CSCE 775: Deep Reinforcement Learning
Homework 1

Your code must run in order to receive credit.

For grading purposes, while you can add helper functions to your file, do not change the signature of the functions you should implement. These functions must accept and return exactly what is specified in the documentation.

Your code should complete in less than two minutes for all problems.

Installation

We will be using the same `conda` environment as in Homework 0.

The entire GitHub repository should be downloaded again as changes were made to other files. You can download it with the green "Code" button and click "Download ZIP" or pull from the repository.

1 Markov Reward Processes (20 pts)

Implement the `mrp` function. Given the definition of a Markov reward process (MRP) compute the state-values (expected future reward) for all states. You can test your code for the particular MRP in Figure 1 using the given homework script. The code will check the mean absolute error for gamma values of 0, 0.1, 0.9, and 0.999. You should have a mean absolute error close to zero. **However, your code should be able to compute the state-value for *any* MRP with a finite number of states, where all states have a finite state-value.**

Running the code:
`python run_hw1.py --task mrp --gamma <discount_factor>`

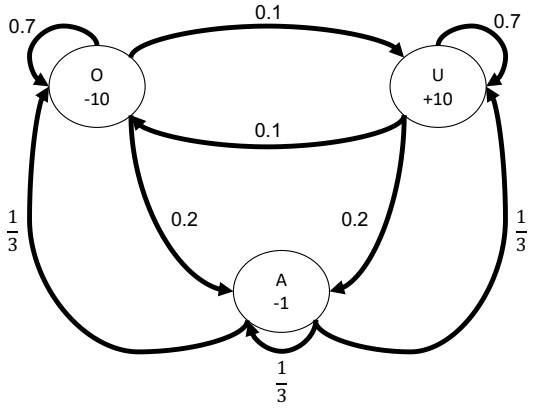

Figure 1: The Markov reward process.

2 Markov Decision Processes (80 pts)

Key building blocks:

- `env.get_actions(state)` function that returns a list of all possible actions given the state

Switches:

- `--env`, environment name. AI Farm is `aifarm_<prob>` where `<prob>` is the probability that the wind blows you to the right. For example `aifarm_0` is deterministic and `aifarm_0.1` is stochastic.
- `--gamma`, to change the discount (default=1.0)

2.1 Dynamic Programming (40 pts)

Implement the `dynamic_programming` function. Use any tabular dynamic programming algorithm to *exactly* compute the optimal state-value function and an optimal policy. You can run your code on the AI Farm using the homework script and visualize the state-value function and policy. **However, your code should be able to compute the optimal state-value function and optimal policy for *any* MDP with a finite number of states, where all states have a finite state-value, and a finite action space.**

Key building blocks:

- `env.state_action_dynamics(state, action)`: returns, in this order, the expected reward $r(s, a)$, all possible next states given the current state and action, their probabilities. Keep in mind, this only returns states that have a non-zero state-transition probability.
- `state_values`: you can obtain $V(s)$ with `state_values[state]`

Running the code:
`python run_hw1.py --task dp --gamma <discount_factor> --env <environment_name>`

2.2 Model-Free Reinforcement Learning (40 pts)

Implement the `model_free` function. Use any tabular model-free reinforcement learning algorithm to approximate the optimal action-value function. You can run your code on the AI Farm using the homework script and visualize the action-value function and policy obtained from behaving greedily with respect to said function. On the deterministic AI Farm, you should be able to find an optimal policy. For the stochastic cases, your policy may not always be optimal, but it should be close to optimal for most states, but it should be close to optimal. **However, your code should be able to estimate the optimal action-value function and for *any* MDP with a finite number of states, where all states have a finite action-value, and a finite action space.**

Important: In this exercise, you will be implementing a model-free algorithm and we will assume that we do not have access to the dynamics of the MDP. Therefore, in your implementation, you cannot use `env.state_action_dynamics(state, action)`.

Key building blocks:

- `env.sample_start_state()` function that returns a start state

- `env.sample_transition(state, action)`: returns, in this order, the next state and reward
- `action_values`: you can obtain $Q(s, a)$ with `action_vals[state][action]`

Running the code:
`python run_hw1.py --task mf --gamma <discount_factor> --env <environment_name>`

What to Turn In
Turn in your implementation of to `code_hw/code_hw1.py` to Blackboard.