ELEN E6885: Introduction to Reinforcement Learning Homework #2

Chenye Yang cy2540@columbia.edu

October 15, 2019

P1

1.

In non-stationary reinforcement learning problems, we usually use a constant step-size in the incremental update of a reinforcement learning algorithm, e.g., $Q_{k+1} = Q_k + \alpha[r_{k+1} - Q_k]$.
True

2.

To find the optimal policy for a given MDP, one can choose to compute and store state-value function $v(s)$ or action-value function $q(s, a)$. One argument in favor of $q(s, a)$ is that it needs to store fewer values.
False

3.

In the Monte Carlo estimation of action values, if a deterministic policy π is used, in most cases one agent (with the same starting state) could not learn the values of all actions.
True

4.

For a stationary problem, Sarsa converges with probability 1 to an optimal policy and action-value function as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the greedy policy.
True

5.

Both SARSA and Q-learning are on-policy learning algorithms.
False

P2

1.

Ans:
In policy iteration algorithm, between each policy improvement process, the policy evaluation process will do iterative computations of the value functions to let them converge to v_π^*. Sequence of monotonically improving policies and values functions:

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0}^* \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1}^* \xrightarrow{\text{I}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

However, in value iteration algorithm, only a single iteration of policy evaluation is performed between each policy improvement. Sequence of monotonically improving policies and values functions:

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*$$

2.

Ans:
From a given same starting value function, the policy iteration and value iteration will both choose the greedy policy improvement method, and as a result get the same improved policy. Following the same improved policy, the first step of policy evaluation in policy iteration and policy evaluation in value iteration will calculate the same value functions. Because the value iteration only have single iteration of policy evaluation, which is exactly same as the first step of policy evaluation in policy iteration. Therefore, in the question's situation, one iteration of two methods will generate the same value function.

P3

1.

Ans:
Let the state set be $S = \{s_A, s_B, s_C, s_D\}$, action set be $A = \{a_1, a_2, a_3\}$.

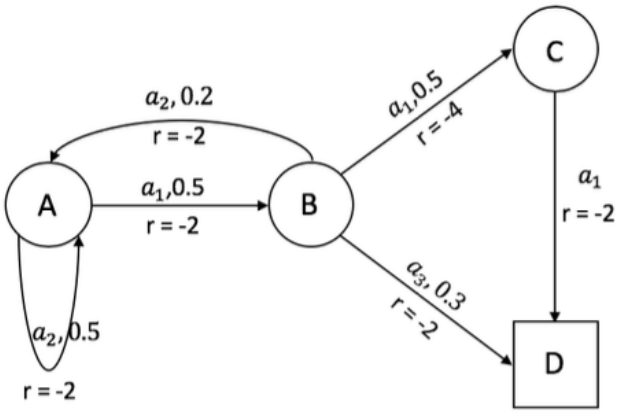

Figure 1: Model-based MDP

The state transition probabilities are:

$$\begin{aligned} &P_{s_A s_A} = 0.5 \quad P_{s_A s_B} = 0.5 \\ &P_{s_B s_A} = 0.2 \quad P_{s_B s_C} = 0.5 \quad P_{s_B s_D} = 0.3 \\ &P_{s_C s_D} = 1 \end{aligned}$$

Thus the state transition probability matrix is:

$$\boldsymbol{P} = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.2 & 0 & 0.5 & 0.3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The reward function are:

$$\begin{aligned} R_{s_A} &= 0.5 \times (-2) + 0.5 \times (-2) = -2 \\ R_{s_B} &= 0.2 \times (-2) + 0.5 \times (-4) + 0.3 \times (-2) = -3 \\ R_{s_C} &= -2 \\ R_{s_D} &= 0 \end{aligned}$$

