**MVA: Reinforcement Learning (2020/2021)** **Homework 1**

# Dynamic Programming

Lecturers: *A. Lazaric, M. Pirotta* *(November 8, 2020)*

Solution by Raphael Reme

**Instructions**

- The deadline is **November 8, 2020. 23h00**
- By doing this homework you agree to the *late day policy, collaboration and misconduct rules* reported on Piazza.
- **Mysterious or unsupported answers will not receive full credit**. A correct answer, unsupported by calculations, explanation, or algebraic work will receive no credit; an incorrect answer supported by substantially correct calculations and explanations might still receive partial credit.
- Answers should be provided in **English**.

## 1 Question

Consider the following grid environment. The agent can move up, down, left and right. Transitions are deterministic. Attempts to move in the direction of the wall will result in staying in the same position. There are two absorbing states: 1 and 14. Taking any action in 1 (resp 14) leads to a reward $r_r$ (resp. $r_g$) ($r(1, a) = r_r, r(14, a) = r_g, \forall a$) and *ends the episode*. Everywhere else the reward is $r_s$. Assume discount factor $\gamma = 1$, $r_g = 10$ and $r_r = -10$, unless otherwise specified.

| | | | |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 |

1. Define $r_s$ such that the optimal policy is the shortest path to state 14. Using the chosen $r_s$, report the value function of the optimal policy for each state.
   There is a simple solution that doesn't require complex computation. You can copy the image an replace the id of the state with the value function.

2. Consider a general MDP with rewards, and transitions. Consider a discount factor of $\gamma < 1$. For this case assume that the horizon is infinite (so there is no termination). A policy $\pi$ in this MDP induces a value function $V^\pi$. Suppose an affine transformation is applied to the reward, what is the new value function? Is the optimal policy preserved?

3. Consider the same setting as in question 1. Assume we modify the reward function with an additive term $c = 5$ (i.e., $r_s = r_s + c$). How does the optimal policy change (just give a one or two sentence description)? What is the new value function?

## Answers

### 1.1

Given a determistic $\pi$, as the environment is also deterministic, the sequence of states (And the trajectory: $(s_t, r_t)_t$) obtained is deterministic and can be infinite if it never reaches the states 1 or 14. There are therefore 14 different trajectories given this $\pi$: one for each starting state. And $V^\pi(s)$ is the sum of the rewards obtain following the sole trajectory generated by $\pi$ from s.
If $s \in \{1, 14\}$ then the trajectory is independant of $\pi$, it always directly terminates with a reward of $\pm 10$. $V^\pi(1) = -10$ and $V^\pi(14) = 10$. If $s \notin \{1, 14\}$ then there are two possibilities depending on $\pi$. Either the trajectory is finite, either it is not (it is then cycling through some other states).
Now let's consider some values for $r_s$:

- $r_s > 0$. Then any optimal policy will lead to an infinite reward for all the states $s \notin \{1, 14\}$ thanks to an infinite walks in the maze: For instance the policy that for any state output the action of going through the neighboring wall (One can notice all the states have a neighboring wall, therefore this policy exists) and leads to infinite trajectory for all $s_0$: $(s_0, r_s)_t$ and $V^\pi(s_0) = \sum_t r_s = +\infty$
- $r_s = 0$. Here it's different. For any state $s \notin \{1, 14\}$ the best that can be done is reaching the reward of the state 14. Any optimal policy will leads to 14 but it is not mandatory that it takes the shortest path. For instance the policy $\pi$ such that $\pi(10) =$ left, $\pi(9) =$ down, $\pi(13) =$ left and so that from any other state it leads to 10, 9 or 13 in a finite list of actions. Then this policy is optimal: the trajectory generated by $\pi$ from s leads to 14 and $V^\pi(s) = 10$. But clearly for the state 10 it's not taking the shortest path to 14.
- $r_s < 0$ Here one can see that now any optimal policy should leads to finite trajectory. As an infinite trajectory will leads to $-\infty$ for the value function. Each trajectory will therefore converge either to 1 or 14. Moreover if the trajectory doesn't take the shortest path, the policy is not optimal as the shortest path will always leads to less negative rewards and should always be the trajectory generated by the optimal policy.

