MDPs & Reinforcement Learning Assignment

K.N.Toosi University of Technology
Artificial Intelligence

Spring 2025

Part I

Practical Assignment

The agent interacts with a discrete environment consisting of states $\{A, B, C, D, X\}$ and deterministic actions $\{\leftarrow, \rightarrow\}$. During training, the agent observes the following transitions. Each row indicates a unique transition (s, a, s', r) and the Q-learning iterations in which it occurred.

Table 1: Q-learning Transition Trace

Iteration	**State** s	**Action** a	**Next State** s'	**Reward** r
1	A	$\rightarrow$	B	0
2	B	$\rightarrow$	C	0
3	C	$\rightarrow$	D	0
4	D	$\rightarrow$	X	1
5	A	$\rightarrow$	C	0
6	C	$\rightarrow$	X	1
7	A	$\rightarrow$	B	0
8	B	$\leftarrow$	A	0
9	A	$\rightarrow$	B	0
10	B	$\rightarrow$	C	0
11	C	$\leftarrow$	B	0
12	B	$\rightarrow$	C	0
13	C	$\rightarrow$	D	0
14	D	$\rightarrow$	X	1
15	A	$\rightarrow$	B	0
16	B	$\rightarrow$	C	0
17	C	$\rightarrow$	X	1
18	A	$\rightarrow$	C	0
19	C	$\rightarrow$	X	1
20	A	$\rightarrow$	B	0
21	B	$\rightarrow$	C	0
22	C	$\rightarrow$	D	0
23	D	$\rightarrow$	X	1
24	A	$\rightarrow$	C	0
25	C	$\rightarrow$	D	0
26	D	$\rightarrow$	X	1

(a) Using the episode data above, compute the following quantities assuming a model-based reinforcement learning approach:

1. At iteration 21, estimate $\hat{T}(A, \rightarrow, B)$ and $\hat{T}(A, \rightarrow, C)$.
2. At iteration 19, estimate $\hat{R}(C, \rightarrow)$.
3. At iteration 24, estimate $\hat{T}(C, \rightarrow, D)$ and $\hat{T}(C, \rightarrow, X)$.

4. At iteration 26, based on all available data, list all state-action pairs (s, a) that are deterministic. Justify your answer using the transition trace.

(b) Assume all Q-values are initialized to 0. The agent applies the Q-learning update rule with learning rate $\alpha = 1.0$ and a discount factor $\gamma > 0$:

$$Q(s, a) \leftarrow r + \gamma \cdot \max_{a'} Q(s', a')$$

Answer the following:

1. After which Q-learning iteration does the value of $Q(B, \rightarrow)$ become strictly positive?
2. After which iteration does $Q(A, \rightarrow)$ become strictly positive?
3. After iteration 23, what is the highest Q-value among all state-action pairs? Explain why that value is the highest.

Part II

Practical Assignment

The agent operates in the following gridworld. Actions: **Up**, **Down**, **Left**, **Right**. Moving into a wall results in no movement. Terminal states grant rewards upon entry and end the episode.

A	B	C
D	E	F
G	H	I

$$R(s,a,s') = \begin{cases} +216 & s' = F, \\ +36 & s' = A, \\ -25 & s' = E, \\ 0 & \text{otherwise.} \end{cases}$$

(a) Starting at G, for each $\gamma \in \{1, 1/6, 0.2\}$:

- Identify the optimal action sequence to maximize return.

(b) Assuming $\gamma = 0.8$:

$$T(B, \text{Right}, C) = p, \quad T(B, \text{Right}, E) = 1-p, T(B, \text{Left}, E) = p, \quad T(B, \text{Left}, A) = 1-p.$$

1. Derive symbolic $Q(B, \text{Left})$ and $Q(B, \text{Right})$ in terms of p (show expectation over next states).
2. Solve for $p \in [0, 1]$ at which the agent is indifferent between Left and Right.

Part III

Practical Assignment

Consider the grid below (4 columns × 3 rows). Actions into walls leave the agent in place.

J	K	L	M
N	O	P	Q
R	S	T	U

Terminal states: M (reward +100), N (reward -50). Others 0.
The agent uses linear function approximation:

$$Q(s,a) = w^\top \phi(s,a), \quad \phi(s,a) \in \{0,1\}^3$$

Binary features:

- $f_1(s,a) = 1$ if action a reduces Manhattan distance to goal state M, otherwise 0.
- $f_2(s,a) = 1$ if action a reduces Manhattan distance to trap state N, otherwise 0.
- $f_3(s,a) = 1$ if the resulting next state lies in the same row as the goal state M, otherwise 0.

