DRL Homework 01

Leon Schmid

April 2023

How-To homework

Please read carefully:

- Please use the Homework submission form to hand in your assignment. (Submissions via Email are also accepted, but generally cause much higher workload for the teaching staff)
- Deadline: Submission by Sat. 29.4. 11.59pm
- Before you submit the homework, make sure to have formed a group of three students and have signed into a respective group on StudIP
- No homework review is necessary as this is the first week, no earlier homework submissions would be available from last week.
- Every single member of the group has to submit the homework via this form (i.e. three group members create 3 sepearte submissions via the form!)
- You are allowed and even encouraged to collaborate with fellow students in this class on this and any subsequent homework assignment.
- It's recommended to use either IPython Notebooks, or plain python files for code, with a MarkDown (.md) file for Task 1

Contents

1 Understanding MDPs

- Answer each question in a few sentences.

1.1 Chess

You are tasked with creating an AI for the game of chess. To solve the problem using Reinforcement Learning, you have to frame the game of chess as a Markov Decision Process (MDP). Describe both the game of chess formally as a MDP, also formalize the respective policy.

1.2 LunarLander

Check out the LunarLander environment on OpenAI Gym: Check out this Link!. Describe the environment as a MDP, include a description how the policy is formalized.

1.3 Model Based RL: Accessing Environment Dynamics

Discuss the Policy Evaluation and Policy Iteration algorithms from the lecture. They explicitly make use of the environment dynamics ($p(s', r|s, a)$).

- Explain what the environment dynamics (i.e. reward function and state transition function) are and give at least two examples.
- Discuss: Are the environment dynamics generally known and can practically be used to solve a problem with RL?

2 Implementing a GridWorld

GridWorlds are some of the most common tools for visualizing RL by creating an easily understandable and visually interpretable MDP (and especially agents and their policy within them). Fundamentally a Gridworld consists of an mxn sized world and the agent-position within this grid. The agent can typically choose between four actions: [LEFT, TOP, RIGHT, DOWN]. Additionally, there is typically a goal - a specific tile, which the agent is supposed to reach. When the agent reaches this tile, a reward is given and the environment is reset. You are tasked with implementing such a GridWorld.

2.1 Look up some examples

Look up some examples of GridWorld! List at least three links or references to different GridWorlds you could find online.

2.2 Implementing the MDP

First, make sure you understand the underlying MDP of a Gridworld.

- Implement a GridWorld Class
- Implement all key features defining an MDP in respectively named methods or attributes of this object
- Specify whether your MDP is deterministic or stochastic

As you have seen in previously researched examples, typically GridWorlds add some additional complexities for the agent. Some typical examples include Walls (non-passable tiles), traps (tiles that penalize passing them with some negative reward), or difficult terrain (typically icy or windy tiles, where agents are passed onto certain neighboring tiles instead of arriving where they planned to). For this task you may use the examples above or come up with your own ideas.

- Implement at least two complexities for your agent to overcome. You may use the examples above or come up with your own ideas (or take inspiration from GridWorlds you found online)

3 Implementing a policy

3.1 Implement the basic agent

- Implement a custom agent (i.e. a policy) to interact with your GridWorld
- The policy should be stochastic with non-zero probability for every action at every state

You are not supposed to implement a Reinforcement Learning algorithm here! Implement some simple heuristic, e.g. walk in the direction of the goal-state with a probability of 0.8, otherwise act randomly, or something similarily simple.

3.2 Evaluate the policy

- Sample at least 1000 episodes of your agent interacting with your self-built GridWorld
- For all states s, which have been reached at least once in these episodes, calculate a MC-estimation of $V_\pi(s)$ of this state.

4 Visualization (Optional)

This task is optional, and only required for outstanding homework submissions.

- Implement a simple visualization (you could go for ASCII art or some visualization tool of your choice) for your GridWorld

- Implement a method to include the MC-estimates of $V_\pi(s)$ onto your visualization.
- Include visualizations of the MC-estimates of $V_\pi(s)$ with MC-estimates from 50, 200, 500, 1000 and 10000 episodes