MSBD6000M Assignment 1

WONG Chong Ki, 20978851
MSc in Big Data Technology, HKUST
Email: ckwongch@connect.ust.hk

WENG Yanbing, 21091234
MSc in Big Data Technology, HKUST
Email: ywengae@connect.ust.hk

Abstract

In this assignment, we investigate a finite-horizon discrete-time asset-allocation problem using tabular Q-learning under various market scenarios. We focus on a CARA utility function and a two-point risky asset return model, contrasting the agent's learning performance across different probabilities and return rates.

1 Analytical Solution

In this assignment, we study a discrete-time asset-allocation problem with a finite time horizon $T = 10$. At each discrete time step $t = 0, 1, \ldots, 9$, the agent holds a current wealth W_t and must decide how much $x_t \in [0, W_t]$ to invest in a *risky* asset, while the remainder $(W_t - x_t)$ goes into a *riskless* asset. Let:

- The risky asset yield Y_t be a random variable taking values

$$Y_t = \begin{cases} a, & \text{with probability } p, \\ b, & \text{with probability } (1-p), \end{cases}$$

- The riskless asset has a fixed interest rate r,
- The horizon is $T = 10$,
- The final utility at $t = T$ is given by a negative exponential (CARA) utility function

$$U(W_{10}) \;=\; -\frac{1}{\alpha}\exp\big(-\alpha\, W_{10}\big).$$

1.1 State Dynamics and Reward

Once we decide x_t, the next-state wealth is

$$W_{t+1} \;=\; x_t\,(1+Y_t) \;+\; \big(W_t - x_t\big)\,(1+r).$$

Because there is no intermediate consumption, the only nonzero reward occurs at the final stage $t = 9 \to t = 10$, when we realize the utility:

$$R_{10} \;=\; U\big(W_{10}\big) \;=\; -\,\frac{1}{\alpha}\,\exp\big(-\,\alpha\,W_{10}\big).$$

We treat this problem as a finite-horizon Markov Decision Process (MDP) with discount factor $\gamma = 1$. In any of the first $T-1$ stages, the reward is 0, while at $t = 10$ the reward is $U(W_{10})$.

1.2 Bellman Formulation

Let $V_t(W)$ denote the value function at time t when the agent's wealth is W. For $t = 0, 1, \ldots, 9$, we have

$$V_t(W) \;=\; \max_{0\le x\le W} E\big[V_{t+1}(W_{t+1})\big],$$

where

$$W_{t+1} \;=\; x\,(1+Y_t) + (W-x)\,(1+r).$$

At the terminal step,

$$V_{10}(W_{10}) \;=\; U(W_{10}) \;=\; -\frac{1}{\alpha}\exp\big(-\,\alpha\,W_{10}\big).$$

1.3 Analytical Derivation under CARA Utility

With the negative-exponential utility (CARA), it is a standard result that the value function retains an exponential form. We outline the key steps:

1. **Terminal condition:** At $t = 10$,

$$V_{10}(W) = U(W) = -\frac{1}{\alpha}\exp(-\alpha\,W).$$

2. **Ansatz for V_t:** Assume

$$V_t(W) = -\,b_t\,\exp\big(-\,c_t\,W\big),$$

for some parameters b_t and c_t that may depend on t but not on W.

3. **Bellman recursion:** For $t = 0, 1, \ldots, 9$,

$$V_t(W) = \max_x \Big\{ p \left[-b_{t+1} e^{-c_{t+1}\,[\,x(1+a)+(W-x)(1+r)\,]} \right] + (1-p) \left[-b_{t+1} e^{-c_{t+1}\,[\,x(1+b)+(W-x)(1+r)\,]} \right] \Big\}.$$

One factors out an exponential in $-\,c_{t+1}\,W$; differentiating w.r.t. x yields a closed-form solution

$$x_t^* \;=\; \frac{1}{c_{t+1}\,\big[(b-r)\;-\;(a-r)\big]}\;\ln\!\Big(\,\frac{(1-p)\,\big(r-b\big)}{p\,\big(a-r\big)}\Big).$$

Because the negative-exponential utility exhibits constant absolute risk aversion, x_t^* does not depend on the current wealth W.

4. **Parameters update:** One solves for (b_t, c_t) recursively, starting from $(b_{10}, c_{10}) = \big(\frac{1}{\alpha}, \alpha\big)$. With CARA and two-point returns, the solution remains exponential in form throughout backward induction.

