REINFORCEMENT LEARNING FOR MARKET MAKING: A BID-ASK STRATEGY OPTIMIZATION

https://github.com/sadeghb/market-making-RL-agent

Abstract. This project explores the application of reinforcement learning (RL) to market making, a challenging problem in financial markets. The RL agent dynamically interacts with a simulated trading environment, leveraging Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN) to learn optimal strategies for quoting bid and ask prices. By effectively balancing profitability and risk management, the agent demonstrates an ability to adapt to market conditions, showcasing the transformative potential of RL in automated trading systems.

1 BACKGROUND AND RELATED WORK

Market making refers to the practice of continuously quoting buy (bid) and sell (ask) prices for financial instruments, such as stocks or cryptocurrencies, to facilitate trading. Market makers provide liquidity by ensuring that traders can easily buy or sell assets without significant price changes. Their goal is to profit from the bid-ask spread, the difference between the buy and sell prices. For example, a market maker might quote a bid price of $100 and an ask price of $102 for an asset, earning a $2 profit for each round trip (buy-sell cycle), provided there are no adverse price movements.

At the core of market making is the limit order book (LOB), which organizes buy and sell orders. Limit orders specify a price at which a trader is willing to buy or sell an asset. For instance, a trader might place a limit buy order at $100, meaning they will only purchase the asset if the price drops to $100 or lower. Similarly, market orders execute immediately at the best available price. The LOB aggregates these orders into two queues: bids (buy orders) and asks (sell orders), ranked by price. Market makers interact with the LOB by placing limit orders to quote bid and ask prices.

Traditional approaches, such as the Avellaneda-Stoikov model [1], rely on analytical methods to balance inventory and capture spreads but often struggle with the complexities of dynamic and volatile markets. Reinforcement learning (RL) has emerged as a promising alternative, leveraging its ability to learn optimal policies from data. For instance, Beysolow [2] explored market-making strategies using RL, emphasizing adaptability in dynamic environments and profit potential while managing risk.

Further advancements in RL-based market-making strategies emphasize deep learning for enhanced adaptability. Li et al. proposed the Predictive and Imitative Market Making Agent (PIMMA), which integrates predictive auxiliary signals and imitation learning to address liquidity and inventory risks, demonstrating superior performance in simulations [3]. Sun et al. employed long short-term memory (LSTM) networks to capture temporal patterns in LOBs, achieving notable improvements over traditional models [4]. Shi et al. extended RL frameworks by incorporating inventory hedging under market impact, offering greater flexibility compared to conventional methods [5]. Together, these approaches underscore RL's potential for developing adaptive and robust market-making algorithms. Finally, we highlight that our work is inspired by [6], in that it explicitly deals with the LOB. However, we develop our own environment, which is substantially different from theirs in terms of both structure and implementation.

2 Problem Definition and Assumptions

The objective is to implement an RL agent for market making. The agent interacts with a simulated market, observing market conditions (details in the methodology section), and quotes buy (bid) and sell (ask) prices. It also executes trades based on these prices. The agent's goal is to profit from the bid-ask spread by buying low and selling high while managing risks such as adverse price movements and inventory imbalances. Additionally, it must remain competitive in a dynamic market by adapting its strategy in real time.

Due to the complexity of real-world financial markets, simplifying assumptions are made to make the problem tractable. These assumptions include: the framework is focused on the single-asset case, although it is extendable to the multi-asset case; the agent assumes fixed trade volumes for simplicity; the market dynamics are modeled using historical data and do not account for the influence of the agent's actions on the market; transaction costs such as fees and slippage are not considered; the market is assumed to operate continuously with no interruptions; and the agent has complete and accurate access to all relevant market information at each timestep.

3 Methodology

3.1 Environment Design

The simulation environment for the RL agent is designed to emulate real-world market-making scenarios. The environment is built around a limit order book (LOB) framework, which captures the dynamics of financial markets at a granular level. The data for the LOB is sourced from historical cryptocurrency market data, specifically second-level snapshots of Bitcoin trading activity. This dataset includes the midpoint price, spread, bid volumes, and ask volumes, allowing for a realistic representation of market conditions.

