CSCE 775: Deep Reinforcement Learning
Coding Homework 2
Due: 02/25/2025 at 11:58pm

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 7 minutes for all problems.
Hint: When training your DNN, make sure your tensors have the appropriate shape. For example, comparing a tensor of shape (N,) and shape (N,1) can give unexpected results.

Key building blocks:

- `env.sample_transition(state, action)`: returns, in this order, the next state and reward
- `env.get_actions(state)` function that returns a list of all possible actions that can be taken in that state
- `env.sample_start_states(num_states)` function that `num_state` start states
- `env.states_to_nnet_input(states)` converts a list of states to a numpy array for the input to a neural network
- `env.is_terminal(state)` returns true if state is terminal

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 Solving the 8-puzzle

Implement the `deep_rl` function. Given an 8-puzzle environment, train a value or policy function using a deep neural network using any reinforcement learning algorithm and return the trained neural network. You should not assume that you have a model of the MDP, so do not use `env.state_action_dynamics`.

Implement the `greedy_action` function. Given a state and your trained network, return a greedy action. The `run_hw2.py` will test your network on 500 test states by running your greedy policy for up to 100 steps. Your code should solve about 40% of them for the canonical 8-puzzle.

You should also test your code with the `--rand` switch, which will generate an MDP for the 8-puzzle with stochastic transitions. Your code should solve about 20% of test states for the randomly generated 8-puzzle MDPs. However, due to the random generation, some environments may be easier or harder.

Running the code:

```
python run_hw2.py
python run_hw2.py --rand
```

What to Turn In

Turn in your implementation of to `code_hw/code_hw2.py` to Blackboard.