Thus the reward vector is:

$$\boldsymbol{R} = \begin{bmatrix} -2 \\ -3 \\ -2 \\ 0 \end{bmatrix}$$

When $\gamma = 0.5$, from matrix-form Bellman equation, the value vector is:

$$\begin{aligned}\boldsymbol{v} &= (1-\gamma \boldsymbol{P})^{-1}\boldsymbol{R} \\ &= (1 - 0.5 \times \begin{bmatrix} 0.5 & 0.5 & 0 & 0 \\ 0.2 & 0 & 0.5 & 0.3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix})^{-1} \times \begin{bmatrix} -2 \\ -3 \\ -2 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -3.97 \\ -3.90 \\ -2 \\ 0 \end{bmatrix}\end{aligned}$$

Thus, the values of all states are:

$$\begin{aligned} v_{s_A} &= -3.97 \\ v_{s_B} &= -3.90 \\ v_{s_C} &= -2 \\ v_{s_D} &= 0 \end{aligned}$$

2.

Ans:
In the first step of iterative policy evaluation, $k = 1$:

$$\begin{aligned} v_{s_D} &= 0 \\ v_{s_C} &= -2 \\ v_{s_B} &= 0.5(-4+0) + 0.3(-2+0) + 0.2(-2+0) = -3 \\ v_{s_A} &= 0.5(-2+0) + 0.5(-2+0) = -2 \end{aligned}$$

In the second step of iterative policy evaluation, $k = 2$:

$$\begin{aligned} v_{s_D} &= 0 \\ v_{s_C} &= -2 \\ v_{s_B} &= 0.5(-4+1\times v_{s_C}) + 0.3(-2+1\times v_{s_D}) + 0.2(-2+1\times v_{s_A}) \\ v_{s_A} &= 0.5(-2+1\times v_{s_B}) + 0.5(-2+1\times v_{s_A}) \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} v_{s_D} &= 0 \qquad \text{use values at k=1 to calculate the values at k=2} \\ v_{s_C} &= -2 \\ v_{s_B} &= -4 + 0.2 v_{s_A} \\ v_{s_A} &= -2 + 0.5 v_{s_B} + 0.5 v_{s_A} \end{aligned}$$

$\Rightarrow$

$$\begin{aligned} v_{s_D} &= 0 \\ v_{s_C} &= -2 \\ v_{s_B} &= -6 \\ v_{s_A} &= -10 \end{aligned}$$

3.

Ans:
$k = 3$, the method of value iteration, following the state values in **2.** $k = 2$:
Policy improvement:
Because:

$$\sum_{s',r} p(s',r|s_C,a_1)[r+\gamma v_{s'}] = p(s_D,r|s_C,a_1)[r+\gamma v_{s_D}] = 1\times(-2+0) = -2$$
$$\sum_{s',r} p(s',r|s_B,a_1)[r+\gamma v_{s'}] = p(s_C,r|s_B,a_1)[r+\gamma v_{s_C}] = 1\times(-4-2) = -6$$
$$\sum_{s',r} p(s',r|s_B,a_2)[r+\gamma v_{s'}] = p(s_A,r|s_B,a_2)[r+\gamma v_{s_A}] = 1\times(-2-10) = -12$$
$$\sum_{s',r} p(s',r|s_B,a_3)[r+\gamma v_{s'}] = p(s_D,r|s_B,a_3)[r+\gamma v_{s_D}] = 1\times(-2+0) = -2$$
$$\sum_{s',r} p(s',r|s_A,a_1)[r+\gamma v_{s'}] = p(s_B,r|s_A,a_1)[r+\gamma v_{s_B}] = 1\times(-2-6) = -8$$
$$\sum_{s',r} p(s',r|s_A,a_2)[r+\gamma v_{s'}] = p(s_A,r|s_A,a_2)[r+\gamma v_{s_A}] = 1\times(-2-10) = -12$$

According to Bellman optimality backup in value iteration method

Therefore:

$$\pi(a|s_C) = 1,\ a = a_1$$
$$\pi(a|s_B) = \begin{cases} 1, & a = a_3 \\ 0, & a = a_1, a_2 \end{cases}$$
$$\pi(a|s_A) = \begin{cases} 1, & a = a_1 \\ 0, & a = a_2 \end{cases}$$

Policy evaluation:

$$v_{s_D} = 0$$
$$v_{s_C} = -2$$
$$v_{s_B} = -2$$
$$v_{s_A} = -2-6 = -8$$

Therefore, the values of all states for $k = 3$ using the value iteration method are:

$$v_{s_D} = 0$$
$$v_{s_C} = -2$$
$$v_{s_B} = -2$$
$$v_{s_A} = -8$$

P4

1.

