Reinforcement Learning
Exercise 2

Mathias Niepert, Vinh Tong

April 21, 2023

1 Formulating Problems (2 Points)

The first step to solve a real world problem using Reinforcement Learning is often to formalize the problem as an MDP. Describe the set of states, the set of actions and the reward signal you would use for the problems a)-c). Are they discrete, continues, how many dimensions, etc? Come up with an own problem and describe it accordingly for d) (you can be creative/try to push the limits).

a) The game of chess

b) A pick and place robot

c) A drone that should stabilize in the air

d) Your own problem

2 Value Functions (3 Points)

a) For k-armed bandits, we defined the value as:

$$q(a) = \mathbb{E}[R_t \mid A_t = a]$$

For MDPs, the state-action value is defined as follows:

$$q(s, a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots \mid S_t = s, A_t = a]$$

Argue why we do not need to consider future rewards in the bandit setting.

b) Show that $v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$.

c) We introduced the *Bellman equation* for v_π in terms of the four-argument function p. Express the recursive relationship of v_π in terms of $p(s' \mid s, a)$ and $r(s, a, s')$.

3 Bruteforce the Policy Space (7 Points)

For this task we will use the FrozenLake environment from gym (https://www.gymlibrary.ml/environments/toy_text/frozen_lake/). The code template can be found on Ilias in *ex02-mdps/ex02-mdps.py*.

FrozenLake is a simple grid world with 4 actions (0-left 1-down 2-right 3-up). However, the ground is slippery (the agent is on a frozen lake), so that it ends up on the correct next field only with probability $\frac{1}{3}$ (e.g. instead of going down it could also end up left or right). When the action would bump the agent into a border it would stay in the same state. At the goal the agent will receive +1 reward, elsewhere it receives 0 reward. An episode terminates when the agent ends up at the goal or in a hole.

We will only consider *discrete* policies $a = \pi(s)$ in this exercise.

Attention: The latest OpenAi-gym may not work for the coming exercises. An old version is required. It has been tested with gym version 0.18.0 (but should also be stable with version 0.18.3) for windows and 0.15.3 for macOS. There might also be version conflict for numpy, wheel library, a virtual environment such as annaconda is recommended!

a) Given a discrete state space of size $|\mathcal{S}|$ and a discrete action space of size $|\mathcal{A}|$ (same size in all states). How many different policies exist? (1P)

b) The Bellman equation for the value function for a policy in the tabular case could be specified by

$$\boldsymbol{v}_\pi = \boldsymbol{r} + \gamma \boldsymbol{P}_\pi \boldsymbol{v}_\pi$$

with $\boldsymbol{v}_\pi = (v_\pi(s_0),\ v_\pi(s_1),\ \ldots)^\top$ being the vector of values at each state, $\boldsymbol{r} = (r(s_0),\ r(s_1),\ \ldots)^\top$ being the expected rewards at each state and $\boldsymbol{P}_\pi$ is the transition probability matrix of the policy π (the entry $P_{i,j}$ gives the probability of ending up in state s_i when being in state s_j before).

Solve the equation for $\boldsymbol{v}_\pi$ (write down intermediate steps). Implement the result in the code in function *def value_policy*. Use $\gamma = 0.8$ as discount factor. The code should now output the optimal values for the policy that always selects left as action and for the policy that always selects right as action. Show this output. Is this what you expect, why? (2P)

c) In function *def bruteforce_policies*: iterate over all possible policies and compute $\boldsymbol{v}_\pi$. What is the optimal value function v_*? How many optimal policies exist? What are the optimal actions for each state? (3P)

Hint: For terminal states it does not matter which action you choose, you can skip them in order to improve performance

d) Try to use larger map sizes (top of the file). Does the solution method still work well? What assumptions does this solution method make that are rarely true in practice? (1P)