1. With $w = [5, -10, 3]^\top$: Compute each $\phi(O,a)$, the corresponding $Q(O,a)$, and state which action is chosen.
2. If the agent nevertheless always chooses *Right* from O, what properties (signs or relative size) of w_1, w_2, w_3 must hold? Provide intuitive justification.
3. Design three binary feature functions $\phi_j(s,a) \in \{0,1\}$, one for each of the following cases:
   - A feature that should have a **positive weight** ($w_j > 0$), encouraging desirable behavior.
   - A feature that should have a **negative weight** ($w_j < 0$), discouraging undesirable behavior.
   - A feature that should have **zero weight** ($w_j = 0$), representing an irrelevant or redundant signal.

   For each feature, explain:

(a) What the feature captures (i.e., its condition over (s, a)),

(b) Why it should be assigned the specified weight sign,

(c) How it would influence the agent's decisions.

4. (a) Consider the transition:

$$(s = O,\ a = \text{Right},\ r = 0,\ s' = P),$$

where

$$\phi(O, \text{Right}) = [1,\ 0,\ 0]^\top, \quad \phi(P, \text{Right}) = [1,\ 1,\ 0]^\top,$$

$$w = [1,\ 1,\ 1]^\top, \quad \alpha = 0.5, \quad \gamma = 1.$$

Update w after one step if the next action is $a' = \text{Right}$.

Part IV
Implementation Assignment

Q-Learning-Based Trading Environment Project Description

Project Overview

This project focuses on applying traditional Q-learning to a simplified financial trading environment. The agent is trained to trade a single asset (e.g., a stock or cryptocurrency) using historical daily market data. The agent operates in a discrete action space with the goal of maximizing cumulative profit while managing trading costs.

Data Preparation

Students will collect historical daily data for a financial instrument, including adjusted close prices and volume. Optionally, technical indicators such as returns, RSI, or moving averages may be computed. All features will be discretized into categorical bins to enable the use of tabular Q-learning. For example, daily return may be binned into `Up`, `Flat`, or `Down`, and RSI may be binned into `Oversold`, `Neutral`, or `Overbought`.

Suggestion for Environment Design

A custom trading environment will be implemented using the `gym.Env` interface. The environment simulates trading one unit of an asset per day with the following specifications:

- **State**: A tuple of discretized feature values (e.g., binned returns, RSI, and volume).
- **Actions**: The agent selects from three discrete actions:
  - `0`: Short (take a short position)
  - `1`: Hold (neutral, no position)
  - `2`: Long (take a long position)
- **Reward**: At each step, the reward is calculated as the profit or loss from the selected position:

$$r_t = \text{position}_t \times \text{return}_t - \text{trading\_cost}$$

  where `return_t` is the price change from day t to $t+1$, and `position` is -1, 0, or +1.

- **Portfolio**: The environment tracks the agent's net asset value (NAV), initialized to 1. NAV updates over time according to the selected actions and market returns.
- **Episode Termination**: An episode runs for a fixed number of steps, or ends early if NAV reaches zero.

Q-Learning Agent

A tabular Q-learning agent is implemented with:

- **State space**: Formed from combinations of discretized feature bins.
- **Q-table**: Maps state-action pairs to expected future rewards.
- **Policy**: ϵ-greedy, balancing exploration and exploitation.
- **Update Rule**:

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

Training and Evaluation

The Q-learning agent is trained on a portion of the dataset (training set). A separate test set is used to evaluate the learned policy. Performance is compared against a Buy-and-Hold benchmark.

Analysis and Reporting

Students will analyze the agent's performance using plots and key metrics:

- Cumulative returns vs Buy-and-Hold
- Agent actions and NAV evolution over time
- Total return, Sharpe ratio, maximum drawdown, win/loss ratio

Deliverables

- Source code for environment and Q-learning agent
- Training and evaluation results with visualizations
- A brief report (2–3 pages) or a Jupyter notebook including explanations, methodology, and discussion of results

Note

Any attempt to use AI tools for generating the code is strictly prohibited. Students will be asked to present and explain their code during a class session.