Reinforcement Learning WS 2018/19
Instructor: Sebastian Blaes <sblaes@tuebingen.mpg.de>
Due-date: 30.11.2018 (hand in at beginning of recitation session)
Exercise Sheet 6
November 23, 2018

1 Model free Reinforcement Learning

This time we will consider the case where the MDP of the system is unknown. You will implement Q-Learning and test it on a simple Gridworld domain.
The code for this exercise contains needs two more files than the previous exercise. You can get all of them from `qlearning3.zip`:

Files:

agent.py The file in which you will write your agents (with small modification)

mdp.py Abstract class for general MDPs.

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

gridworld.py The Gridworld code and test harness. (with small modification)

crawler.py The crawler robot simulation code (not needed this time)

utils.py some utility code, see below.

The remaining files graphicsGridworldDisplay.py, graphicsCrawlerDisplay.py (not required), graphicsUtils.py, and textGridworldDisplay.py can be ignored entirely.
You will need to fill in portions of `agent.py` and modify `gridworld.py`.

Gridworld: Consult the previous exercise sheets for the general usage of the gridworld simulator. You also need the solutions from that exercise to compare to.

1.1 Qlearning

You will write a Q-learning agent, which does very little on construction, but then learns by trial and error interactions with the environment through its update(state, action, nextState, reward) method. A stub of a Q-learner is specified in QLearningAgent in agent.py, and you can select it with the option '-a q'. You should first run your Q-learner through several episodes under manual control without noise (e.g. '-k 5 -n 0 -m'), for example on the MazeGrid. Watch how

the agent learns about the state it was just in. Your actual agent should be an epsilon-greedy learner, meaning it chooses random actions epsilon of the time, and follows its current best q-values otherwise.
Written short questions:

(a) Train your Q-learner on the MazeGrid for 100 episodes.

```
python gridworld.py -g MazeGrid -a q -k 100 -q
```

How are the learned values different from those learned by value iteration (last exercise), and why? How can you make them closer to the optimal values?

(b) Train your Q-learner on the BridgeGrid with no noise (-n 0.0) for 100 episodes. How do the learned q-values compare to those of the value iteration agent? Why will your agent usually not learn the optimal policy?

(c) Train your Q-learner on the CliffGrid for 300 episodes. Compare the value it learns for the start state with the average returns from the training episodes (printed out automatically). Why are they so different?