Abu El Komboz, Tareq 3405686
Wurm, Marcel 3695946
Jain, Likhit 3678905

Reinforcement Learning

Assignment Solution 2

Date: April 29, 2023

2.1

a) Chess

- States: Every possible boardstate, once for each players' current turn
- Actions: Every currently possible move for the player whose turn it is
- Reward: High reward for winning / High negative reward for losing / (Small reward for taking a piece / Negative reward for losign a piece)
- Discrete
- 1-dimensional

b) Pick&Place robot

- States:
  - GoingToPickup/GoingToPlace (discrete)
  - Robot position / Object position / Target position / Current speed / wheel position (continuous)
- Actions: Accelerate/Decelerate / Steer left/right / Pickup / Place
- Reward: Reward for placing object on target (/ Small negative reward for doing nothing)
- 2-dimensional

c) Stabilizing drone

- States: Gyroscope info / Rotor speeds
- Actions: Accelerate/Decelerate Rotors
- Reward: Reward for staying in a stable position (according to gyroscope data) / Very negative reward for crashing
- Continuous
- 3-dimensional

d) Tetris

- States: Board state / Active piece / Next piece
- Actions: Move left/right / Rotate piece left/right / Do nothing
- Reward: Reward for gaining points / Very negative reward for losing
- Discrete
- 2-dimensional

2.2

a) In the bandit setting, we do not model states because choosing an action does not change the environment. We always stay in the same state. Therefore, the action-value does not depend on future rewards as these are independent of the current chosen action.

b)

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}\Big[G_t|S_t = s\Big] \\
&= \sum_{g_t} Pr\{G_t = g_t|s\} \cdot g_t \\
&= \sum_{g_t}\sum_{a} Pr\{g_t, a|s\} \cdot g_t \\
&= \sum_{g_t}\sum_{a} Pr\{a|s\} \cdot Pr\{g_t|a, s\} \cdot g_t \\
&= \sum_{a} Pr\{a|s\} \cdot \sum_{g_t} Pr\{g_t|a, s\} \cdot g_t \\
&= \sum_{a} \pi(a|s) \cdot \mathbb{E}\Big[G_t|S_t = s, A_t = a\Big] \\
&= \sum_{a} \pi(a|s) \cdot \mathbb{E}\Big[G_t|S_t = s, A_t = a\Big] \\
&= \sum_{a} \pi(a|s) q_\pi(s, a)
\end{aligned}$$

c)

$$\begin{aligned}
v_\pi(s) &= \sum_{a} \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big] \\
&= \sum_{a} \pi(a|s) \sum_{s'}\sum_{r} p(s', r|s, a)[r + \gamma v_\pi(s')] \\
&= \sum_{a} \pi(a|s) \sum_{s'}\sum_{r} \Big[p(s', r|s, a) \cdot r + p(s', r|s, a) \cdot \gamma v_\pi(s')\Big] \\
&= \sum_{a} \pi(a|s) \sum_{s'} \left[\sum_{r} p(s', r|s, a) \cdot r + \sum_{r} p(s', r|s, a) \cdot \gamma v_\pi(s')\right] \\
&= \sum_{a} \pi(a|s) \sum_{s'} \left[\sum_{r} p(s', r|s, a) \cdot r + \gamma v_\pi(s') \sum_{r} p(s', r|s, a)\right] \\
&= \sum_{a} \pi(a|s) \sum_{s'} \left[\sum_{r} p(s', r|s, a) \cdot r + \gamma v_\pi(s') p(s'|s, a)\right] \\
&= \sum_{a} \pi(a|s) \sum_{s'} \left[\sum_{r} p(s'|s, a) \cdot p(r|s, a, s') \cdot r + \gamma v_\pi(s') p(s'|s, a)\right] \\
&= \sum_{a} \pi(a|s) \sum_{s'} \left[p(s'|s, a) \sum_{r} p(r|s, a, s') \cdot r + \gamma v_\pi(s') p(s'|s, a)\right] \\
&= \sum_{a} \pi(a|s) \sum_{s'} \Big[p(s'|s, a) \cdot r(s, a, s') + \gamma v_\pi(s') p(s'|s, a)\Big] \\
&= \sum_{a} \pi(a|s) \sum_{s'} p(s'|s, a)\Big[r(s, a, s') + \gamma v_\pi(s')\Big]
\end{aligned}$$