1.4 Key Observations

- **Optimal policy independent of W.** The fraction (or absolute x_t^*) does not scale with wealth.
- **Closed-form solution.** This is simpler than, e.g., CRRA utility, where the solution might scale with wealth.
- **Numerical check via Q-learning.** By discretizing wealth and actions, tabular Q-learning approximates the same policy, converging over enough episodes.

1.5 Conclusion

Hence, for a finite-horizon discrete-time setting with CARA utility and a two-point risky asset distribution, the *optimal strategy* x_t^* is time-dependent but wealth-invariant. The value function is exponential, and the Q-learning code can reproduce this solution if properly discretized.

2 Q-learning Approach

2.1 Brief Introduction to Q-learning

Q-learning is an off-policy, model-free Reinforcement Learning (RL) algorithm based on temporal-difference (TD) learning. It maintains a table (or function approximation) of $Q(s, a)$, representing the expected return (sum of discounted rewards) after

taking action a in state s, and thereafter following an optimal policy. Its update rule is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\Big[r + \gamma \max_{a'} Q(s',a') \; - \; Q(s,a)\Big],$$

where r is the immediate reward, s' is the next state, γ is the discount factor, and α is the learning rate. Even without a perfect model of the environment, Q-learning converges to the optimal action-value function, given sufficient exploration.

2.2 Applying Q-learning to Our Asset-Allocation Problem

In our discrete-time CARA setting:

- **States**: (t, W), where $t \in \{0, \ldots, 9\}$ is the time step, and W is the discretized wealth.
- **Actions**: $x \in \{0, 50, 100, \ldots, W\}$, i.e. how much to invest in the risky asset, in increments of some step size (e.g. 50).
- **Reward**: Zero until $t = 9 \to t = 10$, at which time reward is $U(W_{10})$. We discount with $\gamma = 1$ for a finite horizon.
- **Temporal Difference Update**: After each step from (t, W) via action x to $(t+1, W')$ with reward r, we perform

$$Q(t,W,x) \leftarrow Q(t,W,x) + \alpha\Big[r + \max_{x'} Q(t+1,W',x') - Q(t,W,x)\Big].$$

By repeating many episodes, Q-learning eventually approximates the optimal policy $\pi^*(t, W)$ that maximizes expected utility at the end.

2.3 Implementation Details: Classes and Functions

Our Python code has three main classes:

- **Environment**: Manages discrete wealth levels up to W_{MAX}, with transitions given by the two-point return (a, b) and the riskless rate r. Also computes the final utility at $t = 9$.
- **Agent**: Maintains a tabular $Q[t, w, a]$ array. Chooses actions via ε-greedy, calls `update_q_table` for each step to apply Q-learning. The method `compute_epsilon(episode` does exponential decay from `epsilon_start` to `epsilon_end`.
- **Trainer**: Runs multiple episodes, logs Q-diff and final wealth. The method `train()` iterates over episodes, prints an average Q-diff and final wealth from the last 100 episodes every 1000 steps, and finally plots the training curves.

3 Experimental Results and Analysis

We ran four different market scenarios, each for **30,000 episodes**, investigating how the agent converges under different probabilities and returns. In the logs, we print "Avg Q-diff Last 100" and "Avg Wealth Last 100" for smoothing.

3.1 Scenario 1 - High probability of large positive return

Parameters: $p = 0.8,\ a\_ret = 0.6,\ b\_ret = -0.3,\ \text{riskless\_ret} = 0.02.$

Table 1: Scenario 1 Training Log Excerpt

Episode	eps	Avg Q-diff Last 100	Avg Wealth Last 100
1000	0.0015	0.1999	1767.50
2000	0.0001	0.1243	2434.00
6000	0.0001	0.0944	3043.00
10000	0.0001	0.0570	4206.00
15000	0.0001	0.0325	5032.50
30000	0.0001	0.0417	5742.00

As seen in Table I, the agent's final wealth can exceed 5000–9000 range by the end. The training error eventually dips near 0.04. Figure 1 shows the Q-diff plot and the wealth plot.

