Artificial Intelligence Laboratory 4: Reinforcement Learning DT8042 HT23, Halmstad University

Fabrice Bodson, fabbod23@student.hh.se,

Ali Murtza, alimur23@student.hh.se,

December 2023

Introduction

Q1: What is the learning objective of this lab? Please phrase it in your own words.

The goal of his 4th lab is to make us familiar with Value Iteration, Policy Iteration, Q Learning and TD Learning by manipulating and comparing the different functions in a Jupyter Notebook.

Task 1

Q2: List and describe below what concepts and variables were involved in creating the environment.

The concepts and variables in the environments are :

- Width and Height : to define the grid's size
- Noise: is the probability of not going in the intended direction
- The comes the immediate rewards of non-goal states
- Terminal/goal states: their locations in the grid and rewards
- Finally, the location of the Walls in the grid

Q3: Illustrate the given and the two environments you created.

These are the 2 environments that were created :
See on other pages because the bad output.

-	-	-	1
-	W	-	-1
-	-	-	-

Table 1: Grid n°1

-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	1	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	W	-	-	-	-	-	-	-	-	-	-	-
-	W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-1	-	-	-	-	-	-	W	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	W	-	W	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-1	-	-	-	-	-
-	-	-	-	-	-1	-	-	-	-1	-	-1	-1	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	W	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-1	-	-	-	-	-	-	-	-	-	-1	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	W	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 2: Grid n°2

Task 2

Q4: Please include/write the pseudo-code of your Policy Iteration and TD-learning implementation.

```
def policy_iteration(gamma, T, Rewards, Terminals, Walls):
    N, NA = T.shape[0], T.shape[2]
    V = np.zeros((N, 1))
    Policy = np.zeros((N, 1))

    while True:
        unchanged = True
        old_policy = Policy.copy()

        # Policy Evaluation
        while True:
            delta = 0
            for s in range(N):
                if Walls[s] == 1:
                    continue
                if Terminals[s] == 1:
                    V[s] = Rewards[s]
                    continue
                v = V[s].copy()
                a = int(Policy[s])
```

-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
-	W	-1	-	-	-	-	-	-	-	-	-	-	-	-
-	W	-	-	-	-	-	-	-	-	-	-	-	-	-
-	W	-	-	-	-	W	-	-1	-	-	-	-	-	-
-	W	-	-	-	-	-	-	-	-	-	-	-	-	-
-	W	W	-	-1	-	-	-	-	-	-	-	-	-	-
-	W	-	-	-	-	-	-	-	1	-	-	-	-	-
-	-	-	W	-	-	-	-	-	-1	W	-	-	W	-
-	-	-	W	-	-	-	-	-	-1	-	-	-	-	-
-	-	-	W	-	-	-	-	-	1	-	-	-	-	-
-	-	-	W	-	-	-	-	-	-	-	-	-	-	-
-	-	-	W	-	-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-	W	-	-	-	-1	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Table 3: Grid n°3

```
                V[s] = Rewards[s] + gamma * np.dot(T[s, :, a], V)
                delta = max(delta, np.abs(v - V[s]))
            if delta < 1e-1:
                break

        # Policy Iteration
        for s in range(N):
            if Walls[s] == 1:
                continue
            if Terminals[s] == 1:
                V[s] = Rewards[s]
                continue
            Q = np.zeros((NA, 1))

            for a in range(NA):
                Q[a] = Rewards[s] + gamma * np.dot(T[s, :, a], V)
            Policy[s] = np.argmax(Q)

            if old_policy[s] != Policy[s]:
                unchanged = False
        if unchanged:
            break
    return V, Policy


def TD_learning(N_episodes, alpha, gamma, T, Rewards, Terminals, Walls):
    N, NA = T.shape[0], T.shape[2]
    V = np.zeros((N, 1))
    Policy = np.zeros((N, 1))
    for e in range(N_episodes):
        s = int(np.floor(np.random.uniform(0, N - 1)))
        while Terminals[s] == 1 or Walls[s] == 1:
            s = int(np.floor(np.random.uniform(0, N - 1)))
        while Terminals[s] == 0:
            Q = np.zeros((NA, 1))
            for a in range(NA):
                Q[a] = Rewards[s] + gamma * np.dot(T[s, :, a], V)
            a = np.argmax(Q)
            u = np.random.uniform(0, 1)
            s1 = np.argmax(u < np.cumsum(T[s, :, a]))
            V[s] += alpha * (Rewards[s1] + gamma * V[s1] - V[s])
            Policy[s] = a
            s = s1
            alpha = alpha * 0.9999

    V[Terminals == 1] = Rewards[Terminals == 1]
    return V, Policy
```

Task 3

Q5: Compare value and policy iterations in terms of convergence time. Relate it to computational complexity, current implementation, and number of iterations needed.

After execution of the code :
Value Iteration converged in 29 iterations.
Policy Iteration converged in 6 iterations.

Q6: How are optimal policies changed with immediate reward values? Show some examples (similar to Figure 17.2b in AIMA).

