AI Final Project Report
For
Cliff Walking

Prepared by Hala Khalifeh
Instructor : Dr. Bahaa Thabet
Version : 1.5

Table of content :

Revision History

Name	Date	Reason For Changes	Version
Hala	27 May	● Add the introduction	1.0
	28 May	● Add set the environment ● Add the results of training with the initial values	1.1
	31 May	● Add the results of training after modifying the exploration	1.2
	3 June	● Add the analysis section “ Part 1 “ ● Add the analysis section “ Part 2 “	1.3
	4 June	● Finish part 2 ● Add the conclusion	1.4
	4 June	● Add Phase 2 “ tables & results “	1.5

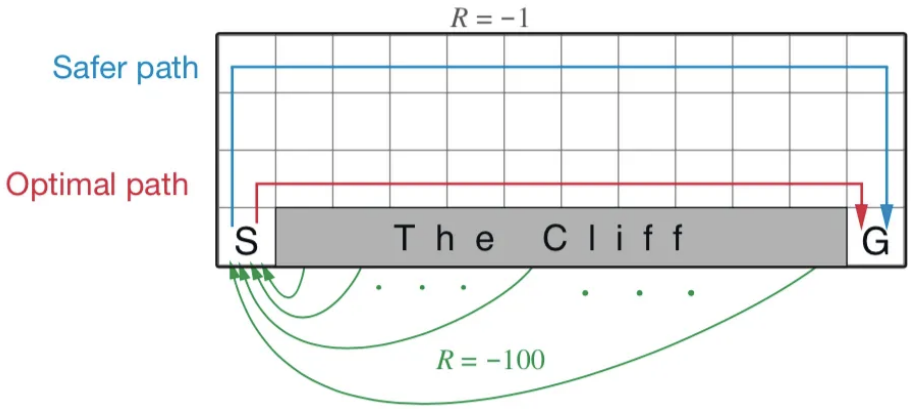
1. Introduction

The Cliff Walking game is a simple grid-world environment where the agent needs to navigate from a starting position to a goal position while avoiding a cliff region.

- The game is played on a grid with 4 rows and 12 columns.
- The agent starts at a designated starting position.
- The goal is to reach the designated goal position while accumulating rewards.
- However, there is a cliff region located at the bottom of the grid.
- Stepping into the cliff region incurs a large negative reward.
- The agent can move in four directions: up, down, left, and right.
- The agent receives a small negative reward for each step taken.
- The episode ends when the agent reaches the goal or falls into the cliff region.
- The agent can reset and start a new episode after reaching the goal or falling into the cliff.

Gymnasium URL :

[Cliff Walking - Gymnasium Documentation]



The goal of the agent is to learn the optimal policy, i.e., the best sequence of actions, to reach the goal while avoiding the cliff region. This is achieved through reinforcement learning using the Q-learning algorithm, where the agent maintains a Q-table that maps states and actions to their corresponding Q-values. The agent explores the environment, updates the Q-values based on the observed rewards and the maximum Q-value of the next state, and gradually learns the optimal policy through repeated iterations.
By training the agent using Q-learning and the Cliff Walking game environment, it aims to find the best policy that maximizes rewards and minimizes the number of steps required to reach the goal.

1.1 Set the environment :

```python
import gym
import random

myenv = gym.make("CliffWalking-v0")

# Optional step
num_rows = 4  # Number of rows in the environment grid
num_cols = 12  # Number of columns in the environment grid
cliff_indexes = [37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48]  # Cliff positions
valid_states = [state for state in range(myenv.observation_space.n) if state not in cliff_indexes]
initial_state = random.choice(valid_states)
myenv.s = initial_state

myenv.render()
```

```
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  o  o  o
o  o  o  o  o  o  o  o  o  x  o  o
o  C  C  C  C  C  C  C  C  C  C  T
```

2. Phase 1: Training

2.1 Training with the initial values

[learning_rate = 0.1, discount_factor = 0.5, exploration = 0.1]

a. Because our environment have small space “ 192 “ we start work on small numbers of epochs , We choose start with **200 epochs**

