

# Reinforcement Learning

## Exercise 2 - Solution

Jonathan Schnitzler - st166934  
ErickVillanuevaVillasenor - st190300  
Eric Choquet - st160996

April 29, 2024

## 1 Formulating Problems

### a) The game of chess

**States** The position of all pieces on the board. A chess board is a 8x8 grid, which starts with 16 white and 16 black pieces on opposing sites. The state space is large (an upper bound from around  $\approx 10^{45}$ <sup>1</sup>).

**Actions** The possible moves of the current player. The number of possible moves is limited by the number of pieces on the board and the rules of chess.

#### Reward Signal

- win, lose or draw the game (by checkmate)
- evaluate the current position of the board (e.g. material advantage, positional advantage)

### b) A pick and place robot

#### States

- position and orientation of the axes
- is holding something
- source of objects and destination

---

<sup>1</sup>see <https://tromp.github.io/chess/chess.html>

### **Actions**

- pick an object
- place an object
- move

### **Reward Signal**

- successfully pick and place an object
- time to pick and place an object
- lost an object

## **c) A drone which should stabilize in air**

### **States**

- tilt angle
- velocity in the three axes
- angular velocity
- crashed

### **Actions**

- adapt speed of individual rotors

### **Reward Signal**

- time in air
- minimize the tilt angle
- minimize steering (and energy consumption)

## **d) Playing tetris**

### **States**

- position of the falling block
- position of the other blocks
- preview of next block

### Actions

- move block left/right
- rotate block

### Reward Signal

- clear a row
- lose the game

## 2 Value Functions

**a) k-armed Bandits as MDP** The Future rewards are independent on the current state. Since there is only one state, i.e. you are about to roll a bandit, the only thing which could change is our policy based on new estimates of the true distribution - exploration.

This is not considered in the expected reward and therefore its basically the same  $R_t = R$  for all rolls. Then it holds that the sum is just an additional factor, which is not relevant for the value function

$$\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} = R \sum_{k=0}^{\infty} \gamma^k = R \frac{1}{1-\gamma} \quad (1)$$

since  $\gamma < 1$ .

**b) Proof Relation of value and action-value function** We shall show that

$$v_{\pi}(s) = \sum_a \pi(a|s) q_{\pi}(s, a) \quad (2)$$

holds. We can use the definition of the value function

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] \quad (3)$$

and the definition of the action-value function

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a] \quad (4)$$

to show the relation. We use the law of total expectation to condition on the action  $A_t = a$ :

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s] \quad (5)$$

$$= \sum_a \pi(a|s) \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]. \quad (6)$$

$$= \sum_a \pi(a|s) q_{\pi}(s, a) \quad (7)$$

c) **Rephrase value function** The bellman equation is given by

$$v_{\pi}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_{\pi}(s')] \quad (8)$$

We can execute the sum over  $r$  and note that the reward is dependent on  $r(s, a, s')$  and the transition probability is dependent on  $p(s'|s, a)$ , i.e.

$$\sum_{s',r} p(s',r|s,a)r = \sum_{s'} p(s'|s,a)r(s, a, s') \quad (9)$$

Which yields the rephrased value function

$$v_{\pi}(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)[r(s, a, s') + \gamma v_{\pi}(s')] \quad (10)$$

### 3 Bruteforce the Policy Space

a) **Number of policies** There are  $3 \cdot 3 = 9$  tiles, which are the states. There are four actions, namely up, down, left and right. If the policy is deterministic, then for each tile the decision tree branches into four additional options, i.e.

$$n_{\pi} = a^s = 4^9 = 2^{18} = 262144 \quad (11)$$

which is do-able for a computer but still kind of ridiculous.



Figure 1: 3x3 gridworld of icy lake

b) **Do you expect the behaviour?** That the left policy can only fail by starting on the left of the object is reasonable. I would have expected that tile  $(x, y) = (1, 1)$  would have a higher value

0	0	0.537
0	0	1.477
0	0	5

Figure 2:  $v_{\pi}$  for  $\pi = \text{Go left policy}$

0.414	0.775	1.311
0.364	0.819	2.295
0.132	0	5

Figure 3:  $v_{\pi}$  for  $\pi = \text{Go right policy}$

**c) Bruteforce** There are two optimal policies for the given optimal value function  $v_*$ , which are shown in Table 4. Namely,

0.498	0.832	1.311
0.536	0.977	2.295
0.306	0	5

Figure 4: Optimal value  $v_*$

1/2	2	2
3	3	2
0	-	-

Figure 5: Best policies  $\pi_*$

**d) Is Bruteforce a good idea?** No. In general it is not feasible to Bruteforce any algorithm, which scales exponentially  $O(e^n)$ . For larger maps this is the case. The assumption for our solution is to calculate every policy and find the optimal one. Other algorithms should be used, which dont calculate the whole space.