Reinforcement Learning WS 2018/19
Instructor: Sebastian Blaes `<sblaes@tuebingen.mpg.de>`
Due-date: 07.12.2018 (hand in at beginning of recitation session)
Exercise Sheet 7
December 3, 2018

1 Q-Learning – Part II

This exercise builds on our last exercise. You will implement eligibility traces for Q-Learning and test it on the simple Gridworld domain and a crawling robot. Use the code from last week (`qlearning3.zip`) with your modifications. The files for the crawler experiment were already included (crawler.py, graphicsCrawlerDisplay.py).

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

1.1 Eligibility traces

You need an implementation of Q-learning from last week. If you did not manage or think your implementation is wrong, you can get inspiration from the solution provided on the webpage.
Add an implementation of eligibility traces. Add a new class by copying QlearningAgent and also wire it up in gridworld.py to be selectable from commandline with '-a qe'. Hint: the opts.agent=='q' is used several times, so make sure you change the code appropriately. Also add a parameter '–lambda' to the commandline to set the eligibility parameter λ The agent superclass has now a function reset which is called at the start of each episode.

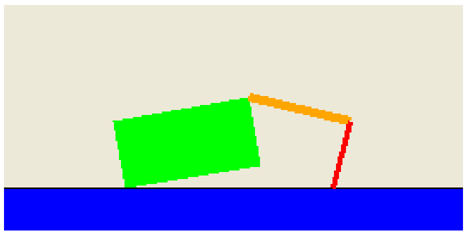
(a) check the values and Q-values after just 5 episodes with $\lambda = 0.5$ on the MazeGrid. `python gridworld.py -g MazeGrid -a qe --lambda=0.5 -k 5 -q` . Compare the result with version without the traces. What do you observe.

(b) Try the same with a larger value of λ. What do you observe and why?

(c) *Bonus:* On large grids, for instance, the normal ϵ-greedy exploration takes very long. Also in the BridgeGrid above we have seen that the agent does not explore very well. What would be done to improve the exploration? Try to implement it and test it on the Bridge and on a self made large grid.

1.2 Crawler Robot

In this part you will let a simple robot learn to locomote. The robot looks as follows:



Try the following command:

```
python crawler.py
```

This will invoke the crawling robot from class using your Q-learner. Play around with the various learning parameters to see how they affect the agent's policies and actions. Ensure that your Q-learner works in this non-episodic environment. Plot the learning curves.