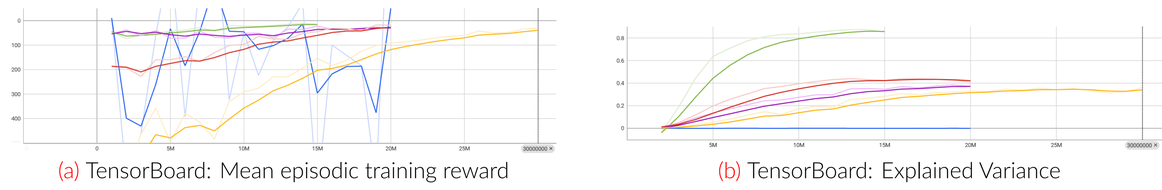


Figure 6. $\mathbf{BM}_{\boldsymbol{\sigma=0.1}}^{\boldsymbol{J=1}}$ Backtest Performance 10ms

Results

	$\mathbf{GBM}_{\sigma=0.1}$	$\mathbf{GBM}_{\sigma=2}$	$\mathbf{BM}_{\sigma=2}$	$\mathbf{BM}_{\sigma=0.1}^{J=1}$	$\mathbf{BM}_{\sigma=1}^{J=4}$
Mean PnL	15.89	145.27	17.73	1.44	-18.29
Sdev PnL	87.54	5102.91	11.89	10.28	13.92
Mean CumReward	-15.39	43.8	5.35	-1.72	-19.2
Sdev CumReward	98.54	5040.97	26.91	13.9	14.57
Mean PnL	4811.78	53555.83	-620.59	673.04	910.79
Sdev PnL	402.81	545.11	211.54	82.97	116.12
Mean CumReward	-736,160.7	X	-12678.93	-958.42	422.95
Sdev CumReward	92715.72	X	3524.97	122.58	118.71
Mean PnL	-17,841.6	-31,394.08	-5780.27	-1201.18	42.24
Sdev PnL	2004.37	392.28	243.25	148.61	47.52
Mean CumReward	-1,927,709.9	X	-31,680.1	-2843.91	-66.85
Sdev CumReward	899,779.09	X	7534.73	316.15	55.74

Figure 7. Performance Agent & Backtest Data: timesteps = 25ms & 10ms



(a) TensorBoard: Mean episodic training reward

(b) TensorBoard: Explained Variance

Figure 8. Purple= $\mathbf{GBM}_{\boldsymbol{\sigma=0.1}}$, Blue = $\mathbf{GBM}_{\boldsymbol{\sigma=2}}$, Green = $\mathbf{BM}_{\boldsymbol{\sigma=2}}$, Red = $\mathbf{BM}_{\boldsymbol{\sigma=0.1}}^{\boldsymbol{J=1}}$, Yellow = $\mathbf{BM}_{\boldsymbol{\sigma=1}}^{\boldsymbol{J=4}}$

We compared the performance of five agents in Figure 7 and plotted their training performance in Figure 8. The results are meaningful and can be improved with more training time but to a lesser extent as the explained variance shows.

Conclusion

We trained multiple baseline agents, finding that jump process-based agents tend to perform better. Despite unconvincing cumulative rewards, they may still perform well. Agents often excel when trained in more stochastic environments. To improve performance, we can implement robust agents like RARL.

References

Marco Avellaneda and Sasha Stoikov. High-frequency trading in a limit order book. *Quantitative Finance*, 8(3):217–224, 2008.

Joseph Jerome et al. Mbt-gym: Reinforcement learning for model-based limit order book trading. 2023. arXiv preprint arXiv:2209.07823.

Jan Sila Michael Mark and Thomas A. Weber. Quantifying endogeneity of cryptocurrency markets. *The European Journal of Finance*, 28(7):784–799, 2022.

Csaba Szepesvári. *Algorithms for reinforcement learning*. Springer nature, 2022.