Ans:
Let $\boldsymbol{v}_k$ and $\boldsymbol{v}_{k+1}$ be the value function at step k and $k+1$, $\boldsymbol{\pi}_k$ and $\boldsymbol{\pi}_{k+1}$ be the best policy at step k and $k+1$, in the value iteration algorithm. From the given, we know that $\boldsymbol{\pi}_k$ is extracted using greedy selection based on $\boldsymbol{v}_k$.

If $\boldsymbol{v}_k = \boldsymbol{v}_{k+1}$, using greedy action selection, which means always select the action leading to greatest state-value function, from corresponding one state to the next state in different steps, the agent will have same actions. Thus $\boldsymbol{\pi}_k$ and $\boldsymbol{\pi}_{k+1}$ will be same, because they based on same value function and having same deterministic action selection.

Because in the value iteration algorithm, the next state-value and next policy are always better than former ones, unless when the algorithm converges and gets the optimal state-value function and optimal policy. Therefore, $\boldsymbol{v}_k = \boldsymbol{v}_{k+1} = \boldsymbol{v}_*$, $\boldsymbol{\pi}_k = \boldsymbol{\pi}_{k+1} = \boldsymbol{\pi}_*$, and in the future iteration they will not change.

From another perspective, considering policy and value are interrelated and action selection method is deterministic, in policy improvement same $\boldsymbol{v}$ will lead to same $\boldsymbol{\pi}$, in policy evaluation, same $\boldsymbol{v}_k$ and same $\boldsymbol{\pi}_k$ will lead to same $\boldsymbol{v}_{k+1}$. Therefore, $\boldsymbol{v}_k = \boldsymbol{v}_{k+1}$ and $\boldsymbol{\pi}_k = \boldsymbol{\pi}_{k+1}$ will cause the algorithm runs into a deterministic iteration, and $\boldsymbol{v}$ and $\boldsymbol{\pi}$ don't change.

Therefore, if the value function at step k is the same as that of step $k+1$, the best policy will never change with further iterations of the value function.

2.

Ans:

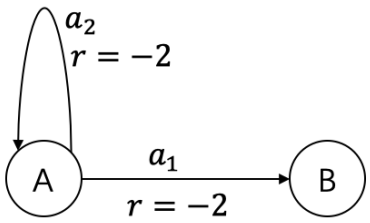

Figure 2: An example MDP

Construct a MDP problem shown in Figure 2, in which the state set is $S = \{s_A, s_B\}$ the action set is $A = \{a_1, a_2\}$ and v_{s_B} is time-varying.
When $k = 1$:

$$\text{assume}: \quad v_{s_A} = -4 \quad v_{s_B} = -3$$
$$\text{improvement}: \quad \pi(a_1|s_A) = 1 \quad \pi(a_2|s_A) = 0$$

When $k = 2$:

$$\begin{aligned} \text{assume}: \quad & v_{s_B} = -4 \\ \text{evaluation}: \quad & v_{s_A} = -5 \\ \text{improvement}: \quad & \pi(a_1|s_A) = 1 \quad \pi(a_2|s_A) = 0 \end{aligned}$$

When $k = 3$:

$$\begin{aligned} \text{assume}: \quad & v_{s_B} = -7 \\ \text{evaluation}: \quad & v_{s_A} = -6 \\ \text{improvement}: \quad & \pi(a_1|s_A) = 0 \quad \pi(a_2|s_A) = 1 \end{aligned}$$

In this example, the best policy at step $k = 1$ is the same as that of step $k = 2$, however the best policy can still change with further iterations of the value function, like at step $k = 3$.

P5

Ans:
Considering the sequence in an episode which has the following format:

$$S_1, A_2, R_2, S_2, ..., R_i, \boldsymbol{S_k, A_k, R_k}, S_m, ..., R_j, \boldsymbol{S_k, A_k, R_k}, S_n$$

In the example sequence, state S_k following action A_k (policy π) appears not only at the middle of the sequence but also at the end of the sequence.S_n is the end state and $m \neq n$. Assume all the episodes have the same format sequence.