2.3

a) Number of policies $= |\mathcal{A}|^{|\mathcal{S}|}$

b) Solving the Bellman equation for v_π:

$$\begin{aligned} v_\pi &= r + \gamma P_\pi v_\pi & &| - \gamma P_\pi v_\pi \\ v_\pi - \gamma P_\pi v_\pi &= r & & \\ (I - \gamma P_\pi) v_\pi &= r & &| \cdot (I - \gamma P_\pi)^{-1} \\ v_\pi &= (I - \gamma P_\pi)^{-1} r & & \end{aligned}$$

The ouput of our code:

$$v_{\pi_1} = \begin{pmatrix} v_{\pi_1}(s_1) \\ v_{\pi_1}(s_2) \\ v_{\pi_1}(s_3) \\ v_{\pi_1}(s_4) \\ v_{\pi_1}(s_5) \\ v_{\pi_1}(s_6) \\ v_{\pi_1}(s_7) \\ v_{\pi_1}(s_8) \\ v_{\pi_1}(s_9) \end{pmatrix} = \begin{pmatrix} 0.00 \\ 0.00 \\ 0.54 \\ 0.00 \\ 0.00 \\ 1.48 \\ 0.00 \\ 0.00 \\ 5.00 \end{pmatrix}, \text{ with } \pi_1 = \text{always left}$$

$$v_{\pi_2} = \begin{pmatrix} v_{\pi_2}(s_1) \\ v_{\pi_2}(s_2) \\ v_{\pi_2}(s_3) \\ v_{\pi_2}(s_4) \\ v_{\pi_2}(s_5) \\ v_{\pi_2}(s_6) \\ v_{\pi_2}(s_7) \\ v_{\pi_2}(s_8) \\ v_{\pi_2}(s_9) \end{pmatrix} = \begin{pmatrix} 0.41 \\ 0.77 \\ 1.31 \\ 0.36 \\ 0.82 \\ 2.30 \\ 0.13 \\ 0.00 \\ 5.00 \end{pmatrix}, \text{ with } \pi_2 = \text{always right}$$

The output matches our expectations. If the policy is always going left, then all states left from the goal have the value 0 because it is not possible to reach the goal from those states by only moving left. The closer the state is to the goal state, the higher its value(which makes sense too). Hence, the goal state has the highest value. The hole state has always the value 0 as it is not possible to reach the goal state from there. The always-right policy is better than the always-left policy.

c) The optimal value function is $v_* = \begin{pmatrix} v_*(s_1) \\ v_*(s_2) \\ v_*(s_3) \\ v_*(s_4) \\ v_*(s_5) \\ v_*(s_6) \\ v_*(s_7) \\ v_*(s_8) \\ v_*(s_9) \end{pmatrix} = \begin{pmatrix} 0.50 \\ 0.83 \\ 1.31 \\ 0.54 \\ 0.98 \\ 2.30 \\ 0.31 \\ 0.00 \\ 5.00 \end{pmatrix}$

There are 32 formally distinct optimal policies. It is reasonable to ignore the actions that are chosen when being in the hole- or the goal-state, as these have essentially no effects. Taking this into account, we identified two different strategies, which show which action to choose given a state:

$$\pi_1 = \begin{pmatrix} \pi_1(s_1) \\ \pi_1(s_2) \\ \pi_1(s_3) \\ \pi_1(s_4) \\ \pi_1(s_5) \\ \pi_1(s_6) \\ \pi_1(s_7) \\ \pi_1(s_8) \\ \pi_1(s_9) \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 2 \\ 3 \\ 3 \\ 2 \\ 0 \\ x \\ y \end{pmatrix} \text{ and } \pi_2 = \begin{pmatrix} \pi_2(s_1) \\ \pi_2(s_2) \\ \pi_2(s_3) \\ \pi_2(s_4) \\ \pi_2(s_5) \\ \pi_2(s_6) \\ \pi_2(s_7) \\ \pi_2(s_8) \\ \pi_2(s_9) \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 2 \\ 3 \\ 3 \\ 2 \\ 0 \\ x \\ y \end{pmatrix}$$

That means that in the first state, it is equally good to take action 1 or to take action 2.

d) The brute force solution method does not work anymore due to a memory exhaustion exception. This method is only useful, iff the space of all policys is relatively small. In practice, this assumption is rarely the case.