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

Dynamic Programming

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

Solution by Mahdi KALLEL

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.

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.

   ***a)* :** First of all, the reward r_s must be chosen to be negative, otherwise our model would wander over the non terminal states of the grid maximizing it's reward to infinity. $\implies$ $\mathrm{r}_s < 0$

   Now that the reward we get from a non terminal point is negative. Our policy becomes the shortest past to one of the terminal states.

   In order to guide the model towards state 14. It has to be more rewarding, from any point, to head to state 14 rather than state 0.

   If we take into account the walls in the grid. In order to get to 1, our model will have to pass by 2 or 5. We have to choose r_s such that in these two states it is more rewarding to head to 14 **(**)**. x
   If we enforce this property then we know that our model will be constantly looking to reach state 14.

   Both 2 and 5 lay at a distance $d_r = 1$ and $d_g = 3$ from 0 and 14 respectively.

   ** $\implies$ $3\ \mathrm{r}_s + r_g > r_s + r_r \implies 2\ \mathrm{r}_s > -r_g + r_r \implies \mathrm{r}_s > -10$.

   In conclusion we have to choose $-10 < r_s < 0$.For the optimal police π^* which follows the shortest path to 14. The optimal value function $v^*$is the sum of rewards on the shortest path from point P to 14. With the exception of point 0 which is a terminal state.

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

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 π in this MDP

induces a value function V^π. Suppose an affine transformation is applied to the reward, what is the new value function? Is the optimal policy preserved?

a)

$$V^\pi(S) = \mathbb{E}\,[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t)|s_0 = s; \pi)]$$

$$\implies V_2^\pi(S) = \mathbb{E}\,[\sum_{t=0}^{\infty} \gamma^t * \mathbf{a} * r(s_t, \pi(s_t)|s_0 = s; \pi) + \mathbf{b}]$$

$$\implies V_2^\pi(S) = a * \mathbb{E}\,[\sum_{t=0}^{\infty} \gamma^t * \mathbf{a} * r(s_t, \pi(s_t)|s_0 = s; \pi)] + \frac{\mathbf{b}}{\mathbf{1}-\gamma}$$

$$\implies V_2^\pi(S) = \mathbf{a} * V^\pi(S) + \frac{\mathbf{b}}{\mathbf{1}-\gamma}$$

b)

$$\pi^*(S) = \arg\max_a(Q^*(s,a))$$

$$Q_2(s,a) = r_2(s,a) + \gamma \sum_{s'} \pi(s'|a,s) * V_2(s')$$

$$\implies Q_2(s,a) = (a * r(s,a) + b) + \gamma \sum_{s'} \pi(s'|a,s) * (a * V(s') + \frac{b}{1-\gamma})$$

$$\implies Q_2(s,a) = (a * r(s,a) + b) + \gamma(\frac{b}{1-\gamma} + a * \sum_{s'} \pi(s'|a,s)V(s')))$$

$$\implies Q_2(s,a) = a * Q(s,a) + \frac{b}{1-\gamma}$$

The additive constant being the same for all actions, it does not affect the argmax.

If a $= 0$ then all actions are equivalent.

If $a \geq 0$ then the ordering of the actions remains the same and therefore the policy.

If $a \leq 0$ the ordering of the actions is changed and the policy changes.

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?

   Answer if c is such that $-10 < r_s + c < 0$ the optimal policy does not change.

   if $r_s + c > 0$ then : $V(r) = r_r, V(g) = r_g$ and $V(S) = +\infty$ for all non terminal states.

   if $-10 > r_s + c$ then the optimal policy will no longer be the shortest past to ***g*** but will be the shortest past to one of the terminal states that depends from the point.

2 Question

Consider infinite-horizon γ-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

$$\tau^{\pi_Q} V^\star = r(s,\pi_Q(s)) + \sum_{s'} P(s'|s,\pi_Q(s))V^\star S)$$

$$Q^\star(s,\pi_Q(s)) = r(s,\pi_Q(s) + \sum_{s'} P(s'|\pi_Q(s),s) * V^\star(s)$$

$$\implies Q^\star(s,\pi_Q(s)) = \tau^{\pi_Q} V^\star$$

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

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

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

$$\implies (1-\gamma)|V^\star(s) - V^{\pi_Q}(s)| \leq 2|V^\star(s) - Q(s,\pi_Q(s))|$$

$$\implies |V^\star(s) - V^{\pi_Q}(s)| \leq \frac{2|\max_a Q^\star(s,a) - \max_{a'} Q(s,a)|_\infty}{1-\gamma} (**)$$

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

$$(**) - Q(s,a') \geq -\max_{a'} Q(s,a)$$

$$(**) Q^\star(s,a) - Q(s,a') \geq Q^\star(s,a) - \max_{a'} Q(s,a)$$

$$(**) \max_a \{Q^\star(s,a) - Q(s,a')\} \geq \max_a \{Q^\star(s,a) - \max_{a'} Q(s,a)\} = \max_a Q^\star(s,a) - \max_{a'} Q(s,a)$$

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

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

$$\sum_s \mu^{\pi'}(s) \sum_a (\pi'(a|s) - \pi(a|s)) Q^\pi(s,a) + r(s,a)\pi'(a|s) - r(s,a)\pi'(a|s)$$

$$g^{\pi\prime} + \sum_s \mu^{\pi'}(s) \sum_a (\pi'(a|s) - \pi(a|s)) Q^\pi(s,a) - r(s,a)\pi'(a|s)$$

$$g^{\pi\prime} + \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s)(Q^\pi(s,a) - r(s,a)) - \pi(a|s)Q^\pi(s,a)$$