According to the last-visit Monte Carlo method, considering $v_{S_n} = 0$, the estimated value of state S_k will always be $v_{S_k} = R_k$, which is not the real value of state S_k following policy π, no matter how many episodes are calculated.

Therefore, the last-visit Monte Carlo method for estimating v_π is not guaranteed to converge to v_π for a finite MDP with bounded rewards and $\gamma \in [0, 1)$.

P6

1.

Ans:
For the initial policy, we assume that agent randomly choose an action from action set $A = \{up, down, right, left\}$, that is to say, when agent is at a state $s \in S = \{(1,1),(1,2),(2,1),(2,2),(1,3),(2,3)\}$ it can go up, down, right, left with same probability at 0.25.
$k = 1$
Policy evaluation:

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'|s') q_\pi(s', a')$$

Using Matlab to help the calculation, code as below:

```
% Matlab
syms q_2_2_up q_2_2_down q_2_2_right q_2_2_left
syms q_2_1_up q_2_1_down q_2_1_right q_2_1_left
syms q_1_2_up q_1_2_down q_1_2_right q_1_2_left
syms q_1_1_up q_1_1_down q_1_1_right q_1_1_left
[q_2_2_up,q_2_2_down,q_2_2_right,q_2_2_left,q_2_1_up,q_2_1_down,
    q_2_1_right,q_2_1_left,q_1_2_up,q_1_2_down,q_1_2_right,q_1_2_left,
    q_1_1_up,q_1_1_down,q_1_1_right,q_1_1_left]=...
    solve(q_2_2_right==0+1*(0.8*(5+0)+0.1*0.25*(q_2_2_up+q_2_2_down+
        q_2_2_right+q_2_2_left)+0.1*0.25*(q_1_2_up+q_1_2_down+
        q_1_2_right+q_1_2_left)),...
    q_2_2_up==0+1*(0.8*0.25*(q_2_2_up+q_2_2_down+q_2_2_right+
        q_2_2_left)+0.1*(5+0)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right
        +q_2_1_left)),...
    q_2_2_down==0+1*(0.8*0.25*(q_1_2_up+q_1_2_down+q_1_2_right+
        q_1_2_left)+0.1*(5+0)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right
        +q_2_1_left)),...
    q_2_2_left==0+1*(0.8*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
        q_2_1_left)+0.1*0.25*(q_2_2_up+q_2_2_down+q_2_2_right+
        q_2_2_left)+0.1*0.25*(q_1_2_up+q_1_2_down+q_1_2_right+
        q_1_2_left)),...
    ...
    q_2_1_right==0+1*(0.8*0.25*(q_2_2_up+q_2_2_down+q_2_2_right+
        q_2_2_left)+0.1*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
        q_1_1_left)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
        q_2_1_left)),...
    q_2_1_up==0+1*(0.8*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
        q_2_1_left)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
        q_2_1_left)+0.1*0.25*(q_2_2_up+q_2_2_down+q_2_2_right+
        q_2_2_left)),...
    q_2_1_down==0+1*(0.8*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
        q_1_1_left)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
        q_2_1_left)+0.1*0.25*(q_2_2_up+q_2_2_down+q_2_2_right+
```

```
    q_2_2_left)) ,...
15  q_2_1_left==0+1*(0.8*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
    q_2_1_left)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
    q_2_1_left)+0.1*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)) ,...
16  ...
17  q_1_2_right==0+1*(0.8*(-5+0)+0.1*0.25*(q_1_2_up+q_1_2_down+
    q_1_2_right+q_1_2_left)+0.1*0.25*(q_2_2_up+q_2_2_down+
    q_2_2_right+q_2_2_left)) ,...
18  q_1_2_up==0+1*(0.8*0.25*(q_2_2_up+q_2_2_down+q_2_2_right+
    q_2_2_left)+0.1*(-5+0)+0.1*0.25*(q_1_1_up+q_1_1_down+
    q_1_1_right+q_1_1_left)) ,...
19  q_1_2_down==0+1*(0.8*0.25*(q_1_2_up+q_1_2_down+q_1_2_right+
    q_1_2_left)+0.1*(-5+0)+0.1*0.25*(q_1_1_up+q_1_1_down+
    q_1_1_right+q_1_1_left)) ,...
20  q_1_2_left==0+1*(0.8*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)+0.1*0.25*(q_2_2_up+q_2_2_down+q_2_2_right+
    q_2_2_left)+0.1*0.25*(q_1_2_up+q_1_2_down+q_1_2_right+
    q_1_2_left)) ,...
21  ...
22  q_1_1_right==0+1*(0.8*0.25*(q_1_2_up+q_1_2_down+q_1_2_right+
    q_1_2_left)+0.1*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
    q_2_1_left)) ,...
23  q_1_1_up==0+1*(0.8*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
    q_2_1_left)+0.1*0.25*(q_1_2_up+q_1_2_down+q_1_2_right+
    q_1_2_left)+0.1*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)) ,...
24  q_1_1_down==0+1*(0.8*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)+0.1*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)+0.1*0.25*(q_1_2_up+q_1_2_down+q_1_2_right+
    q_1_2_left)) ,...
25  q_1_1_left==0+1*(0.8*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)+0.1*0.25*(q_1_1_up+q_1_1_down+q_1_1_right+
    q_1_1_left)+0.1*0.25*(q_2_1_up+q_2_1_down+q_2_1_right+
    q_2_1_left)) ,...
26  q_2_2_up,q_2_2_down,q_2_2_right,q_2_2_left,q_2_1_up,q_2_1_down,
    q_2_1_right,q_2_1_left,q_1_2_up,q_1_2_down,q_1_2_right,
    q_1_2_left,q_1_1_up,q_1_1_down,q_1_1_right,q_1_1_left);
```

