Assignment 2 - Markov Decision Processes

Jonathan Stumber
Christian Reiser
Magnus Ostertag

1. Formulating Problems

1.1 a) The game of chess

1.1.1 States

On a board of size 8x8=64 are up to 16 units for both colours (white and black), which can be rook, knight, bishop, queen, king, or pawns. We can estimate an upper bound for the states by the number of ways to have 32 fields possessed $\binom{64}{32}$ times the number of ways the possessed filed can belong to black or white $\binom{32}{16}$ times the number of choices for each possessed field (6 units + possibility of being empty) 7^{32} times the choices for the active player 2.

$$\binom{64}{32} * \binom{32}{16} * 7^{32} * 2 \approx 2.43 * 10^{54}$$

This includes some repetition of states (if a filed is empty), and some impossible states (i.e. pawns are in the last row, or multiple kings). You can also add an end state, where no player is active. And for some rules you should track if the king has moved, the last time a pawn moved, or a figure was captured, if the situation has occurred before.

Additionally, there can be a clock tracking the time for each player.

1.1.2 Set of actions

Do a legal move with one unit of a maximum of 16 units. Additionally, there are special moves like Castling. The number of legal moves can be much lower, especially towards the end of a game. Also, you can offer a draw to your opponent.

1.1.3 Reward signal

Terminal reward signals: Winning is positive, draw is neutral, losing is negative. There can also be a intermediate goal of maximizing the difference between your own and the opponents units: $max(ownWorth - opponentsWorth)$.

If tracking time, there should be a positive reward for your opponent needing more time and a negative reward of needing time to think.

1.1.4 Dimensions

2D board

and optional time.

1.1.5 Discrete

If no time is tracked, everything is discrete: The board, the time-steps and the units.

I time is tracked, the time is continuous.

1.2 b) A pick and place robot

We consider a simple endeffector of a robotic arm:

1.2.1 States

3D position state, 3D velocity state, binary open or closed state. Additionally there can be angular velocities/positions and the consideration other parts of the robotic arm.

1.2.2 Set of actions

Actions are to control the gains of each motor to set position, velocity, and open/closed state.

1.2.3 Reward signal

Positive reward for successful pick and place. Negative reward for passed time and drops.

1.2.4 dimensions

There are three spacial dimensions.

1.2.5 discrete/continuous

The open or closed endeffector can be discrete state, whereas velocity and position are continuous.

1.3 c) A drone that should stabilize in the air

We consider a quadrocopter, which should hover at a given position.

1.3.1 States

Angular (pitch, yaw, roll), Position (x,y,z) and velocity (u,v,w) and angular velocity.

1.3.2 Set of actions

Set voltage of each propeller-motor.

1.3.3 Reward signal

A reward signal could be the negative cost of the difference between target state and actual state.

1.3.4 Dimensions

There are three spacial dimensions.

1.3.5 Discrete/continous

The position, angle, velocities and voltage can be continuous.

1.4 d) Our own problem: Inverted pendulum

1.4.1 States

There are only two states, the position in the rail and angle of pendulum.

1.4.2 Set of actions

Acceleration is the only action.

1.4.3 Reward signal

A reward signal could be the negative cost of the difference between the target and actual position and angle.

1.4.4 Dimensions

There is one spacial dimension.

1.4.5 Discrete/continuous

Acceleration, position, and angle can be continuous.

2. Action Selection Strategies

2.1 Considering future rewards in the bandit setting

There is no need to consider future rewards in the bandit setting, because actions have no effect on the environment and thus have no influence on future rewards.

2.2 Show

The value $v(s)$ of state s under a policy π can be obtained with marginalizing over the possible states a:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[G_t | S_t = s] \\ &= \sum_a \mathbb{E}_\pi[G_t | S_t = s, A_t = a] * P(A_t = a | S_t = s) \\ &= \sum_a q_\pi(s,a)\pi(a|s)\,. \end{aligned}$$

2.3 Recursive value relationship

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

3. Bruteforce the Policy Space

3.1 a) number of policies

In each state the agent can take all actions. The number of different policies are

$$|S|^{|A|} = 4^9 = 262144 \tag{1}$$

where $|S| = 3 * 3 = 9$ is the size of the state space and $|A| = 4$ is the size of the action space. In this case even the senseless actions in the terminal space are included. Exluding these two leads to 4^7.

3.2 b)

Bellman Equation:

$$\mathbf{v}_\pi = \mathbf{r} + \gamma \mathbf{P}_\pi \mathbf{v}_\pi. \tag{2}$$

Subtract $\gamma \mathbf{P}_\pi \mathbf{v}_\pi$:

$$-\gamma \mathbf{P}_\pi \mathbf{v}_\pi + \mathbf{v}_\pi = \mathbf{r}. \tag{3}$$

Factorize:

$$(-\gamma \mathbf{P}_\pi + I)\mathbf{v}_\pi = \mathbf{r}. \tag{4}$$

Multiply with $(-\gamma \mathbf{P}_\pi + I)^{-1}$:

$$(-\gamma \mathbf{P}_\pi + I)^{-1}(-\gamma \mathbf{P}_\pi + I)\mathbf{v}_\pi = (-\gamma \mathbf{P}_\pi + I)^{-1}\mathbf{r}. \tag{5}$$

simplify:

$$\mathbf{v}_\pi = (-\gamma \mathbf{P}_\pi + I)^{-1}\mathbf{r}. \tag{6}$$

Value function for policy left (always going left):

```
0      0.     0.53691275
0.     0.     1.47651007
0.     0.     5.
```

Value function for policy right (always going right):

```
0.41401777    0.77456266    1.31147541
0.36398621    0.8185719     2.29508197
0.13235862    0.            5.
```

This is similar to what we expected. In the always-left policy you can only get to the goal by chance if you start at the right wall. If you start in the middle or the left, you will never get to the goal.

In the always-right policy you can always get to the goal, unless you start in a hole. If you are on the fare right, you are guarantied to end up in the goal eventually, because you always move up, down or stand still. So, the only reachable termination state is the goal. With increasing distance the values gets smaller, because for each move the reward is discounted stronger.

3.3 c)

v$*$ =

0.49756712	0.83213812	1.31147541
0.53617147	0.97690441	2.29508197
0.3063837	0.	5.

We found two optimal policies:

```
1 2 2
3 3 2
0 h g

2 2 2
3 3 2
0 h g
```

3.4 d)

The assumption is that the number of possible policies to evaluate is too large to work in practice.

Increasing the size to 4x4 should theoretically work, however it takes indefeasibly long to evaluate all possible policies. With more computing power, time, or move efficient code somewhat larger sizes are possible, but the computational complexity increases very fast with the size and possible actions.