  We just need to ensure that the terminal state will always be 14 for all states ($\neq 1$). (For instance $r_s = -50$, from state 2 the best action to take is left in order to finish in state 1 and have a final -10 reward.) Therefore we want to have that from all states it's worh to go up to the state 14 which can be stated as: $\forall s \notin \{1, 14\}$, $r_s \times d(s, 14) + 10 > r_s \times d(s, 1) - 10$ (with $(d(s, s')$ is the shortest trajectory/path from $s$ to $s'$))

  Resolution: $\forall s \notin \{1, 14\}$, $r_s \times (d(s, 14) - d(s, 1)) > -20$

  $\Leftrightarrow \forall s \notin \{1, 14\}$, such as $d(s, 1) \neq d(s, 14)$, $r_s > \frac{-20}{d(s,14)-d(s,1)}$

  $\Leftrightarrow r_s > \frac{-20}{\max_{s \neq 1}(d(s,14)-d(s,1))}$

  $\Leftrightarrow r_s > \frac{-20}{d(2,14)-d(2,1)}$

  $\Leftrightarrow r_s > \frac{-20}{3-1}$

  $\Leftrightarrow r_s > -10$

To sum up: any $r_s$ in $]-10, 0[$ will leads to an optimal policy that will generate for all states the shortest trajectory from s to 14. (Except for state 1 where the episode ends immediately.)
Let's have for instance $r_s = -1$ then the value function can be represented as:

| | | | |
|---|---|---|---|
| 5 | -10 | 7 | 6 |
| 6 | 7 | 8 | 5 |
| 7 | 8 | 9 | 4 |
| 8 | 9 | 10 | 3 |

### 1.2

Let $\pi$ a policy and $r$ a reward function. We have $V^\pi(s) = \mathbb{E}_{s_t}[\sum_{t=0}^{+\infty} \gamma^t r(s_t, \pi_t(s_t))|s_0 = s]$. Where $(s_t)_t$ is a random sequence of state implied by the policy and the environment.
Let $r'(s,a) = \alpha r(s,a) + \beta$. Then with this same policy we would have

$$V'^\pi(s) = \mathbb{E}_{s_t}[\sum_{t=0}^{+\infty} \gamma^t r'(s_t, \pi_t(s_t))|s_0 = s]$$

$$V'^\pi(s) = \mathbb{E}_{s_t}[\alpha \sum_{t=0}^{+\infty} \gamma^t r(s_t, \pi_t(s_t)) + \frac{\beta}{1-\gamma}|s_0 = s]$$

$$V'^\pi(s) = \alpha \mathbb{E}_{s_t}[\sum_{t=0}^{+\infty} \gamma^t r(s_t, \pi_t(s_t))|s_0 = s] + \frac{\beta}{1-\gamma}$$

$$V'^\pi(s) = \alpha V^\pi(s) + \frac{\beta}{1-\gamma}$$

Now assume that $\pi^\star$ was an optimal policy for $r$. Then $\forall \pi$, $V^{\pi^\star} \geq V^\pi$. We have then $\alpha \geq 0 \Leftrightarrow V'^{\pi^\star} \geq V'^\pi$! Meaning that $\pi^\star$ is an optimal policy for $r'$ if and only if $\alpha \geq 0$ (If $\alpha = 0$ then all policy are equivalent).

### 1.3

With $r'_s = r_s + 5 = 4$ (And in order to have a full afine transformation we could also set $r'_r = r_r + 5 = -5$ and $r'_g = r_g + 5 = 15$ but that don't really matter) as said in 1.1, the optimal policies are those that induce infinite walks for any starting states $\notin \{1, 14\}$. And the maximal value function is therefore $+\infty$ for those states.
Note that the previous result does not hold in this case!

## 2 Question

Consider infinite-horizon $\gamma$-discounted Markov Decision Processes with $S$ states and $A$ actions. Denote by $Q^\star$ the Q-function of the optimal policy $\pi^\star$. Prove that, for any function $Q(s,a)$, the following inequality holds for any $s$

$$V^{\pi_Q}(s) \geq V^\star(s) - \frac{2\|Q^\star - Q\|_\infty}{1-\gamma}$$

where $e = (1, \dots, 1)$, $\|Q^\star - Q\|_\infty = \max_{s,a} |Q^\star(s,a) - Q(s,a)|$ and $\pi_Q(s) = \arg\max_a Q(s,a)$. Thus $\pi^\star(s) = \arg\max_a Q^\star(s,a)$.