The solution is:

$q((2,2),up) = 18/11 \quad q((2,2),down) = -6/11 \quad q((2,2),right) = 4 \quad q((2,2),left) = 4/11$
$q((2,1),up) = 6/11 \quad q((2,1),down) = -2/11 \quad q((2,1),right) = 12/11 \quad q((2,1),left) = 4/11$
$q((1,2),up) = 6/11 \quad q((1,2),down) = -18/11 \quad q((1,2),right) = -4 \quad q((1,2),left) = -4/11$
$q((1,1),up) = 2/11 \quad q((1,1),down) = -6/11 \quad q((1,1),right) = -12/11 \quad q((1,1),left) = -4/11$

Policy improvement:

$$\pi(a|(2,2)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases} \quad \pi(a|(2,1)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases}$$
$$\pi(a|(1,2)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases} \quad \pi(a|(1,1)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases}$$

$k = 2$
Policy evaluation, using Matlab;

$q((2,2), up) = 419/110 \quad q((2,2), down) = 23/22 \quad q((2,2), right) = 49/11 \quad q((2,2), left) = 73/55$
$q((2,1), up) = 76/55 \quad q((2,1), down) = 36/55 \quad q((2,1), right) = 183/55 \quad q((2,1), left) = 1$
$q((1,2), up) = 299/110 \quad q((1,2), down) = -1/22 \quad q((1,2), right) = -39/11 \quad q((1,2), left) = 3/5$
$q((1,1), up) = 52/55 \quad q((1,1), down) = 12/55 \quad q((1,1), right) = 31/55 \quad q((1,1), left) = 3/11$

Policy improvement:

$$\pi(a|(2,2)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases} \quad \pi(a|(2,1)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases}$$
$$\pi(a|(1,2)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases} \quad \pi(a|(1,1)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases}$$

$k = 3$
Policy evaluation, using Matlab;