By varying the immediate reward values optimal policies are changed for all algorithms as follows:
Higher positive rewards may encourage the agent to explore more and strive for higher cumulative rewards.
Negative rewards may discourage certain actions, influencing the agent's behavior.
By experimenting with different reward values and observing the resulting policies, it shows how each algorithm reacts to changes in the reward structure, influencing the agent's decision-making in the given environment.

Q7: Compare TD-Learning and Q-Learning results with each other. Also with value and policy iterations. Remember that value and policy iteration solve Markov Decision Processes where we know the model (T and Rewards). TD-Learning and Q-Learning are (passive and active, respectively) Reinforcement Learning methods that don't have the model but use data from simulations. Data are simulated from the model we know, but the model is not used in TD or Q-Learning.

comparing the results we gain:

- TD-Learning and Q-Learning are model-free methods that learn from simulated experiences without assuming knowledge of the model.
- TD-Learning is a passive learning method where the agent learns from experiences without actively exploring.
- Q-Learning is an active learning method where the agent explores the environment and updates its policy based on the observed experiences.
- Value Iteration and Policy Iteration use the known model (T and Rewards) to iteratively improve the value function and policy.

Q8: What are the effects of epsilon and alpha values and how are they modified?

The modifications to epsilon and alpha during the Q-learning implementation control the trade-off between exploration and exploitation and adjust the learning rate over time. These modifications aim to strike a balance between quickly adapting to new information and stabilizing the learning process as the algorithm gains more experience.

Q9: What is the effect of a number of episodes?

The number of episodes in the provided TD learning code directly affects the overall learning process and the quality of the resulting policy. Increasing the number of episodes typically leads to better-learned policies that more closely approximate the optimal policy.

Please include the pseudo-code of your implementation in the report.

```
    import numpy as np

def TD_learning_with_episodes(N_episodes, alpha, gamma, T, Rewards,
    Terminals, Walls):
    N, NA = T.shape[0], T.shape[2]
    V = np.zeros((N, 1))
    Policy = np.zeros((N, 1))

    for episode in range(N_episodes):
        s = int(np.floor(np.random.uniform(0, N - 1)))

        while Terminals[s] == 1 or Walls[s] == 1:
            s = int(np.floor(np.random.uniform(0, N - 1)))

        while Terminals[s] == 0:
            Q = np.zeros((NA, 1))

            for a in range(NA):
                Q[a] = Rewards[s] + gamma * np.dot(T[s, :, a], V)

            a = np.argmax(Q)
            u = np.random.uniform(0, 1)
            s1 = np.argmax(u < np.cumsum(T[s, :, a]))

            V[s] += alpha * (Rewards[s1] + gamma * V[s1] - V[s])
            Policy[s] = a
            s = s1

    V[Terminals == 1] = Rewards[Terminals == 1]
    return V, Policy

# Usage
gamma = 1
alpha = 0.9
N_episodes = 20000

V_TD, Policy_TD = TD_learning_with_episodes(N_episodes, alpha, gamma, T,
    Rewards, Terminals, Walls)
```

Task 4 (extra credits)

Q10: Describe how the environment is created in this task. Present and discuss your experiment findings.

The findings of the experiment are the transition probabilities, rewards, terminal state, walls, directions and grid dimensions.
In transition probabilities matrix defines the probability of transitioning from one state to another for each action. In this experiment, diagonal moves are possible, and the noise associated with actions is distributed across (next) states.
In rewards array represents the immediate rewards associated with each state. These rewards influence the agent's decision-making process. Positive rewards typically indicate desirable states, while negative rewards may represent obstacles or penalties.
In terminal states, terminal states are those where the episode ends, and no further actions can be taken. Terminal states often have associated final rewards.
Walls can influence the agent's movement by restricting certain paths. Understanding the locations of walls is important for pathfinding algorithms and for assessing the difficulty of navigating the environment.
The Directions array contains the labels for different actions, including the newly introduced diagonal movements (UL, UR, DL, DR).
The grid dimensions (W and H) indicate the width and height of the environment

Conclusion

Add some reflections/conclusions about the lab.
In conclusion, this report provided a comprehensive exploration of reinforcement learning algorithms, including Value Iteration, Policy Iteration, Q-Learning, and TD-Learning. The environments were defined by grid dimensions, noise, rewards, terminal states, and wall locations. The pseudo-codes for Policy Iteration and TD-Learning were presented. Comparisons between value and policy iterations highlighted the convergence times, with policy iteration generally converging faster. Changes in optimal policies due to variations in immediate reward values were discussed, emphasizing the impact on agent behavior. Furthermore, a comparison of TD-Learning and Q-Learning emphasized their model-free nature, with TD-Learning being passive and Q-Learning being active. The effects of epsilon and alpha values in Q-Learning were discussed, addressing the balance between exploration and exploitation and learning rate adjustments. Finally, the influence of the number of episodes on TD-Learning was underscored, indicating its crucial role in determining the quality of the learned policy. The report provides valuable insights into the intricate dynamics of different reinforcement learning algorithms in various scenarios.