The observation space for the RL agent comprises (i) observable market features at each timestep, such as the midpoint price, spread, and volumes on both sides of the book, as well as (ii) the agent's cash balance and current inventory level and a rolling average of the bitcoin price. This information provides the agent with a comprehensive view of the market dynamics. The action space is defined as the agent's bid and ask prices, which it quotes at each timestep. To simplify the problem, we assume constant trade volumes for the agent's orders.

The reward function is defined in two variants, each addressing different aspects of profitability and risk management. The first reward function simplifies the objective by rewarding bids that are below and asks that are above the rolling average of bitcoin price. It also rewards transactions, so to provide liquidity to other traders. While this formulation omits explicit risk controls, it emphasizes realized profitability from executed trades. The second reward function focuses on the change in the agent's total wealth, which is the sum of its cash and the mark-to-market value of its inventory. This formulation includes a quadratic penalty on large inventory positions to mitigate risk by discouraging excessive exposure to price fluctuations.

3.2 Algorithm Selection

For this project, we select two RL algorithms: Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN). These algorithms are chosen for their adaptability to market-making, scalability of training, and ability to learn optimal strategies under dynamic market conditions. Modifications, such as custom reward

shaping and action discretization, are applied to tailor them to the unique requirements of this problem.

Proximal Policy Optimization (PPO): PPO is a popular policy-gradient algorithm known for its stability and sample efficiency. While traditionally designed for continuous action spaces, we discretize the actions for this project to improve performance in the market-making context. This approach allows for structured and efficient quoting of bid and ask prices. PPO's clipped objective function ensures that policy updates remain stable, reducing the risk of catastrophic performance drops during training.

Deep Q-Networks (DQN): DQN is a value-based algorithm that approximates the action-value function using deep neural networks. While traditionally designed for discrete action spaces, DQN can be adapted for market-making by discretizing the bid and ask price ranges. It provides an alternative reinforcement learning approach for addressing market-making tasks alongside PPO.

3.3 Implementation Details

We utilize the implementation of PPO and DQN from the `stable_baselines3` library. To address convergence issues observed with continuous action spaces, we discretize the action space for both PPO and DQN. The action space is defined as:

$$\{(a, b) \mid a, b \in \{0, 1, ..., 10\}\},$$

where a and b represent discrete adjustments to the bid and ask prices, respectively.

The observations provided to the agent include the midpoint and spread of the market at the current time, bid and ask price levels with their respective volumes, as well as the agent's current cash balance and inventory. The agent's bid and ask prices are computed as:

$$(\text{midpoint} - \text{spread} * \text{a} \,/\, 10,\ \text{midpoint} + \text{spread} * \text{b} \,/\, 10).$$

This design ensures that the agent's quoting strategy is tied to market dynamics, enabling consistent price adjustments.

To reduce computational overhead, we focus on a single-asset case in this project. For performance evaluation, we compare our RL agents against two baselines. The first is a naïve strategy, which quotes:

$$(\text{midpoint} - \text{spread} \,/\, 2,\ \text{midpoint} + \text{spread} \,/\, 2).$$

The second baseline is the well-known Avellaneda-Stoikov model, slightly modified to disregard market closure constraints, aligning it with our experimental setup.

Experiments are conducted using a dataset containing historical secondly LOB data for Bitcoin. The main performance metric is the agent's wealth over time, defined as the sum of cash and inventory value (converted to cash using the current midpoint). This metric captures both realized profits and unrealized gains, reflecting the agent's ability to balance profitability and risk management effectively.

4 Results and Analysis

The performance of the agents is evaluated using three key metrics: cash balance, inventory, and wealth, all tracked over time. These metrics collectively capture the agents' behaviors and performances. The analysis focuses on PPO with both reward functions (Reward 1: rolling average-based with transaction reward, Reward 2: wealth-based with inventory penalty), as well as the naïve agent and the Avellaneda-Stoikov (A-S) model. Since the naïve agent and A-S are independent of the reward function, their behaviors remain consistent across comparisons. While the accompanying notebook includes the analysis of

DQN agents, they are omitted here due to space constraints and because their behavior and performance were largely similar to those of the PPO agents. The results are shown in Figure 1, where `a-s` refers to the A-S model, while `simple` represent the naïve agent.