$q((2,2), up) = 1209/275 \; q((2,2), down) = 827/275 \; q((2,2), right) = 5189/1100 \; q((2,2), left) = 3717/1100$
$q((2,1), up) = 86/25 \; q((2,1), down) = 422/275 \; q((2,1), right) = 439/110 \; q((2,1), left) = 1699/550$
$q((1,2), up) = 1737/550 \; q((1,2), down) = 973/550 \; q((1,2), right) = -3611/1100 \; q((1,2), left) = 1621/1100$
$q((1,1), up) = 3331/1100 \; q((1,1), down) = 247/220 \; q((1,1), right) = 1431/550 \; q((1,1), left) = 651/550$

Policy improvement:

$$\pi(a|(2,2)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases} \quad \pi(a|(2,1)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases}$$
$$\pi(a|(1,2)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases} \quad \pi(a|(1,1)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases}$$

When $k = 5$, policy will change to:

$$\pi(a|(2,2)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases} \quad \pi(a|(2,1)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases}$$
$$\pi(a|(1,2)) = \begin{cases} 1, & a = left \\ 0, & else \end{cases} \quad \pi(a|(1,1)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases}$$

Then the policy reaches the optimal policy and will keep unchanged in next iterations. Therefore, the optimal policy is as follow and shown in Figure 3:

$$\pi_*(a|(2,2)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases} \quad \pi_*(a|(2,1)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases}$$
$$\pi_*(a|(1,2)) = \begin{cases} 1, & a = left \\ 0, & else \end{cases} \quad \pi_*(a|(1,1)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases}$$

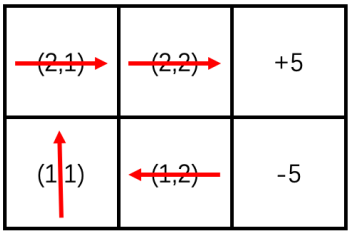

Figure 3: Optimal policy

2.

Ans:
With the optimal policy from **1.**, the transition probabilities:

$$P_{(1,1),(2,1)} = 0.8 \quad P_{(1,1),(1,1)} = 0.1 \quad P_{(1,1),(1,2)} = 0.1$$
$$P_{(2,1),(2,2)} = 0.8 \quad P_{(2,1),(2,1)} = 0.1 \quad P_{(2,1),(1,1)} = 0.1$$
$$P_{(1,2),(1,1)} = 0.8 \quad P_{(1,2),(2,2)} = 0.1 \quad P_{(1,2),(1,2)} = 0.1$$
$$P_{(2,2),(2,3)} = 0.8 \quad P_{(2,2),(1,2)} = 0.1 \quad P_{(2,2),(2,2)} = 0.1$$

The **first** round of value iteration:
Policy evaluation:

$$v_{(1,1)} = 0 + 0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0 = 0$$
$$v_{(1,2)} = 0 + 0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0 = 0$$
$$v_{(2,1)} = 0 + 0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0 = 0$$
$$v_{(2,2)} = 0 + 0.8 \times 5 + 0.1 \times 0 + 0.1 \times 0 = 4$$

Policy improvement:

$$\pi(a|(2,2)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases} \quad \pi(a|(2,1)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases}$$
$$\pi(a|(1,2)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases} \quad \pi(a|(1,1)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases}$$

The transition probabilities:

$$\begin{aligned}
&P_{(1,1),(2,1)} = 0.8 \quad P_{(1,1),(1,1)} = 0.1 \quad P_{(1,1),(1,2)} = 0.1 \\
&P_{(2,1),(2,2)} = 0.8 \quad P_{(2,1),(2,1)} = 0.1 \quad P_{(2,1),(1,1)} = 0.1 \\
&P_{(1,2),(2,2)} = 0.8 \quad P_{(1,2),(1,1)} = 0.1 \quad P_{(1,2),(1,3)} = 0.1 \\
&P_{(2,2),(2,3)} = 0.8 \quad P_{(2,2),(1,2)} = 0.1 \quad P_{(2,2),(2,2)} = 0.1
\end{aligned}$$