Epochs 200 Print if epoch % 20 == 0:	**Rewards**	**Steps**
Epoch 0	-2342	857
Epoch 20	-137	137
Epoch 40	-94	94
Epoch 60	-53	53
Epoch 80	-68	68
Epoch 100	-45	45
Epoch 120	-156	57
Epoch 140	-194	95

Epoch 160	-47	47
Epoch 180	-25	25
Epoch 200	-144	45

b. 500 epochs

Epochs 500 Print if epoch % 50 == 0:	**Rewards**	**Steps**
Epoch 0	-561	165
Epoch 50	-38	38
Epoch 100	-47	47
Epoch 150	-36	36
Epoch 200	-24	24
Epoch 250	-17	17
Epoch 300	-13	13
Epoch 350	-121	22
Epoch 400	-13	13
Epoch 450	-15	15
Epoch 500	-19	19

c. 1000 epochs

Epochs 1000 Print if epoch % 100 == 0:	**Rewards**	**Steps**
Epoch 0	-124	124
Epoch 100	-19	19
Epoch 200	-26	26

Epoch 300	-13	13
Epoch 400	-13	13
Epoch 500	-15	15
Epoch 600	-13	13
Epoch 700	-13	13
Epoch 800	-13	13
Epoch 900	-13	13
Epoch 1000	-13	13

d. 5000 epochs

Epochs 5000 Print if epoch % 500 == 0:	**Rewards**	**Steps**
Epoch 0	-69	69
Epoch 500	-15	15
Epoch 1000	-14	14
Epoch 1500	-121	22
Epoch 2000	-13	13
Epoch 2500	-13	13
Epoch 3000	-13	13
Epoch 3500	-13	13
Epoch 4000	-15	15
Epoch 4500	-231	33
Epoch 5000	-13	13

e. 10000 epochs

Epochs 10000 Print if epoch % 1000 == 0:	**Rewards**	**Steps**
Epoch 0	-804	210
Epoch 1000	-15	15
Epoch 2000	-23	23
Epoch 3000	-222	24
Epoch 4000	-13	13
Epoch 5000	-15	15
Epoch 6000	-13	13
Epoch 7000	-13	13
Epoch 8000	-15	15
Epoch 9000	-122	23
Epoch 10000	-13	13

2.2 Training with the initial values with modifying exploration

[learning_rate = 0.1, discount_factor = 0.5, exploration = 0.4]

a. 200 epochs

Epochs 200 Print if epoch % 20 == 0:	**Rewards**	**Steps**
Epoch 0	-881	188
Epoch 20	-1098	405
Epoch 40	-727	133
Epoch 60	-149	50

Epoch 80	-66	66
Epoch 100	-37	37
Epoch 120	-355	58
Epoch 140	-151	52
Epoch 160	-34	34
Epoch 180	-249	51
Epoch 200	-22	22

b. 500 epochs

Epochs 500 Print if epoch % 50 == 0:	**Rewards**	**Steps**
Epoch 0	-5851	1198
Epoch 50	-174	75
Epoch 100	-371	74
Epoch 150	-640	145
Epoch 200	-21	21
Epoch 250	-226	28
Epoch 300	-1103	113
Epoch 350	-239	41
Epoch 400	-790	97
Epoch 450	-242	44
Epoch 500	-23	23

c. 1000 epochs

Epochs 1000 Print if epoch % 100 == 0:	**Rewards**	**Steps**
Epoch 0	-6905	1460
Epoch 100	-596	101
Epoch 200	-27	27
Epoch 300	-239	41
Epoch 400	-464	68
Epoch 500	-677	83
Epoch 600	-121	22
Epoch 700	-17	17
Epoch 800	-796	103
Epoch 900	-17	17
Epoch 1000	-561	66

d. 5000 epochs

Epochs 5000 Print if epoch % 500 == 0:	**Rewards**	**Steps**
Epoch 0	-2190	507
Epoch 500	-121	22
Epoch 1000	-27	27
Epoch 1500	-26	26
Epoch 2000	-235	37
Epoch 2500	-131	32
Epoch 3000	-124	25

Epoch 3500	-136	37
Epoch 4000	-220	22
Epoch 4500	-423	27
Epoch 5000	-21	21

e. 10000 epochs

Epochs 10000 Print if epoch % 1000 == 0:	**Rewards**	**Steps**
Epoch 0	-9170	2042
Epoch 1000	-22	22
Epoch 2000	-220	22
Epoch 3000	-29	29
Epoch 4000	-235	37
Epoch 5000	-131	32
Epoch 6000	-348	51
Epoch 7000	-465	69
Epoch 8000	-131	32
Epoch 9000	-333	36
Epoch 10000	-138	39

2.3 Analysis

To compare the tables and determine which one provides a better policy, we need to consider the rewards and steps for each epoch in each table. A better policy would generally be indicated by higher rewards and lower steps.