### Answer

Let $s$ and $a$ any state and action.

$$V^\star(s) - V^{\pi_Q}(s) = V^\star(s) - \mathcal{T}^{\pi_Q} V^{\pi_Q}(s)$$

$$V^\star(s) - V^{\pi_Q}(s) = V^\star(s) - \mathcal{T}^{\pi_Q} V^\star(s) + \mathcal{T}^{\pi_Q} V^\star(s) - \mathcal{T}^{\pi_Q} V^{\pi_Q}(s)$$

By definition $Q^\star(s, \pi_Q(s)) = r(s, \pi_Q) + \sum_{s'} p(s'|s, \pi_q(s)) V^\star(s') = \mathcal{T}^{\pi_Q} V^\star(s)$
And $V^\star(s) = Q^\star(s, \pi^\star(s))$. Therefore:

$$V^\star(s) - V^{\pi_Q}(s) = Q^\star(s, \pi^\star(s)) - Q^\star(s, \pi_Q(s)) + \mathcal{T}^{\pi_Q} V^\star(s) - \mathcal{T}^{\pi_Q} V^{\pi_Q}(s)$$

$$|V^\star(s) - V^{\pi_Q}(s)| \leq |Q^\star(s, \pi^\star(s)) - Q^\star(s, \pi_Q(s))| + |\mathcal{T}^{\pi_Q} V^\star(s) - \mathcal{T}^{\pi_Q} V^{\pi_Q}(s)|$$

$$|V^\star(s) - V^{\pi_Q}(s)| \leq |Q^\star(s, \pi^\star(s)) - Q(s,a) + Q(s,a) - Q^\star(s, \pi_Q(s))| + |\mathcal{T}^{\pi_Q} V^\star(s) - \mathcal{T}^{\pi_Q} V^{\pi_Q}(s)|$$

$$|V^\star(s) - V^{\pi_Q}(s)| \leq |Q^\star(s, \pi^\star(s)) - Q(s,a)| + |Q(s,a) - Q^\star(s, \pi_Q(s))| + |\mathcal{T}^{\pi_Q} V^\star(s) - \mathcal{T}^{\pi_Q} V^{\pi_Q}(s)|$$

$$|V^\star(s) - V^{\pi_Q}(s)| \leq 2||Q^\star - Q||_\infty + ||\mathcal{T}^{\pi_Q}V^\star - \mathcal{T}^{\pi_Q}V^{\pi_Q}||_\infty$$
$$|V^\star(s) - V^{\pi_Q}(s)| \leq 2||Q^\star - Q||_\infty + \gamma||V^\star - V^{\pi_Q}||_\infty \text{ (By contraction of the bellman operator)}$$
$$||V^\star - V^{\pi_Q}||_\infty \leq 2||Q^\star - Q||_\infty + \gamma||V^\star - V^{\pi_Q}||_\infty$$
$$||V^\star - V^{\pi_Q}||_\infty \leq \frac{2||Q^\star - Q||_\infty}{1-\gamma}$$

Therefore $\forall s$ we have $V^\star(s) - V^{\pi_Q}(s) = |V^\star(s) - V^{\pi_Q}(s)| \leq ||V^\star - V^{\pi_Q}||_\infty \leq \frac{2||Q^\star - Q||_\infty}{1-\gamma}$

$$\Rightarrow V^{\pi_Q}(s) \geq V^\star(s) - \frac{2||Q^\star - Q||_\infty}{1-\gamma}$$

# 3 Question

Consider the average reward setting ($\gamma = 1$) and a Markov Decision Process with $S$ states and $A$ actions. Prove that

$$g^{\pi'} - g^\pi = \sum_s \mu^{\pi'}(s) \sum_a (\pi'(a|s) - \pi(a|s))Q^\pi(s,a) \tag{1}$$

using the fact that in average reward the Bellman equation is

$$Q^\pi(s,a) = r(s,a) - g^\pi + \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')Q^\pi(s',a'), \quad \forall s,a,\pi$$

and $\mu^\pi$ is the **stationary distribution** of policy $\pi$. Note also that $g^\pi = \sum_s \mu^\pi(s) \sum_a \pi(a|s)r(s,a)$.

