Reinforcement Learning WS 2024
TAs: René Geist and Pavel Kolev
Due-date: 22.10.2024, 14:00 (upload to ILIAS as one file)
Filename: homework1-NAME1-NAME2-...zip or .pdf
Homework Sheet 1
October 15, 2024

1 Optimal Policy – small Example

Consider the system shown in Figure 1. The only decision to be made is that in the top state, where two actions are available, *left* and *right*. The numbers show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right}. What policy is optimal if $\gamma = 0$? If $\gamma = 0.9$? If $\gamma = 0.5$?. *Hint:*[1]

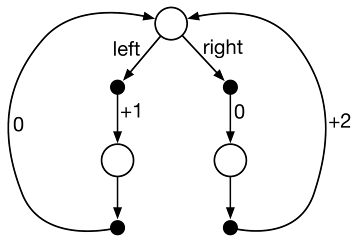

Figure 1: Simple system with one decision point.

2 Value Estimation in Grid Worlds

We will get our hands on a simple Gridworld domain, which is also used in some of the following exercises. For a simple random policy we will estimate the values of states.
We will use Python 3 for the coding exercises. The only package you need to have installed is `tkinter`. The code for this exercise contains the following files, available as zip archive `1_gridworld_explore.zip`:

Files:

agent.py The file in which we will later write our agents, currently just the random agent is implemented.

[1]Compute the value of the top state for both policies. Geometric series might be useful here.

mdp.py Abstract class for general MDPs.

environment.py Abstract class for general reinforcement learning environments (compare to mdp.py)

gridworld.py The Gridworld main code and test harness (You will need to write a bit code here, this time)

gridworldclass.py Implementation of the Gridworld internals.

utils.py some utility code, see below.

The remaining files graphicsGridworldDisplay.py, graphicsUtils.py, and textGridworldDisplay.py can be ignored entirely.
You will need to fill in portions in `gridworld.py`.

2.1 Gridworld: Getting started

To get started, run the Gridworld harness in interactive mode:

```
python3 gridworld.py -m
```

You will see the an two-exit Gridworld. Your agent's position is given by the blue dot, and you can move with the arrow keys. Notice that the agent's value estimates are shown, and are all zero. Manual control may be a little frustrating if the noise level is not turned down (-n), since you will sometimes move in an unexpected direction. Such is the life of a Gridworld agent! You can control many aspects of the simulation. A full list is available by running:

```
python3 gridworld.py -h
```

You check out the other grids, change the noise or discount, change the number of episodes to run and so on. If you drop the manual flag (-m) you will get the RandomAgent by default. Try:

```
python3 gridworld.py -g MazeGrid
```

You should see the random agent bounce around the grid until it happens upon the exit. Not the finest hour for an AI agent; we will build better ones soon.
Next, either use the text interface (-t) or look at the console output that accompanies the graphical output. Do a manual run through any grid you like, and notice that unless you specify quiet (-q) output, you will be told about each transition the agent experiences. Coordinates are in (row, col) format and any arrays are indexed by [row][col], with 'north' being the direction of decreasing row, etc. By default, most transitions will receive a reward of zero, though you can change this with the living reward option (-r). Note particularly, that the MDP is such that you first must enter a pre-terminal state and then take the special 'exit' action before the episode actually ends (in the true terminal state (-1, -1)).
You should definitely look at agent.py, mdp.py, and environment.py closely, and investigate parts of gridworld.py as needed. The support code should be ignored.

2.2 Implement return computation and value estimation

Implement the computation of the return of an episode in `gridworld.py`, see `runEpisode` function, and the computation of the estimated value of the start state (see main function) and `# Exercise 1` comments. Consult the slide on the *Value Function* (number 44).

(a) What is the value of the start state in the `MazeGrid` environment for the random policy (default discount factor $\gamma = 0.9$). Try different number of episodes (use -k xxxx -q) and provide mean value and the standard deviation over the episodes. Try also $k = 10000$.

(b) How many episodes do you need to be sure (confidence 95%) that your mean estimate is within ± 0.0004 of the true mean?
Hint:[2]

(c) Do the same with the `DiscountGrid` for a discount $\gamma = 0.95$. Now the confidence interval should be 0.1, i.e. with 95% confidence the mean estimate should be within ± 0.05 of the true mean.

(d) Does the value estimate predict the long-term average discounted rewards within confidence interval 0.1 for 500 episodes?

[2]Central limit theorem