The **second** round of value iteration:
Policy evaluation:

$$\begin{aligned}
v_{(1,1)} &= 0 + 0.8 \times 0 + 0.1 \times 0 + 0.1 \times 0 = 0 \\
v_{(1,2)} &= 0 + 0.8 \times 0.9 \times 4 + 0.1 \times 0 + 0.1 \times (-5) = 2.38 \\
v_{(2,1)} &= 0 + 0.8 \times 0.9 \times 4 + 0.1 \times 0 + 0.1 \times 0 = 2.88 \\
v_{(2,2)} &= 0 + 0.8 \times 5 + 0.1 \times 0.9 \times 4 + 0.1 \times 0 = 4.36
\end{aligned}$$

Policy improvement:

$$\pi(a|(2,2)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases} \quad \pi(a|(2,1)) = \begin{cases} 1, & a = right \\ 0, & else \end{cases}$$

$$\pi(a|(1,2)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases} \quad \pi(a|(1,1)) = \begin{cases} 1, & a = up \\ 0, & else \end{cases}$$

The transition probabilities:

$$\begin{aligned}
&P_{(1,1),(2,1)} = 0.8 \quad P_{(1,1),(1,1)} = 0.1 \quad P_{(1,1),(1,2)} = 0.1 \\
&P_{(2,1),(2,2)} = 0.8 \quad P_{(2,1),(2,1)} = 0.1 \quad P_{(2,1),(1,1)} = 0.1 \\
&P_{(1,2),(2,2)} = 0.8 \quad P_{(1,2),(1,1)} = 0.1 \quad P_{(1,2),(1,3)} = 0.1 \\
&P_{(2,2),(2,3)} = 0.8 \quad P_{(2,2),(1,2)} = 0.1 \quad P_{(2,2),(2,2)} = 0.1
\end{aligned}$$

3.

Ans:
Traces:

$$(1,1) - (1,2) - (1,3) \quad (1,1) - (1,2) - (2,2) - (2,3) \quad (1,1) - (2,1) - (2,2) - (2,3)$$

Monte Carlo estimates for states $(1,1)$, $\gamma = 1$:

$$\begin{aligned}
v_{(1,1)} &= \frac{[0 + 1 \times (0 + 1 \times (-5))] + [0 + 1 \times (0 + 1 \times (0 + 1 \times 5))] + [0 + 1 \times (0 + 1 \times (0 + 1 \times 5))]}{3} \\
&= 5/3
\end{aligned}$$

Monte Carlo estimates for states $(2,2)$, $\gamma = 1$:

$$\begin{aligned}
v_{(2,2)} &= \frac{(5 + 1 \times 0) + (5 + 1 \times 0)}{2} \\
&= 5
\end{aligned}$$

4.

Ans:
Learning rate of $\alpha = 0.1$, a discount factor of $\gamma = 0.9$, and assuming initial $V_0 = 0$. Trials 1) is $(1,1) - (1,2) - (1,3)$, and 2) is $(1,1) - (1,2) - (2,2) - (2,3)$.

TD(0)-learning updates;
Step 1: $v_{(1,1)} = 0 + 0.1 \times (0 + 0.9 \times 0 - 0) = 0$
Step 2: $v_{(1,2)} = 0 + 0.1 \times (-5 + 0.9 \times 0 - 0) = -0.5$
Step 3: $v_{(1,3)} = 0 + 0.1 \times (0 + 0.9 \times 0 - 0) = 0$
Step 4: $v_{(1,1)} = 0 + 0.1 \times (0 + 0.9 \times (-0.5) - 0) = -0.045$
Step 5: $v_{(1,2)} = -0.5 + 0.1 \times (0 + 0.9 \times 0 - (-0.5)) = -0.45$
Step 6: $v_{(2,2)} = 0 + 0.1 \times (5 + 0.9 \times 0 - 0) = 0.5$
Step 7: $v_{(2,3)} = 0 + 0.1 \times (0 + 0.9 \times 0 - 0) = 0$

Therefore, TD(0)-learning agent makes updates as follow:

$$
\begin{aligned}
v_{(1,1)} &= -0.045 \\
v_{(1,2)} &= -0.45 \\
v_{(2,1)} &= 0 \\
v_{(2,2)} &= 0.5 \\
v_{(1,3)} &= 0 \\
v_{(2,3)} &= 0
\end{aligned}
$$