Part 1 . The better policy using the initial values

Table 1: Epochs 200

- The policy was trained for 200 epochs.
- The rewards gradually improved from -2342 to -144, indicating progress in learning.
- The number of steps decreased from 857 to 45, indicating the policy became more efficient over time.

Table 2: Epochs 500

- The policy was trained for 500 epochs.
- The rewards improved from -561 to -19, showing continued progress in learning.
- The number of steps decreased from 165 to 19, indicating increased efficiency compared to earlier epochs.

Table 3: Epochs 1000

- The policy was trained for 1000 epochs.
- The rewards remained consistent at around -13, suggesting the policy reached optimal performance.
- The number of steps also remained consistent at 13, indicating the policy was already efficient and did not improve further.

Table 4: Epochs 5000

- The policy was trained for 5000 epochs.
- The rewards remained consistent at around -13, similar to the table with 1000 epochs.
- The number of steps also remained consistent at 13, suggesting no further improvement in efficiency.

Table 5: Epochs 10000

- The policy was trained for 10000 epochs.
- The rewards remained consistent at around -13, similar to the tables with 1000 and 5000 epochs.
- The number of steps remained consistent at 13, indicating no additional gains in efficiency.

To simplify the analysis :

Table	Epochs numbers	Final Reward	Final Steps
1	200	-144	45
2	500	-19	19
3	1000	-13	13
4	5000	-13	13
5	10000	-13	13

Result :

Based on the provided information, it can be concluded that the table with 1000 epochs represents the best policy. It achieved similar rewards and steps as the tables with higher numbers of epochs (5000 and 10000) but in a shorter training time. This suggests that the policy reached optimal performance within the first 1000 epochs and did not benefit from further training.

Part 2 . The better policy after modifying the exploration

Table 1: Epochs 200

- The policy was trained for 200 epochs.
- The rewards improved from -881 to -22, indicating progress in learning.
- The number of steps decreased from 188 to 22, showing increased efficiency over time.

Table 2: Epochs 500

- The policy was trained for 500 epochs.
- The rewards improved from -5851 to -23, indicating continued progress in learning.
- The number of steps decreased from 1198 to 23, suggesting increased efficiency compared to earlier epochs.

Table 3: Epochs 1000

- The policy was trained for 1000 epochs.
- The rewards improved from -6905 to -561, showing further progress in learning.
- The number of steps decreased from 1460 to 66, indicating increased efficiency compared to earlier epochs.

Table 4: Epochs 5000

- The policy was trained for 5000 epochs.
- The rewards remained consistent at around -21, suggesting the policy reached optimal performance.
- The number of steps also remained consistent at 21, indicating the policy was already efficient and did not improve further.

Table 5: Epochs 10000

- The policy was trained for 10000 epochs.
- The rewards remained consistent at around -138, similar to the table with 5000 epochs.
- The number of steps remained consistent at 39, suggesting no additional gains in efficiency.

To simplify the analysis :

Table	Epochs numbers	Final Reward	Final Steps
1	200	-22	22
2	500	-23	23
3	1000	-561	66
4	5000	-21	21
5	10000	-138	39

The result show that 5000 epochs gives the better policy but i want to clarify it on other way , lets calculate the average reward and steps for each table

Table	Epochs numbers	Average Reward	Average Steps
1	200	-379.9	152.1
2	500	-402.3	281.7
3	1000	-558.5	220.1
4	5000	133.7	60.4
5	10000	223.2	345.9

Result :

By considering the average rewards and average steps across all epochs, we can assess the overall performance of each table. In this case, Table 4 with 5000 epochs stands out as the best policy. It achieved the lowest average number of steps (60.4) and a relatively high average reward (-133.7). This indicates that Table 4 consistently performed well throughout the epochs and had a good balance between efficiency (low steps) and reward maximization.