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

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

$$\textbf{Stationary distribution property} \implies \sum_{s,a} \pi'(a|s)p(s'|s,a)\mu^{\pi'}(s) = \sum_s \mu^{\pi'}(s) \sum_a \pi'(a|s)p(s'|s,a)$$

$$= \sum_s P^\pi(s',s) = \mu^{\pi'}(s')$$

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

$$g^{\pi\prime} - g^\pi + \sum_s \mu^{\pi\prime}(s) \sum_a \pi(a|s)Q^\pi(s,a) - \sum_s \mu^{\pi'}(s) \sum_a \pi(a|s)Q^\pi(s,a)$$

$$g^{\pi\prime} - g$$

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
The environment state is represented by the following variables :
c_t : takes 3 values, representing the direction of the call (Up,Down,None) on the t^{th} floor.
$o_{i,t}$ binary if floor "t" is requested on elevator "i".
p_i : stands for the current position of the elevator. Takes 6 values.
d_i : direction of the i^{th} elevator. Takes 3 values : (Up,Down,Stationary)

* The direction of the call c_i needs to be included as if you're in the 3rd floor wanting to go up, and the elevator is going to floor 0 it's better that it picks you up on on it's way up.
* $p_i + d_i$ Are necessary for the agent to access roughly the elevators path and thus co-ordinate between the two.
* $o_{i,t}$ Is necessary for the elevator to stop at the intended floors

Therefore our input vector consists of 22 values for a state space with a cardinality of $N = 3^6 * 2^{12} * 6^2 * 3^2$ it will be impossible dealing with such a large space in a tabular way. We will need to use value function approximations.

For each elevator the set of possible actions is Up / Down / Stay. Therefore we have 3x3 = 9 possible actions.
Our main objective is to minimize the mean transport time for each passenger, taking into account the waiting time for the elevator T_w and the trip time T_t let $T_p = T_t + T_w$

The training being done in simulation we can access quantities T_t and T_w for each passenger and thereby the average wait time for each passer $\bar{T_p}$

We can consider training in an episodic way, each episode ending when there are no more calls i.e $\forall_i c_i = 0$. Let **E** be the states verifying this property and M_1 and M_2 the set of actions moving one and two elevators respectively

$$R(s,a) = \begin{cases} 0 \text{ if s} \notin E \text{ and a} \notin M_1 \cup M_2 \\ -\epsilon \text{ if s} \notin E \text{ and a} \in M_1 \\ -2\epsilon \text{ if s} \notin E \text{ and a} \in M_2 \\ -\bar{T_p} \text{ if s} \in E \end{cases}$$

Where ϵ is a small number representing energy consumption. It's goal is to encourage sparse solutions in terms of movement.

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 γ as provided by `env.gamma` and the terminal condition seen in class. Return the optimal value function and the optimal policy.

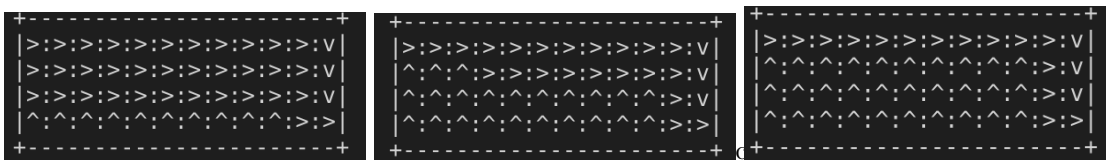
2. (coding) Implement `value_iteration` in `vipi.py`. The terminal condition is based on $\|V_{new} - V_{old}\|_\infty$ and the tollerance is 10^{-5}. Return the optimal value function and the optimal policy.

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)`

   Answer : We make our environment stochastic by changing the "prob succ" parameter from 1 to $P \leq 1$. By changing this parameter, the direction in which our agent moves is the chosen direction by the action with probability P_{succ} and a random direction with $P = 1 - P_{succ}$.

   When decreasing P_{succ} our agent becomes more cautious by getting further from the cliff. It gets closer to the cliff only when the exit is close.

   The number of iterations for Value Iteration and Policy iteration does not change very much with respect to P. We notice that Policy iteration takes considerably less steps.



(a) P=1 (b) P=0.8 (c) P=0.6

Figure 1: Evolution of model policy with P

P	Value Iteration	Policy Iteration
1	1376	6
0.8	1194	10
0.6	1218	5

Table 1: Number of steps per algorithm.

4. (written) Compare value iteration and policy iteration. Highlight pros and cons of each method. **Answer:** A Policy iteration iteration step consists a "policy evaluation" then "policy improvement". Whilst value iteration is finding the optimal value function then the greedy policy with respect to it.

   In terms of complexity one round of value iteration is $O(|S| * |A|)$ whilst policy iteration takes $O(|S|^2 * \frac{\log(1|\epsilon)}{\log(1|\gamma)})$ for policy evaluation and $O(|S||A|)$ for policy improvement.

   "Policy iteration" generates a better policy for each step. Whilst the value function generated at a "Value iteration" step does not correspond necessarily to any policy.

   In our experiment we confirm the empirical statement that "Policy iteration" converges faster than "Value iteration". In our case it is an order of magnitude faster.