Exercise 2
Dynamic Programming

1 Policy and Value Iteration

In this exercise you will implement important dynamic programming approaches by yourself. You will use them to solve a very simple gridworld MDP example as shown in Figure 1. To make things easier you can find algorithm outlines in Figures 2, 3 and 4.

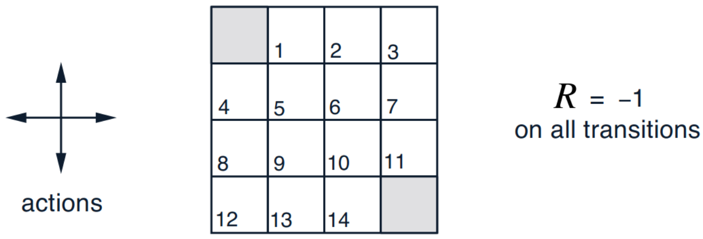

Figure 1: A gridworld MDP. Upper left and lower right states are terminal states.

Programming Tasks:

1. **Policy evaluation (one step)** Fill in code inside `policy_evaluation_one_step`, which performs just one step of policy evaluation (the inner part of the loop inside Figure 2).
2. **Policy evaluation** Using `policy_evaluation_one_step` implement the method `policy_evaluation`, which iteratively performs one step of policy evaluation until the change in value function is smaller then some threshold.
3. **Policy improvement** Program the method `policy_improvement` which returns the optimal greedy policy w.r.t a given value function inside `policy_improvement`.
4. Using methods programmed in 1. - 3. implement:
   (a) the **policy iteration** (PI) algorithm (Figure 3) inside `policy_iteration`
   (b) the **value iteration** (VI) algorithm (Figure 4) inside `value_iteration` (*Hint:* You can either do this *easy* way and reuse methods of 1. - 3. or do it the *efficient* way by following the algorithm outline in Figure 4).

2 Optimal Policies

Try to answer the following questions regarding optimal policies.

Questions:

1. Imagine the optimal state-value function V of a MDP is given to you but you don't have the state transition probabilities P. Could you derive the optimal policy from V?
2. What would happen if you would have access to the optimal action-value function Q instead?
3. Can there be multiple optimal deterministic policies for one MDP? If so, explain the conditions under which this is the case.

Input π, the policy to be evaluated
Initialize an array $V(s) = 0$, for all $s \in \mathcal{S}^+$
Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)
Output $V \approx v_\pi$

Figure 2: Policy evaluation.

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
       $\Delta \leftarrow 0$
       For each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
       $a \leftarrow \pi(s)$
       $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
       If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ $false$
   If *policy-stable*, then stop and return V and π; else go to 2

Figure 3: Policy iteration.

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
    $\Delta \leftarrow 0$
    For each $s \in \mathcal{S}$:
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
    $\pi(s) = \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Figure 4: Value iteration.