Reinforcement Learning, Tutorial 02

Philipp Kratzer

Machine Learning and Robotics Lab

May 6th, 2020

Announcements

- Gradings now available in ilias
- Third exercise sheet is available

1

Task: Formalize problems for Reinforcement Learning. Describe the set of states, the set of actions and the reward signal you would use for the problems.

- No single correct solution
- Choice of states/actions/rewards strongly affects performance of algorithms
- At present finding good representations is often more art than science

1a - The game of chess

- states: all possible board configurations
- actions: possible moves in the state
- reward: +1 win, -1 loose, 0 else

1b - Pick and place robot

- states: joint space of the robot and location of object
- actions: motor controls
- reward: distance of object to goal position

- Can the state also be the camera image?

1c - A drone you want to stabilize in the air

- states: position, rotation and velocities of the drone
- actions: control signal for propellers
- reward: negative reward for velocities, negative reward for crash

1d - Your own problems

- Throwing a Basketball
  set of states: Distance and angle to the basket.
  set of actions: The three angles and three velocities with which the ball is thrown
  reward signal: Hit or miss, i.e. $R = 1, 0$.
- A student prepares for exams
  states: awake, asleep
  actions: study, sleep, play computer
  reward: exam results

1d - Your own problems

- Telescope searching extraterrestrial lifeforms
  States: searching, found alien waving.
  Actions: move telescope (continuous angles, two axis), zoom in/out.
  Rewards: positive (promising section (high reward), alien found (notify press, Nobel prize), no life possible in section (medium reward) all info is useful info, just question of how efficient is the search).
- Agent that picks music in a club. Good music and a good mix animates people to dance, bad music and randomly mixed music does the opposite.
  States: Number of people on the dance floor
  Actions: possible songs
- And a lot more: Toaster, Social media bot, Protein folding, games, cooking, . . .

2a

Task: 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 follow:

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

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

- In the bandit setting, the action has no influence on the future (no state change)

2b

Task: Show that $v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$

$$
\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \\
&= \sum_a \pi(a \mid s) \mathbb{E}_\pi [R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \\
&= \sum_a \pi(a \mid s) q(a \mid s)
\end{aligned}
$$

2c

Task: Express the *Bellman equation* in terms of $p(s' \mid s, a)$ and $r(s, a, s')$.

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

3

Task: How many different *discrete* policies exist?

- $|\mathcal{A}|^{|\mathcal{S}|}$

Task: Solve $\boldsymbol{v}_\pi = \boldsymbol{r} + \gamma \boldsymbol{P}_\pi \boldsymbol{v}_\pi$ for $\boldsymbol{v}_\pi$

$$\begin{aligned} \boldsymbol{v}_\pi &= \boldsymbol{r} + \gamma \boldsymbol{P}_\pi \boldsymbol{v}_\pi \\ \boldsymbol{v}_\pi - \gamma \boldsymbol{P}_\pi \boldsymbol{v}_\pi &= \boldsymbol{r} \\ (I - \gamma \boldsymbol{P}_\pi)\boldsymbol{v}_\pi &= \boldsymbol{r} \\ \boldsymbol{v}_\pi &= (I - \gamma \boldsymbol{P}_\pi)^{-1} \boldsymbol{r} \end{aligned}$$

3

Task: What is the value function for the policy that always selects left:

0.	0.	0.54
0.	0.	1.48
0.	0.	5.

Task: What is the value function for the policy that always selects right:

0.41	0.77	1.31
0.36	0.82	2.3
0.13	0.	5.

Task: Optimal Value function:

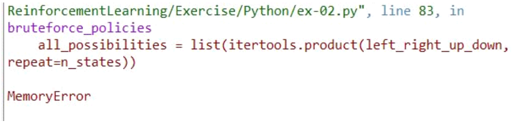
0.50	0.83	1.31
0.54	0.98	2.30
0.31	0.	5.

Task: Optimal Policy:

Down/Right	Right	Right
Up	Up	Right
Left	-	-

3d

Task: Assumptions this solution method makes that are rarely true in practice:

- Enough computational resources (takes forever even for small problems!)
  - "After half an hour on the 4x4 map I interrupted the program"
  - 

```
ReinforcementLearning/Exercise/Python/ex-02.py", line 83, in
bruteforce_policies
    all_possibilities = list(itertools.product(left_right_up_down,
repeat=n_states))

MemoryError
```

- Accurate knowledge of dynamics
- finite, tabular state/action space

Next exercise sheet

- Next exercise sheet available
- It will be about dynamic programming
- Programming part is again on the FrozenLake environment
- Sourcecode on github
  `https://github.com/humans-to-robots-motion/rl-course`