*Note: All the information provided to prove Eq. 1 are mentioned in the question. Start from the definition of $Q$ and use the property of stationary distribution.*

## Answer

We have $g^{\pi'} = \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s)r(s,a)$.
It's then possible to write $r(s,a) = Q^\pi(s,a) + g^\pi - \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')Q^\pi(s',a')$

$$g^{\pi'} = \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s)[Q^\pi(s,a) + g^\pi - \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')Q^\pi(s',a')]$$

$$\begin{aligned} g^{\pi'} = g^\pi \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s) + \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s)Q^\pi(s,a) \\ - \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')Q^\pi(s',a') \end{aligned}$$

As $\sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s) = 1$ we have:

$$\begin{aligned} g^{\pi'} - g^\pi = \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s)Q^\pi(s,a) \\ - \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a'|s')Q^\pi(s',a') \end{aligned}$$

$$\begin{aligned} g^{\pi'} - g^\pi = \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s)Q^\pi(s,a) \\ - \sum_{s,a,s'} \mu^{\pi'}(s)\pi'(a|s)p(s'|s,a) \sum_{a'} \pi(a'|s')Q^\pi(s',a') \end{aligned}$$

By definition of $\mu^{\pi'}$: $\mu^{\pi'}(s') = \sum_s \mu(s)\mathbb{P}'(s'|s) = \sum_{s,a} \mu(s)\mathbb{P}'(s', a|s) = \sum_{s,a} \mu(s)p(s'|s, a)\pi'(a|s)$. Therefore we have:

$$\begin{aligned} g^{\pi'} - g^{\pi} &= \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s) Q^{\pi}(s, a) \\ &\quad - \sum_{s'} \mu^{\pi'}(s') \sum_{a'} \pi(a'|s') Q^{\pi}(s', a') \\ g^{\pi'} - g^{\pi} &= \sum_s \mu^{\pi'}(s) \sum_a (\pi'(a|s) - \pi(a|s)) Q^{\pi}(s, a) \end{aligned}$$

# 4 Question

Provide an MDP modeling, specifying all its defining elements, of the following process:

- Elevator dispatching. The elevator controllers assign elevators to service passenger requests in real-time while optimizing the overall service quality, e.g. by minimizing the waiting time and/or the energy consumption. The agent can simultaneously control all the elevators. In order to model this problem, consider a 6-story building with 2 elevators. Explain the choices made for modeling this problem.

## Answer

### Assumptions

One big assumption that should be done in order to simplify the problem is time discretisation. Even if the users demands are rather continuous, the time discrtisation will only lead on an approximation of the user waiting time with an error smaller than the discretisation step. This is acceptable if the discretisation step is small enough.
To simplify even more we can state that an elevator should never change of direction when it's moving. And that it can only stop at a precise floor. This leads to a discretisation of states: we should only consider integer positions for the elevators (As it's always moving if it's at a non-integer position and will continue on the same direction at least up to the next integer position).
This is related to time discretisation. Let's assume that elevators have constant speed, let's call T the moving time from two neighboring floor: T could be our time step. (We could also try to have an non constant time discretisation by taking a new step each time a new demand arrives or an elevator stops at a floor which would be an events-based model.)
Another simplification that I would do is that elevators have no size limit. It can transport as many users as possible. And I won't try to model the number of people inside the elevator.

### Modelisation

Actions are easy to model. For each elevator there is only 3 actions possible at each step: Do nothing, moving up, moving down. It can be resprented with $A = \{0, 1, -1\}^2$. If $(a_1, a_2) \in A$, $a_1$ is action for the elevator 1 and $a_2$ for elevator 2.
With elevators position discretise we can consider the set of positions for the two elevators: $E = [|1, 6|]^2$
And we can model the users waiting for an elevator by $W = \{0, 1\}^6$. (It can be as many people waiting. And any of the two elevator can take the waiting users.) For any $w \in W, w_i = 1$ means that an elevator is required at the $i^{\text{th}}$ floor.
And finally there are floors requested by users for each elevator that could be model with: $F = \{0, 1\}^{6^2}$. For any $(f_1, f_2) \in F$, having $f_{1_i} = 1$ (resp. $f_{2_i}$) means that elevator 1 (resp. 2) has been asked to go to the $i^{\text{th}}$ floor by an user.
($W$ represents outside buttons used to call elevators. $F$ represents inside buttons that users uses to go to a specific floor)
We can now model states as follow: $s = (e, w, f_1, f_2)$ with $e \in E, w \in W, (f_1, f_2) \in F$.
$S = E \times W \times F$