Cash Balance: The naïve agent exhibits highly erratic behavior, with its cash balance frequently oscillating between extreme highs and lows. This suggests an overly enthusiastic trading strategy that does not manage liquidity effectively. In contrast, the other three agents–PPO1, PPO2, and A-S– demonstrate much more stable cash balance trajectories, reflecting controlled and measured trading strategies.

Inventory: Examining inventory evolution reveals distinct behaviors among the agents. PPO2 emerges as the most conservative, maintaining low inventory levels to minimize risk. This aligns with its reward function, which explicitly penalizes large inventory positions. On the other hand, A-S appears to be inventory-hungry, consistently holding high inventory levels. This behavior reflects its formulation, which prioritizes profitability and can be adjusted by altering the tradeoff between risk and profit. PPO1 strikes a balanced approach, neither excessively conservative nor overly aggressive, showcasing a moderate and adaptable inventory management strategy.

Wealth: In terms of wealth evolution, both PPO1 and A-S demonstrate profitability, with performances that are comparable. This highlights PPO1's ability to compete with a theoretically robust model like A-S. Conversely, PPO2 and the naïve agent fail to achieve profitability, remaining under water throughout the evaluation period. PPO2's conservative strategy, while risk-averse, limits its ability to capitalize on opportunities, and the naïve agent's erratic behavior undermines its performance.

Overall Assessment: Overall, PPO1, with its simplified reward function, achieves a commendable balance between profitability and risk management. Its moderate inventory levels and stable cash balance translate to competitive performance against the A-S model. Further tuning of the PPO1 algorithm and hyperparameters could potentially enhance its profitability, providing a strong foundation for RL-based market-making strategies.

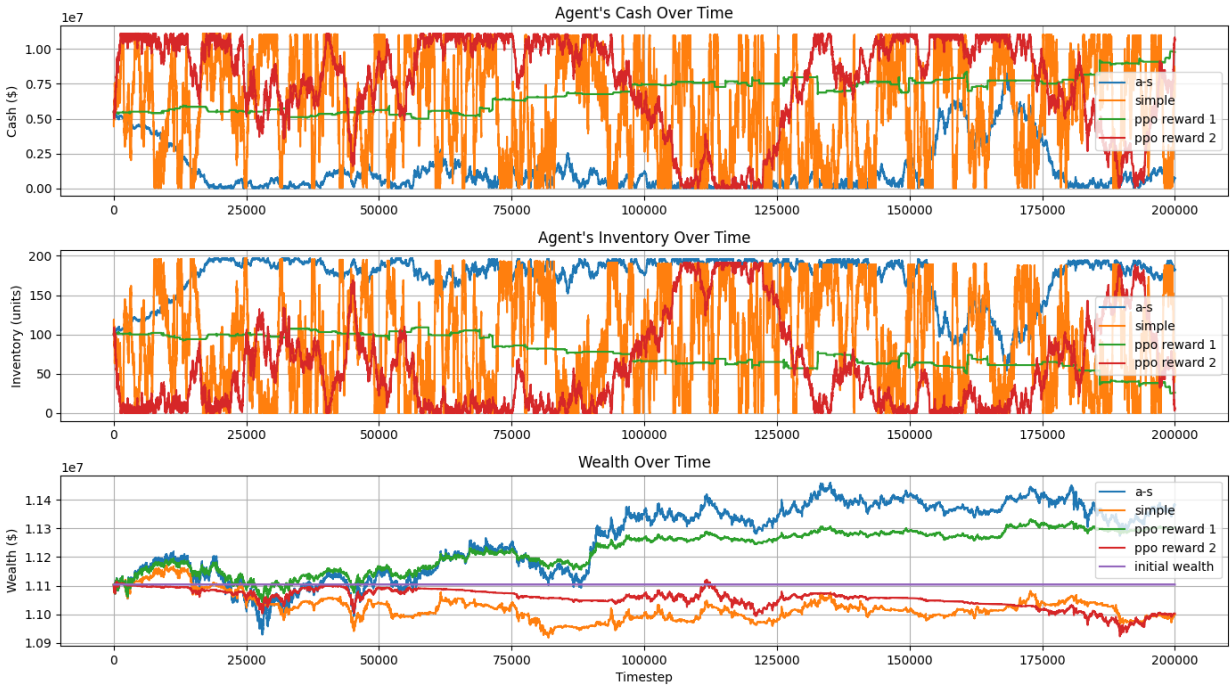

Figure 1: Comparison of Cash Balance, Inventory, and Wealth Over Time Across Agents

5 DISCUSSION

This project aimed to develop an RL agent for market making, with the goal of balancing profitability, risk management, and market liquidity. Despite progress made, several challenges and limitations emerged during the development and implementation process.

