DRL Homework 02

Leon Schmid

Deadline: May 1st 23.59

Abstract

Homework 2 is your first Coding Exercise. Please make sure to get help and work efficiently on this project wherever possible: Work together with your group and spread the workload. Explicitly feel invited to work together with other groups. Make use of the Coding Support Sessions and Element if any questions come up.

This homework asks you to implement a gridworld and an n-step SARSA algorithm to solve it. All implementation details not specified below are left for you to decide.

Contents

1 Task 01

You are tasked with creating your own gridworld. You can find an example of a gridworld here: https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld$_{t}d.html$ - this is also discussed in Lecture 04. You generally have a lot of freedom (and have to make respective choices!) with this implementation, but make sure to implement at least the following

- Your gridworld should be at least 5x5 in size.
- At least a few tiles should be blocked (i.e. the agent can not access them)
- Exactly one tile should yield a positive reward when accessing it. This tile should correspond to a terminal state!
- Multiple tiles should yield a negative reward when stepping on them
- There should be a fixed starting point for your agent

- For a few tiles, there should be a non-deterministic state transition function. Come up with your own ideas, for inspiration feel invited to google windy gridworld
- Make your sure implementation creates an object representing this gridworld
- Your gridworld should either be created randomly (make sure paths to the terminal state are not completely blocked!) or be created in a fixed way (i.e. with you specifying either on initialization or hardcoding where positive/negative rewards are and where blocked tiles are, etc.)
- Your gridworld implementation should implement at least the followng functions, similar to an OpenAI gym (checking this out is not necessary, but might help: https://gym.openai.com/docs/environments)
  - reset() — Resets the gridworld to its initial state. Returns the initial state.
  - step(action) — Applies the state transition dynamics and reward dynamics based on the state of the environment and the action argument. Returns (1) The new state, (2) the reward of this step, (3) a boolean indicating whether this state is terminal.
  - visualize() — visualizes the current state (e.g. printing characters representing the grid on the command line, don't create too much effort with this if you are not familiar with tools making this easy)

2 Task 02

Implement Tabular n-step SARSA to solve the gridworld you have created.

- Your implementation should work for any n and be parameterized in that regard. Specifically make sure you can recover both a 1-Step SARSA and Monte-Carlo Control algorithm from your implementation
- **Optionally:** Create a visualization of the Q-values controlling your policy. This takes extra effort, but can help you understand what is happening if there is a bug in your implementation.
- Basic SARSA can be rather inefficient - under many configurations even a gridworld might take a few minutes to be solved. Have patience.

3 Task 03

Get in contact with 3 other groups and review their submission for homework 01. The goal of this task is to help you review your submission to last weeks homework by (1) actively engaging with the submission from different groups and (2) by receiving feedback to your submission.

- Use any channel available to you to get in contact with other groups - we recommend element though!
- Give feedback to their submission in 5-10 lines
- Forward the feedback via email to them
- Please make sure your feedback is critical but constructive