24B3019

FINAL REPORT for Citadel Build the Algorithmic Black Box

Abstract

This project studies the behavior of a reinforcement learning (RL) agent operating in a simulated financial market with transaction costs and inventory risk. A discrete-action trading agent was trained using Proximal Policy Optimization (PPO) within a custom Markov Decision Process (MDP) that captures price dynamics, inventory constraints, and risk-adjusted rewards. In addition, a simplified multi-agent market simulation was constructed to analyze emergent market properties such as volatility clustering and herding behavior.

Empirical results show that the RL agent learns a cautious trading strategy that adapts to market volatility by reducing position sizes during high-risk regimes and increasing activity during stable periods. Performance evaluation across low- and high-volatility environments demonstrates improved risk-adjusted returns compared to random and naive benchmark strategies, at the cost of reduced trading frequency under extreme volatility. The findings highlight both the potential and limitations of reinforcement learning for algorithmic trading in stylized market environments.

1. Methodology

1.1 Market and Order Book Design

The market environment is designed as a simplified price-driven trading system rather than a full limit order book. Asset prices evolve exogenously according to a stochastic process, capturing short-term randomness while remaining computationally efficient. This abstraction allows the agent to focus on decision-making under uncertainty rather than micro-level order matching mechanics.

To study market-wide effects, a separate multi-agent simulation was constructed in which multiple heterogeneous agents generate buy and sell orders over time. The resulting order flow produces stylized market behavior such as clustered volatility and correlated trading activity, enabling higher-level analysis of collective dynamics.

Key design choices:

- Prices evolve continuously over time with stochastic noise.
- Transaction costs penalize excessive trading.
- Inventory limits constrain leverage and prevent degenerate solutions.
- Market traces are recorded to analyze aggregate behavior.

This design balances realism with tractability and aligns with common practices in reinforcement learning–based market research.

1.2 Reinforcement Learning Agent Setup

State Space

At each timestep, the RL agent observes a normalized state vector consisting of:

- Current asset price (normalized by initial price),
- Current inventory position (scaled by inventory limits),
- Available cash balance (normalized),
- Time progress within the episode.

This state representation provides sufficient information for the agent to manage risk, control exposure, and adapt trading behavior over time.

Action Space

The agent operates in a discrete action space:

- **Hold** (no trade),
- **Buy** (increase inventory by one unit),
- **Sell** (decrease inventory by one unit).

Discrete actions simplify policy learning while preserving essential trading behavior.

Reward Function

The reward function is explicitly risk-aware and defined as:

[
r_t = \Delta \text{Portfolio Value}_t - \lambda \cdot \text{Drawdown}_t
]

where:

- Portfolio value includes both cash and marked-to-market inventory,
- Drawdown penalizes deviations from the historical peak portfolio value,

- (\lambda) controls risk sensitivity.

This formulation encourages profitability while discouraging excessive risk-taking and large drawdowns, aligning the agent's objective with realistic trading goals.

Learning Algorithm

The agent is trained using **Proximal Policy Optimization (PPO)**, chosen for its stability, robustness to noisy rewards, and widespread use in continuous control problems. PPO's clipped objective ensures controlled policy updates, which is particularly important in non-stationary and stochastic environments such as financial markets.

---

2. Experiments

2.1 Experimental Setup

The agent was trained and evaluated under different market regimes:

- **Low-volatility environments**, characterized by smaller price fluctuations.
- **High-volatility environments**, with larger and more frequent price movements.

Each experiment was run for multiple episodes to reduce variance, and performance was compared against simple baselines such as random trading and passive holding.

2.2 Performance Across Volatility Regimes

Low Volatility

In low-volatility environments, the agent exhibits:

- Higher trading frequency,
- Gradual accumulation and liquidation of positions,

- Stable growth in portfolio value.

The agent effectively exploits small price movements while keeping inventory within moderate bounds, resulting in consistent positive rewards.

High Volatility

Under high volatility:

- The agent significantly reduces trading activity,
- Inventory exposure is kept close to zero,
- Drawdowns are minimized at the cost of reduced profit opportunities.

This behavior reflects learned risk aversion rather than random inactivity, indicating that the reward structure successfully shapes conservative decision-making during unstable market conditions.

2.3 Key Metrics

The following metrics were used to evaluate performance:

- **Cumulative Reward**: Measures overall objective optimization.
- **Profit and Loss (PnL)**: Tracks economic performance over time.
- **Sharpe Ratio**: Assesses risk-adjusted returns.
- **Maximum Drawdown**: Quantifies downside risk.
- **Inventory Dynamics**: Analyzes position sizing and exposure control.

Across experiments, the RL agent consistently outperformed random baselines in reward and drawdown metrics, though absolute PnL remained sensitive to volatility and transaction costs.

---

3. Conclusion

3.1 Learned Trading Behavior

The agent learned several realistic trading behaviors:

- Dynamic risk control through inventory management,
- Reduced exposure during volatile periods,
- Preference for incremental gains over aggressive speculation,
- Implicit market timing based on reward feedback rather than explicit prediction.

These behaviors emerge endogenously from the reward design rather than being hard-coded.