A major challenge in this project stemmed from the nature of the data, which only provided the LOB and lacked direct information about executed transactions. Additionally, the data did not include asset value explicitly, requiring us to use the midpoint price as a proxy. To determine whether a bid or ask order was matched, we devised a solution using the next timestep: if the agent's bid was higher than the best bid at the subsequent step (i.e., the highest unmatched bid), it was considered matched. This approach allowed us to infer transaction outcomes while working within the limitations of the available data.

High volatility in asset value posed another significant challenge for market making. Rapid price fluctuations increased the risk of inventory losses and made it more difficult to set competitive bid and ask prices. Addressing this issue would likely require further hyperparameter tuning and extended training times to enable the agent to adapt more effectively to volatile market conditions.

However, the current approach has several limitations. First, the use of the midpoint as a proxy for asset value overlooks potential bid-ask spread dynamics, which may lead to suboptimal quoting strategies. Second, the simplified treatment of the market assumes that agent actions do not influence market dynamics, ignoring feedback effects that occur in real-world scenarios. Finally, transaction costs, such as fees and slippage, are not accounted for, which can impact profitability in practical implementations.

6 CONCLUDING REMARKS AND FUTURE WORK

This project explored reinforcement learning (RL) approaches to market making by implementing Proximal Policy Optimization (PPO) and Deep Q-Networks (DQN) with two distinct reward functions. As baselines, we compared our RL models against the Avellaneda-Stoikov model and a simplified agent inspired by it.

Our findings indicate that PPO and DQN consistently outperformed the simple agent across both reward functions. However, when compared to the Avellaneda-Stoikov model, the results varied. The first reward function, which measures the change in the agent's cash balance, led both models to become profitable and have comparable performances to that of the Avellaneda-Stoikov baseline. In contrast, the second reward function, which accounts for the change in the agent's total wealth (cash balance + inventory value) and includes a risk-controlling inventory penalty term, did not enable either PPO or DQN to achieve profitability. The results achieved by the environment driven by the first reward function showcase the potential of RL to match theoretical benchmarks under appropriate conditions.

For future work, we propose extending this framework to the multi-asset market-making scenario, which presents additional complexities and opportunities for optimization. Incorporating advanced considerations, such as risk sensitivity, inventory hedging, and strategies for low-liquidity environments or sparse data, and generalizing the action space to include bid and ask volumes, could further enhance the robustness and applicability of RL-based market-making strategies. These directions hold promise for bridging the gap between theoretical models and practical implementations in dynamic financial markets.

REFERENCES

[1] M. Avellaneda and S. Stoikov, "High-frequency trading in a limit order book," *Quantitative Finance*, vol. 8, no. 3, pp. 217–224, 2008.

[2] T. Beysolow II, "Market making via reinforcement learning," *Applied Reinforcement Learning with Python: With OpenAI Gym, Tensorflow, and Keras*, pp. 77–94, 2019.

[3] S. Li, Y. Chen, H. Niu, J. Zheng, Z. Lin, J. Li, J. Guo, and Z. Wang, "Toward automatic market making: An imitative reinforcement learning approach with predictive representation learning," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024.

[4] T. Sun, D. Huang, and J. Yu, "Market making strategy optimization via deep reinforcement learning," *IEEE Access*, vol. 10, pp. 9085–9093, 2022.

[5] J. Shi, S. H. Tang, and C. Zhou, "Market-making and hedging with market impact using deep reinforcement learning," in *Proceedings of the 5th ACM International Conference on AI in Finance*, pp. 652–659, 2024.

[6] H. Guo, J. Lin, and F. Huang, "Market making with deep reinforcement learning from limit order books," in *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2023.