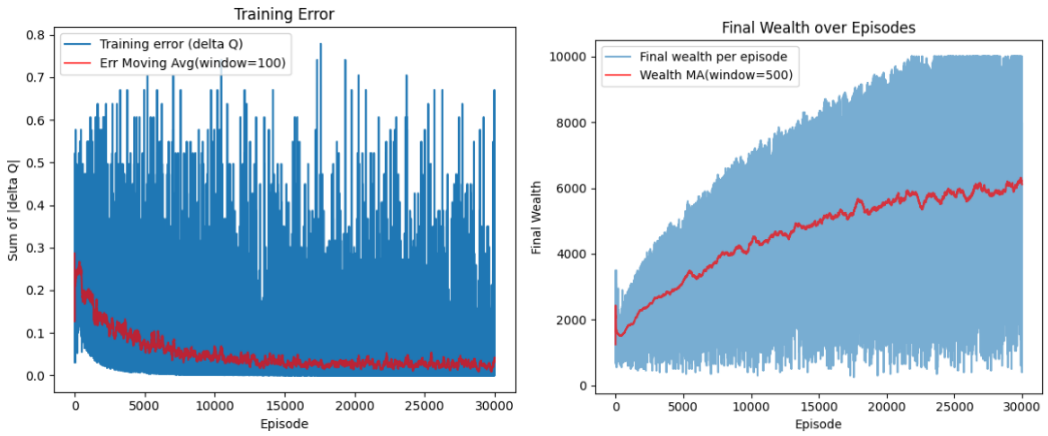

Figure 1: Scenario 1: Q-diff (left) and Final Wealth (right).

3.2 Scenario 2 - Moderately favorable returns

Parameters: $p = 0.7,\ a\_ret = 0.4,\ b\_ret = -0.2,\ \text{riskless\_ret} = 0.01.$

Table 2: Scenario 2 Training Log Excerpt

Episode	eps	Avg Q-diff Last 100	Avg Wealth Last 100
1000	0.0015	0.2521	1470.00
3000	0.0001	0.1381	2219.50
7000	0.0001	0.1014	2668.50
12000	0.0001	0.1065	2851.00
20000	0.0001	0.0653	3257.00
30000	0.0001	0.1168	2692.50

The final wealth is more modest, typically 2000–3000, with occasional spikes or dips. The Q-diff hovers around 0.06–0.12 near the end. Figure 2 confirms partial but not perfect convergence, reflecting a moderate positive expectation.

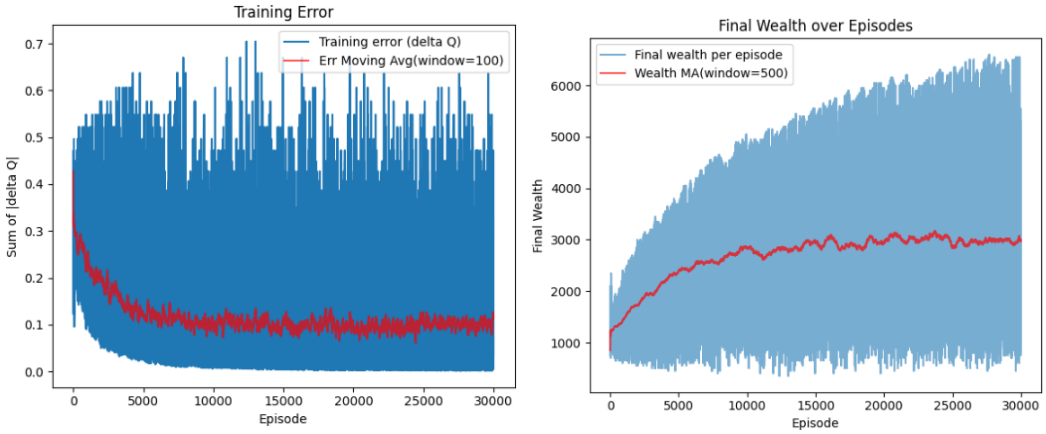

Figure 2: Scenario 2: Q-diff (left) and Final Wealth (right).

3.3 Scenario 3 - Balanced returns, moderate risk

Parameters: $p = 0.6,\ a\_ret = 0.35,\ b\_ret = -0.05,\ \text{riskless\_ret} = 0.015.$

Table 3: Scenario 3 Training Log Excerpt

Episode	eps	Avg Q-diff Last 100	Avg Wealth Last 100
1000	0.0015	0.2099	1612.50
3000	0.0001	0.1162	2411.50
7000	0.0001	0.0930	2634.50
12000	0.0001	0.0852	2776.50
20000	0.0001	0.0885	2651.00
30000	0.0001	0.0950	2675.50

As shown in Table III, final wealth typically lies between 2000–3000. The risk is lower, so the agent's net returns are moderate. Q-diff around 0.08–0.10 indicates partial convergence (Figure 3).

