Asset Allocation Using Temporal Difference Learning

LI Yiying

17. März 2025

1 Introduction

This report explores the implementation and evaluation of a Temporal Difference (TD) Learning-based investment strategy for discrete-time asset allocation. The problem is modeled as a Markov Decision Process (MDP), where an investor allocates wealth between risky and risk-free assets over T=10 time steps. The objective is to learn the optimal investment strategy that maximizes the Constant Absolute Risk Aversion (CARA) utility of final wealth. This report includes a detailed explanation of the implementation, including function docstrings, a description of testing methodology covering unit, system, and acceptance tests, and analysis of five test cases verifying the effectiveness of the strategy.

2 Implementation

The problem is formulated as a discrete-state MDP with finite time steps. The primary components include an investment environment (`Asset` class) that simulates the process of allocating wealth between risky and risk-free assets, a TD Learning Algorithm (`td_learning` function) that implements an epsilon-greedy Q-learning approach to learn optimal policies, and a testing framework that ensures correctness and stability of the implementation through structured validation. The Asset class defines the investment environment where initial wealth is $w_0 = 100$, at each time step wealth is allocated between risky and risk-free assets. The risky asset has uncertain returns, yielding a with probability p and b with probability $1 - p$, while the risk-free asset provides a constant return r. The TD Learning algorithm is implemented using epsilon-greedy Q-learning to balance exploration and exploitation while updating the Q-table.

3 TD Learning Algorithm

The TD learning function follows an epsilon-greedy Q-learning approach:

```python
def td_learning(env, lr, num_episodes, eps=0.1, min_eps=0.01):
    """
    Temporal Difference (TD) learning algorithm.

    Args:
        env (Asset): The investment environment.
        lr (float): Learning rate.
        num_episodes (int): Number of training episodes.
        eps (float): Initial exploration probability.
        min_eps (float): Minimum exploration probability.

    Returns:
        np.array: Trained Q-table.
    """
    Q = np.zeros((env.T, env.num_actions))
    eps_decay = (eps - min_eps) / num_episodes
    for _ in range(num_episodes):
        state = env.reset()
        eps -= eps_decay
        done = False
        while not done:
            action = np.random.randint(0, env.num_actions) if np.random.
                uniform(0, 1) < eps else np.argmax(Q[state, :])
            next_state, _, reward, done = env.step(action)
            next_action = np.random.randint(0, env.num_actions) if np.
                random.uniform(0, 1) < eps else np.argmax(Q[state, :])
            Q[state, action] += lr * (reward + (0 if done else Q[next_state
                , next_action]) - Q[state, action])
            state = next_state
            action = next_action
    return Q
```

4 Testing Methodology

To evaluate the strategy, we employ five test cases, each representing different market conditions. The testing approach includes unit tests that validate correctness of key functions (`utility()`, `step()`), system tests that assess the full execution of the MDP, and acceptance tests that ensure learned strategies align with economic reasoning. The following table summarizes test cases and expected outcomes:

Test Case	Parameters (a, b, p, r)	Expected Behavior
1. Always High Return	$(0.8, 0.5, 0.4, 0.1)$	Full investment in risky asset.
2. Always Low Return	$(0.1, 0.1, 1.0, 0.8)$	Full investment in risk-free asset.
3. Equal Expected Return	$(0.6, 0.2, 0.5, 0.4)$	Conservative investment due to uncertainty.
4. High Expected Return	$(0.8, 0.2, 0.5, 0.4)$	Greater investment in risky asset.
5. Small Risk Premium	$(0.3, 0.1, 0.4, 0.15)$	Early aggressive investment, later conservative.

Tab. 1: Test Cases and Expected Results

5 Conclusion

The results confirm that TD Learning successfully learns optimal investment strategies under various market conditions. The observed policies align with economic intuition: when the expected return of the risky asset significantly exceeds that of the risk-free asset, the model learns to invest more in the risky asset. Key takeaways include that TD Learning adapts to different financial scenarios and determines optimal allocations dynamically, higher expected returns for risky assets encourage higher investments, while lower risk premiums promote more conservative strategies, and the training stability is affected by market conditions, requiring careful tuning of training epochs. Future work could extend the model to continuous action spaces, implement deep reinforcement learning (e.g., DQN) for better function approximation, and incorporate real financial datasets for empirical validation. This study demonstrates how reinforcement learning techniques can be applied to asset allocation, offering an adaptive, data-driven approach to investment decision-making.