Homework : Reinforcement Learning

Baghino, Gianluca & Gallego, Natalia
ENSTA Paris
gianluca.baghino@ensta-paris.fr
natalia.gallego@ensta-paris.fr

I. Question 1 : Enumerate all the possible policies π : S -¿ A

Policy π assigns an action a for each state. π represents a reinforcement learning policy. It is a rule or strategy that specifies the action an agent should take in each state to maximize its long-term reward.

- From S_0: we can choose the actions a_1, or a_2.
- From S_1: we can choose the actions a_0.
- From S_2: we can choose the actions a_0.
- From S_3: we can choose the actions a_0.

Therefore, the possible policies are:

- π_1 :
  $\pi(S_0) = a_1 \quad \pi(S_1) = a_0$
  $\pi(S_2) = a_0 \quad \pi(S_3) = a_0$
- π_2 :
  $\pi(S_0) = a_2 \quad \pi(S_1) = a_0$
  $\pi(S_2) = a_0 \quad \pi(S_3) = a_0$

II. Question 2 : Write the equation for each optimal value function for each state

The equation for the optimal value function $V^*(S_0)$ is given by:

$$V^*(S) = R(S) + \lambda \max_a \sum_{S'} T(S, a, S')V^*(S')$$

We will evaluate each action a_0, a_1, and a_2 for each S.

For this we will use the following matrices given by the problem.

$$T(S, a_0, S') = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1-x & 0 & x \\ 1-y & 0 & 0 & y \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$T(S, a_1, S') = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$T(S_0, a_2, S') = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

A. S_0

For Action a_0 : The transition function is:
Thus, the sum evaluates to:

$$\sum_{S'} T(S_0, a_0, S')V^*(S') =$$

$$0 \cdot V^*(S'_0) + 0 \cdot V^*(S'_1) + 0 \cdot V^*(S'_2) + 0 \cdot V^*(S'_3) = 0$$

Therefore, for action a_0:

$$\sum_{S'} T(S_0, a_0, S')V^*(S') = 0$$

For Action a_1: The transition function is:
The sum evaluates to:

$$\sum_{S'} T(S_0, a_1, S')V^*(S') =$$

$$0 \cdot V^*(S'_0) + 1 \cdot V^*(S'_1) + 0 \cdot V^*(S'_2) + 0 \cdot V^*(S'_3) = V^*(S'_1)$$

For Action a_2 : The transition function is:
The sum evaluates to:

$$\sum_{S'} T(S_0, a_2, S')V^*(S') =$$

$$0 \cdot V^*(S'_0) + 0 \cdot V^*(S'_1) + 1 \cdot V^*(S'_2) + 0 \cdot V^*(S'_3) = V^*(S'_2)$$

Now, we can combine the contributions from each action into the equation for $V^*(S_0)$:

$$V^*(S_0) = 0 + \lambda \max\left(0, V^*(S'_1), V^*(S'_2)\right)$$

S_1

1) For a_0: :

$$\sum_{S'} T(S_1, a_0, S')V^*(S') =$$

$$(1-x)V^*(S'_1) + xV^*(S'_3)$$

2) For a_1: :

$$\sum_{S'} T(S_1, a_1, S')V^*(S') = 0$$

3) For a_2: :

$$\sum_{S'} T(S_1, a_2, S')V^*(S') = 0$$

Thus, the equation for $V^*(S_1)$ simplifies to:

$$V^*(S_1) = \lambda \max\left((1-x)V^*(S_1') + xV^*(S_3'), 0\right)$$

$$V^*(S_1) = \lambda((1-x)V^*(S_1') + xV^*(S_3'))$$

S_2

Evaluating each action:

4) For a_0: :

$$\sum_{S'} T(S_2, a_0, S')V^*(S') =$$

$$(1-y) * V^*(S_0') + y * V^*(S_3')$$

5) For a_1: :

$$\sum_{S'} T(S_2, a_1, S')V^*(S') = 0$$

6) For a_2: :

$$\sum_{S'} T(S_2, a_2, S')V^*(S') = 0$$

Thus, the equation for $V^*(S_2)$ becomes:

$$V^*(S_2) = 1 + \lambda \max\left((1-y) * V^*(S_0') + y * V^*(S_3'), 0\right)$$

$$V^*(S_2) = 1 + \lambda((1-y) * V^*(S_0') + y * V^*(S_3'))$$

4. For S_3

Evaluating each action:

- **For** a_0:

$$\sum_{S'} T(S_3, a_0, S')V^*(S') = V^*(S_0')$$

- **For** a_1:

$$\sum_{S'} T(S_3, a_1, S')V^*(S') = 0$$

- **For** a_2:

$$\sum_{S'} T(S_3, a_2, S')V^*(S') = 0$$

Thus, the equation for $V^*(S_3)$ simplifies to:

$$V^*(S_3) = 10 + V^*(S_0')$$

III. Question 3 : Is there exist a value for x, that for all $\lambda \in [0,1)$, and y $\in [0,1]$, $\pi^*(S_0)$ = a2, Justify your answer

Taking into account the value we obtained for $V*(S_0)$ in the previous equation, we can conclude that to meet the criteria we are asked for we must meet:

$$V^*(S_2') \geq V^*(S_1')$$

Using the other expressions obtained we can conclude that:

$$1+\lambda[(1-y)V*(S_0')+yV*(S_3')] \geq \lambda[(1-x)V*(S_1')+xV*(S_3')]$$

$$V^*(S_3) = 10 + V^*(S_0')$$

$$1+\lambda[V*(S_0')+10y] \geq \lambda[x(10+V*(S_0')-V*(S_1'))+V*(S_1')]$$

$$x \leq \frac{1+\lambda[10y+V*(S_0')-V*(S_1')]}{\lambda[10+V*(S_0')-V*(S_1')]}]$$

IV. Question 4 : Is there exist a value for y, that for all x ≥ 0, and y $\in [0,1]$, $\pi^*(S_0)$ = a2, Justify your answer

In the same way as before, we use the expressions found in Question 2. With this, we know that we must satisfy the property:

$$V^*(S_1') \leq V^*(S_2')$$

$$1+\lambda[V*(S_0')+10y] \leq \lambda[x(10+V*(S_0')-V*(S_1'))+V*(S_1')]$$

$$10\lambda y \leq \lambda[x(10+V*(S_0')+V*(S_1'))+V*(S_1')-V*(S_0')]-1$$

$$y \leq \frac{10x+V*(S_0')(x-1)+V*(S_1')(x+1)}{10} + \frac{1}{10\lambda}$$

V. Question 5

```
gamma = 0.9
x, y = 0.25, 0.25
epsilon = 0.0001

rewards = {
    'S0': {'a0': 0, 'a1': 0, 'a2': 0},
    'S1': {'a0': 0, 'a1': 0, 'a2': 0},
    'S2': {'a0': 1, 'a1': 0, 'a2': 0},
    'S3': {'a0': 10, 'a1': 0, 'a2': 0}
}

transitions = {
    'S0': {'a0': [0, 0, 0, 0], 'a1': [0, 1, 0, 0],
    'a2': [0, 0, 1, 0]},
    'S1': {'a0': [0, 1 - x, 0, x], 'a1': [0, 0, 0,
    0], 'a2': [0, 0, 0, 0]},
```

```
    'S2': {'a0': [1 - y, 0, 0, y], 'a1': [0, 0, 0,
    0], 'a2': [0, 0, 0, 0]},
    'S3': {'a0': [1, 0, 0, 0], 'a1': [0, 0, 0, 0],
    'a2': [0, 0, 0, 0]}
}

V = {'S0': 0, 'S1': 0, 'S2': 0, 'S3': 0}

def value_iteration():
    global V
    iteration = 0
    while True:

        V_old = V.copy()
        max_delta = 0

        for s in V:
            q_values = []

            for a in rewards[s]:

                q_value = rewards[s][a]

                for next_state, prob in zip(V,
    transitions[s][a]):
                    q_value += gamma * prob * V_old
    [next_state]

                q_values.append(q_value)

            V[s] = max(q_values)

            max_delta = max(max_delta, abs(V[s] -
    V_old[s]))

        iteration += 1

        if max_delta < epsilon:
            print(f"Converged after {iteration}
    iterations.")
            break

value_iteration()

print("Optimal Values (V*):", V)
```