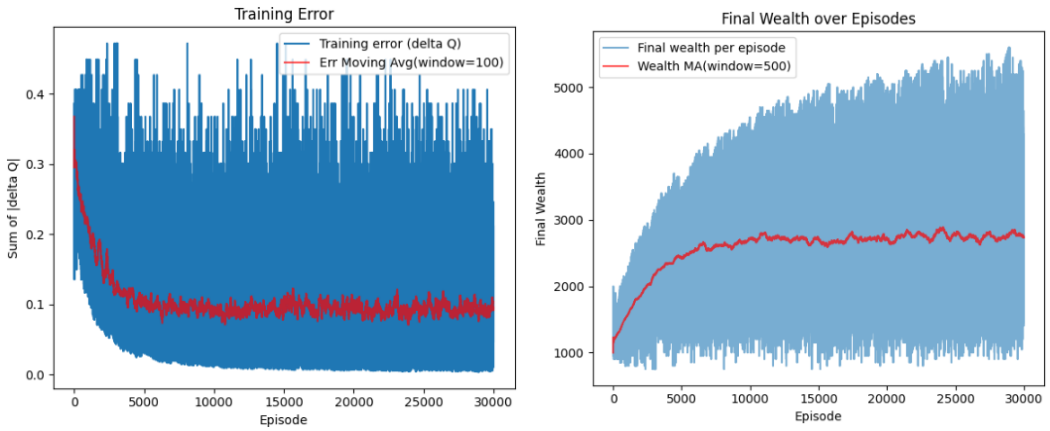

Figure 3: Scenario 3: Q-diff (left) and Final Wealth (right).

3.4 Scenario 4 - Very high probability, moderate high return

Parameters: $p = 0.9,\ a\_ret = 0.5,\ b\_ret = -0.2,$ riskless_ret $= 0.03$.

Table 4: Scenario 4 Training Log Excerpt

Episode	eps	Avg Q-diff Last 100	Avg Wealth Last 100
1000	0.0015	0.0882	2607.50
3000	0.0001	0.0441	3697.50
7000	0.0001	0.0273	4420.50
12000	0.0001	0.0225	4975.50
20000	0.0001	0.0181	7346.50
30000	0.0001	0.0059	7677.00

With a 90% chance of +50%, the agent invests heavily. Final wealth surpasses 7000–8000 in the last episodes. The Q-diff dips below 0.01, as shown in Figure 4.

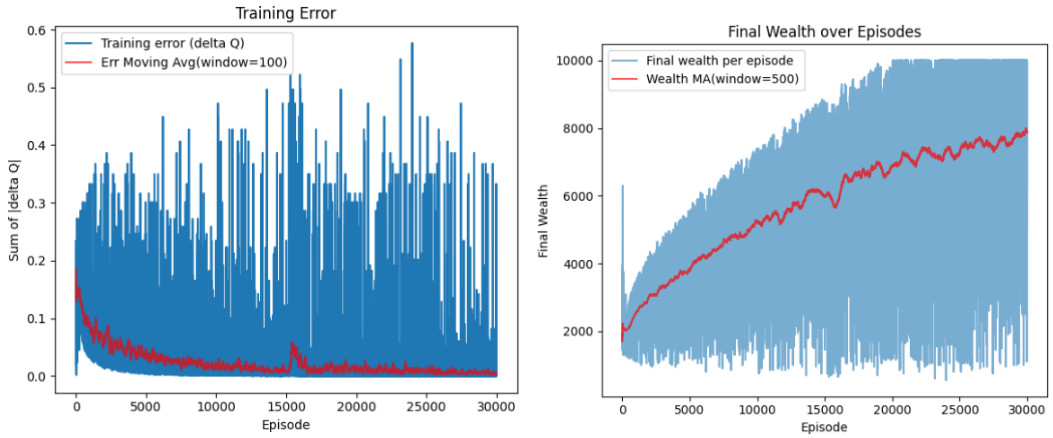

Figure 4: Scenario 4: Q-diff (left) and Final Wealth (right).

3.5 Overall Observations

In all scenarios, Q-diff generally diminishes over time, validating that tabular Q-learning is converging. Final wealth depends on the net positivity of the risky asset distribution:

- **Higher p and larger** a_ret lead the agent to invest more in the risky asset, raising final wealth (Scenarios 1 & 4).
- **Moderate returns or probabilities** yield more modest wealth, but stable partial convergence (Scenarios 2 & 3).

Hence, the tabular Q-learning approach successfully learns near-optimal policies for each set of parameters under negative-exponential (CARA) utility.