Exercise 2

Kai Schneider

May 2, 2021

Task 1 Formulating Problems

as a Markov Decision Process $MDP = \{S, A, T, R, \gamma\}$

a.) chess

Discrete problem with a deterministic trasition function.

- **states:** all allowed configurations of the playing pieces (32, 16 for each player) on the field ($8x8 = 64$ segments). The number and type of the remaining pieces influences the possible configurations.
- **actions:** movements of the playing pieces. The number of available actions is determined by the current state (remaining pieces and reachable fields).
- **reward:** the only meaningful reward is determined by winning or loosing the game, since loosing playing pieces might be negative in the short run but benefit the long term strategy.

b.) pick & place robot

Continous problem (at least states and actions)

- **states:** all possible configurations of the joint angles the endeffector/toolhead ($s \in \mathbb{R}^n$)
- **actions:** all possible changes in joint angles and endeffector states ($a \in \mathbb{R}^n$)
- **reward:** multiple possibilities for a reward signal, e.g.:
  - positive reward for (succesfully) delivering a part
  - positive reward for moving with a part to the place location
    (and vice versa a neg. reward for moving without one)

c.) drone

State- and action-space are also continuous, although the drone itself surely operates discrete.

- **states:** 3D position and orientation (6 DOF) of the drone
  (for the controller also speed and acceleration)
- **actions:** changes/corrections in the rpm of the motors (assuming a multicopter-like drone)
- **reward:** reward could be a value relative to the deviation to the target value/position/orientation.

d.) own problem - commissioning

Picking and packing a predefined list of articles/objects from a larger range of things.

- **states:** the current state is always described by the already collected items
  (or in contrast all articles which still have to be collected).
- **actions:** Each a describes moving to another article and collecting it. A is the always the set of all a's for the remaining objects.
- **reward:** Like in the chess example, only the result after reaching the terminal state (collected all articles) is meaningful. Depending on the optimization goal a reward relative to the time (picking rate) or the covered distance (energie) might be a good choice.

Task 2 Value Functions

$$
\begin{aligned}
&\text{k-bandit:} && q(a) = \mathbb{E}[R_t | A_t = a] \\
&\text{MDP:} && q(s,a) = \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots | S_t = s, A_t = a]
\end{aligned}
$$

a.)

In contrast to the MDP, the bandits don't have multiple possible states. So each action A_t immediately returns a reward R_t. This reward doesn't depend on previous states/actions, so the potential future rewards aren't relevant here.

b.)

$$
\begin{aligned}
\upsilon_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] &= \sum_a \Pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma \mathbb{E}[G_{t+1} | S_{t+1} = s']] \\
&\Leftrightarrow \sum_a \Pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma \sum_{a'} \Pi(a'|s') \mathbb{E}[G_{t+1} | S_{t+1} = s', A_{t+1} = a']] \\
&= \sum_a \Pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma \sum_{a'} \Pi(a'|s') q_\pi(s', a')] \\
&= \sum_a \Pi(a|s) q_\pi(s, a)
\end{aligned}
$$

with slides 2.31 & 2.32 □

c.)

$$
\begin{aligned}
\upsilon_\pi(s) &= \sum_a \Pi(a|s) \sum_{s',r} p(s', r | s, a)[r + \gamma \upsilon_\pi(s')] \\
&= \sum_a \Pi(a|s) \sum_{s'} \sum_r p(s', r | s, a)[r + \gamma \upsilon_\pi(s')] \\
&= \sum_a \Pi(a|s) \left[\sum_{s'} \sum_r p(s', r | s, a) r + \gamma \sum_{s'} \sum_r p(s', r | s, a) \upsilon_\pi(s') \right] \\
&= \sum_a \Pi(a|s) \left[\sum_{s'} p(s' | s, a) r(s, a, s') + \gamma \sum_{s'} p(s' | s, a) \upsilon_\pi(s') \right]
\end{aligned}
$$

with slides 2.31 & 2.21 □

Task 3

a.)

With state space size of $|S|$ and an action space size of $|A|$ there exist $|\Pi| = |A|^{|S|}$ different policies.

b.)

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

The return for the $3x3$ grid is the following:

policy left: $[0, 0, 0.537, 0, 0, 1.477, 0, 0, 5]$
policy right: $[0.414, 0.775, 1.311, 0.364, 0.819, 2.295, 0.132, 0, 5]$

As we can see the values for the *policy right* are on average much higher. This makes intuitivly sense because we start in the upper left corner of the grid and want to reach the bottom right one.

c.)

The optimal value function v_π is:
$[0.49756712, 0.83213812, 1.31147541, 0.53617147, 0.97690441, 2.295081970.3063837, 0, 5]$

There exist 8 optimal policies which result in this value function:

$$\begin{aligned} &[2, 2, 2, 3, 3, 2, 0, 0, 1] \\ &[1, 2, 2, 3, 3, 2, 0, 0, 2] \\ &[2, 2, 2, 3, 3, 2, 0, 0, 2] \\ &[2, 2, 2, 3, 3, 2, 0, 0, 3] \\ &[2, 2, 2, 3, 3, 2, 0, 0, 0] \\ &[1, 2, 2, 3, 3, 2, 0, 0, 1] \\ &[1, 2, 2, 3, 3, 2, 0, 0, 3] \\ &[1, 2, 2, 3, 3, 2, 0, 0, 0] \end{aligned}$$

d.)

Because of the complexity, even for a "slightly" larger state space of $4x4$ the computation time grows exponentially.
While we had $|A|^{|S|} = 4^9 = 262144$ different policies with the $3x3$ grid, we already have $4^{16} = 4294967296$ for the 4x4.

In real world problems the state & action spaces are typically much larger, so this kind of method to solve the problem is extremly inpractical.