

Reinforcement Learning for Financial Trading

Quant Finance Portfolio

December 25, 2025

Abstract

This document outlines the mathematical foundations of Reinforcement Learning (RL) as applied to the trading project (Project 12). We discuss the Markov Decision Process (MDP) formulation, the Bellman equation, and Q-Learning.

1 Markov Decision Process (MDP)

The trading problem is modeled as an MDP tuple (S, A, P, R, γ) :

- S : State space (e.g., market trend, current holdings).
- A : Action space (Buy, Sell, Hold).
- $P(s'|s, a)$: Transition probability distribution (market dynamics).
- $R(s, a, s')$: Reward function (e.g., Change in Net Worth).
- $\gamma \in [0, 1]$: Discount factor.

2 Value Function and Bellman Equation

The goal is to find a policy $\pi : S \rightarrow A$ that maximizes the expected cumulative discounted reward:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (1)$$

The optimal action-value function $Q^*(s, a)$ satisfies the Bellman Optimality Equation:

$$Q^*(s, a) = \mathbb{E}_{s' \sim P(\cdot|s, a)} \left[R(s, a, s') + \gamma \max_{a'} Q^*(s', a') \right] \quad (2)$$

3 Q-Learning Algorithm

Since the market transition probabilities P are unknown, we use Q-Learning, a model-free algorithm. The update rule is derived from the temporal difference error:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right) \quad (3)$$

where α is the learning rate.

4 Application to Trading

In our implementation:

- **State**: Defined by price trend (Up/Down) and position status (Long/Neutral).
- **Action**: Discrete actions $\{0 : \text{Hold}, 1 : \text{Buy}, 2 : \text{Sell}\}$.
- **Reward**: The immediate change in portfolio value (Realized + Unrealized PnL).

By interacting with the environment, the agent approximates $Q^*(s, a)$ effectively learning a trading strategy that maximizes long-term wealth.

5 Deep Q-Networks (DQN)

In high-dimensional or continuous state spaces, maintaining a Q-table is infeasible. DQN approximates the Q-function using a neural network $Q(s, a; \theta)$ with weights θ . The loss function used to train the network is:

$$L(\theta) = \mathbb{E}[(y - Q(s, a; \theta))^2] \quad (4)$$

where $y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$ is the target, computed using a separate *target network* with weights θ^- to stabilize training. Additionally, *Experience Replay* is used to break correlations between consecutive samples.

6 Proximal Policy Optimization (PPO)

PPO improves upon standard policy gradient methods by preventing large, destructive policy updates. It uses a clipped surrogate objective function:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t) \right] \quad (5)$$

where:

- $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio.
- \hat{A}_t is the estimated advantage.
- ϵ is a hyperparameter (usually 0.1 or 0.2).

This ensures the new policy does not deviate too far from the old one, enabling stable training.

7 Generalized Advantage Estimation (GAE)

To reduce variance in policy gradient estimates, Schulman et al. introduced GAE. The advantage $\hat{A}_t^{GAE(\gamma, \lambda)}$ is defined as an exponentially weighted average of k -step advantages:

$$\hat{A}_t^{GAE} = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{t+l}^V \quad (6)$$

where $\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD error. λ controls the bias-variance trade-off.

8 Partially Observable MDPs (POMDP) and DRQN

Financial markets are partially observable; the full state of the economy is never known. This problem is modeled as a POMDP. Deep Recurrent Q-Networks (DRQN) address this by replacing the first fully connected layer of a DQN with a Recurrent Neural Network (RNN) or LSTM.

1. The agent receives an observation o_t (not the full state s_t).
2. The LSTM maintains an internal hidden state $h_t = \text{LSTM}(h_{t-1}, o_t)$, which acts as a summary of the history.
3. The Q-function is approximated as $Q(h_t, a; \theta)$.

This allows the agent to integrate information over time and detect regimes or trends that are not visible in a single snapshot.

9 Conclusion

Reinforcement Learning in finance requires handling non-stationarity, partial observability, and noise. By employing advanced techniques like PPO for stability, GAE for variance reduction, and DRQN for temporal memory, we can build robust trading agents capable of adapting to complex market dynamics.