### Environment modelisation

At each time step, an action $a_t = a$ is chosen by a policy $\pi$ from the state $s_t = (e, w, f)$. (Could also consider history dependant/sotchastic policy)
The environment induces a new state $s_{t+1} = (e', w', f')$ according to the probability that we can simplify $p(e', w', f'|e, w, f, a) = p(e'|e, a) \times p(w'|e, w) \times p(f'_1|e, f_1) \times p(f'_2|e, f_2)$: With easy assumptions that the next elevator positions only depends on the previous one and the action and that the users requests are independant.
We could/should add some constraints on these laws. Here are some example:

- $p(w'|e, w) = \prod_i p(w'_i|e, w_i)$: The waiting users at a floor is independant from the waiting users of the other floors. (Note that if for $w$ this assumptions seems reasonable. It doesn't seems the case for $f_1$ and $f_2$. But we could also suppose it.)
- $p(w'_i = 0|e, w_i) = 1$ if $i \in e$: When an elevator reaches a floor, the users waiting at that floor are all taken.
- $p(f'_{k_i} = 0|e, f_k) = 1$ if $i \in e$: When an elevator reaches a floor, the request for this floor is reset.
- $p(w'_i = 0|e, w_i = 1) = 0$ if $i \notin e$: The demand does not resume by itself. (users won't take the stairs)

We could/should add much more assumptions. (Deterministic behavior of elevators when an action is taken. Non concellable requests. Etc.)

### Rewards

The reward function can be design quite simply now:
$r(e, w, f, a) = -(||f||_1 + \alpha||w||_1 + \beta||a||_1)$. Which simply state elevators should bring people at the right floors ($||f||_1$ represents the number of floors still requested in both elevators, with the correct environment modelisation, the only way for having it down would be to go to these floors). Elevator should also reduce the people waiting for elevators. ($||w||_1$) And finally it should reduce its actions. (Moving one way or another cost the same amount which is greater than doing nothing.).
$\alpha, \beta$ parameters should be correctly choosen so that the elevators focus on what is important. (I would take $\beta < \alpha < 1$: First focus on current user in the elevators, then focus on waiting user. And finally focus on minimising fuel costs. If $\beta$ is too large the optimal strategy would be staying still...)

# 5 Question

Implement value iteration and policy iteration. We have provided a custom environment in the starter code.

1. (coding) Consider the provided code `vipi.py` and implement `policy_iteration`. Use $\gamma$ as provided by `env.gamma` and the terminal condition seen in class. Return the optimal value function and the optimal policy.
2. (coding) Implement `value_iteration` in `vipi.py`. The terminal condition is based on $\|V_{new} - V_{old}\|_\infty$ and the tollerance is $10^{-5}$. Return the optimal value function and the optimal policy.
3. (written) Run the methods on the deterministic and stochastic version of the environment. How does stochasticity affect the number of iterations required, and the resulting policy?
   You can render the policy using `env.render_policy(policy)`
4. (written) Compare value iteration and policy iteration. Highlight pros and cons of each method.

## Answers

### 5.1

See the completed code in vipi.py.

### 5.2

See the completed code in vipi.py.

### 5.3

When using a sotchastic environment, the number of iterations does not really change. It's around 5 for the policy iteration. And around 1200 for the value iteration.
The optimal policy is different as it has to be more careful and try to avoid the bottom line (which is a penalty line). The more random it is the more it tries to go away from the bottom line. One can see that once we have reached less than 0.25 chances of success, it's worth predicting any other actions rather than the one we want to perform. (The unchosen actions are more likely to occur than the chosen one!)

### 5.4

Both succeed to find the optimal policy. But the fastest one is the policy iteration. It does a lot less iterations ( 5 vs 1200) than the value iteration. Even if each of its iteration is a lot more time consuming, this algorithm is still around 5 times faster on my laptop.
But note that at each iteration of the policy algorithm we solve a linear system. And that could lead to huge numerical errors if we are not careful. (For instance trying to inverse the matrix rather than solving the linear system.)