1. Parameters and Rewards Initialization

- **Discount Factor** γ: We initialize the discount factor γ to **0.9**. This parameter helps balance immediate rewards and future rewards.
- **Transition Probabilities**: The values for x and y are set to **0.25** each. These values influence the probabilities of moving from one state to another based on the action taken.
- **Rewards**: We define a ‘rewards‘ dictionary for each state-action pair:
  - $R(S_0, a_0) = 0$, $R(S_0, a_1) = 0$, and $R(S_0, a_2) = 0$.
  - $R(S_2, a_0) = 1$ and $R(S_3, a_0) = 10$.
- **Termination Condition**: We set $\epsilon = 0.0001$, defining the threshold for convergence. The iteration will stop once the values stabilize, with the difference between the old and updated values for each state falling below this threshold.

2. Transition Matrix

- **Transition Probabilities Setup**: Each state-action pair has a list of transition probabilities to other states. This using the T matrices given by the exercise.

3. Value Iteration Function

- **Initialization**: `V` is a dictionary that starts with zero values for each state ($\{S0, S1, S2, S3\}$). `V_old` is used to track the values from the previous iteration, enabling us to calculate the change in values to check for convergence.
- **Iteration Process**:
  - For each **state** S, the function iterates over **all possible actions** a in that state, calculating $Q(s, a)$ values based on:
    1) **Immediate Reward** $R(s, a)$: The value starts with the immediate reward for taking action a in state s.
    2) For each possible next state s', the code multiplies the **transition probability** $P(s'|s, a)$ with the value $V(s')$ from the previous iteration and applies the **discount factor** λ. These values are summed up to get the discounted expected future reward.
  - Once all $Q(s, a)$ values are calculated for each action, the **maximum** $Q(s, a)$ across all actions is selected, updating $V(s)$ for that state.
  - **Max Delta**: The `max_delta` variable keeps track of the largest change in $V(s)$ values for each iteration. It's used to check if the difference in values has fallen below ϵ, indicating convergence.

4. Termination Check

- At the end of each iteration, if `max_delta < epsilon`, the values have effectively stopped changing, and the loop breaks. This ensures that the values for each state have reached near-optimal values, and further iterations won't significantly improve them.

5. Result

- **Output**: The function prints the optimal values $V^*(S)$ for each state. These values represent the maximum long-term rewards achievable from each state under the optimal policy.

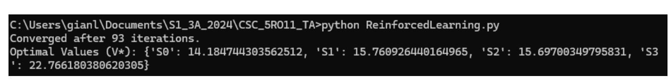

Fig. 1. Result of the Python Script

The results show the outcomes of applying the value iteration algorithm for the specified states after reaching convergence.

1) **Convergence After 93 Iterations:** This means that the algorithm looped 93 times before the values in each state stabilized, reaching the termination condition set by $|V_k(S) - V_{k-1}(S)| < 0.0001$ for each state.

2) **Optimal Values V^*:** The values shown for each state represent the maximum expected cumulative reward achievable from each state under an optimal policy π^*.
   - $V^*(S_0) = 14.18$: Starting in S_0, the agent can achieve a maximum expected reward of approximately 14.18 in the long run by following the optimal policy. Actions here influence reaching other states that yield future rewards.
   - $V^*(S_1) = 15.76$: The value for S_1 is higher than S_0, suggesting that from this state, the optimal policy can yield more favorable rewards.
   - $V^*(S_2) = 15.70$: Starting from S_2, the agent has a similar value to S_1. The presence of a reward in S_2 and transitions to other states help keep its value relatively high.
   - $V^*(S_3) = 22.77$: The highest value is in S_3, primarily due to the reward of 10 obtainable from taking action a_0. The optimal policy has a strong incentive to transition toward S_3 due to this reward, explaining the high value here.