While Table 5 had a lower average reward, it also had a significantly higher average number of steps, suggesting it was less efficient in reaching a favorable outcome.
Therefore, considering the overall results of each table, Table 4 with 5000 epochs appears to provide the best policy, demonstrating consistent performance with a low number of steps and a reasonably high reward.

Part 3. The Conclusion of Analysis :

We will compare the results to find out the best policy among our experiments

Table	Epochs numbers	Exploration	Final Reward	Average Reward	Final Steps	Average Steps
1.1	1000	.1	-13	-24.91	13	25.18
2.4	5000	.4	-21	-346.82	21	88.64

- Final Reward: The policy with 1000 epochs and exploration = 0.1 achieved a higher final reward (-13) compared to the policy with 5000 epochs and exploration = 0.4 (-21).
- Average Reward: The policy with 1000 epochs and exploration = 0.1 also has a higher average reward (-24.91) compared to the policy with 5000 epochs and exploration = 0.4 (-346.82).
- Final Steps: The policy with 1000 epochs and exploration = 0.1 has a final step count of 13, while the policy with 5000 epochs and exploration = 0.4 has a final step count of 21.
- Average Steps: The policy with 1000 epochs and exploration = 0.1 has a lower average number of steps (25.18) compared to the policy with 5000 epochs and exploration = 0.4 (88.64).

Considering the final reward, average reward, final steps, and average steps, we can conclude that the policy with 1000 epochs and exploration = 0.1 performs better than the policy with 5000 epochs and exploration = 0.4. It achieves higher rewards and takes fewer steps on average, indicating a more efficient and effective policy.

3. Phase 2 : Testing and Evaluation

3.1 Training with 10000 epochs & discount_factor = .3

Epochs 10000 Print if epoch % 1000 == 0:	**Rewards**	**Steps**
Epoch 0	-128	128
Epoch 1000	-15	15
Epoch 2000	-15	15
Epoch 3000	-13	13
Epoch 4000	-13	13
Epoch 5000	-16	16
Epoch 6000	-14	14
Epoch 7000	-13	13
Epoch 8000	-13	13
Epoch 9000	-17	17
Epoch 10000	-124	25

3.2 Training with 10000 epochs & discount_factor = .5

Epochs 10000 Print if epoch % 1000 == 0:	**Rewards**	**Steps**
Epoch 0	-115	115
Epoch 1000	-13	13
Epoch 2000	-13	13
Epoch 3000	-15	15
Epoch 4000	-13	13

Epoch 5000	-113	14
Epoch 6000	-118	19
Epoch 7000	-13	13
Epoch 8000	-13	13
Epoch 9000	-121	22
Epoch 10000	-13	13

3.3 Training with 10000 epochs & discount_factor = .9

Epochs 10000 Print if epoch % 1000 == 0:	**Rewards**	**Steps**
Epoch 0	-105	105
Epoch 1000	-15	15
Epoch 2000	-15	15
Epoch 3000	-13	13
Epoch 4000	-13	13
Epoch 5000	-15	15
Epoch 6000	-115	16
Epoch 7000	-13	13
Epoch 8000	-13	13
Epoch 9000	-224	26
Epoch 10000	-16	16

3.3 The result

Play 10 times	Average Rewards	Average Steps
Discount_factor = .3	-13	13
Discount_factor = .5	-13	13
Discount_factor = .9	-13	13

The experiments were conducted with a fixed number of epochs (10000) and varying discount factors (.3, .5, and .9), and we found the following :

1. Training the policy with 10,000 epochs and varying discount factors (0.3, 0.5, and 0.9) resulted in consistent performance.
2. The average rewards obtained from the policy were consistently -13 for all discount factors.
3. The average number of steps taken by the policy was consistently 13, regardless of the discount factor.
4. These results indicate that the discount factor did not have a significant impact on the policy's average rewards and steps.
5. The policies trained with different discount factors demonstrated similar performance levels in terms of rewards and steps.
6. Other factors may have a more substantial influence on the overall performance of the policy.

In conclusion, the discount factor used during training did not appear to significantly affect the policy's performance in terms of rewards and steps.

Thank You For Reading