Final Project: Implementation of PSRL and UCRL in the RLPy platform

Imanol Arrieta Ibarra

March 18, 2015

Abstract

RLPy is a python library that implements several Reinforcement Learning algorithms in an object-oriented environment. By incorporating UCRL and PSRL to this platform I am able to easily compare these algorithms across a great gamma of already implemented domains. In the following document I will compare PSRL, UCRL, SARSA and LSPI across two domains. The first is a classic MDP chain problem and the second one is the solution of a labyrinth.

1 MDP Chain

This experiment consists of a chain of 10 nodes. The agent starts in the first node and has to reach the 10th in order to get a reward of 1. Figure 1 shows how the different algorithms performed on this domain. Each line is the average over 5 different runs, the shadow area around the line represents the confidence interval. The graph is a little misleading, because of the exploring in PSRL and UCRL, the average of multiple runs will be lower than the optimum. If we were to take a single run of each algorithm all would typically end up converging to the optimum.

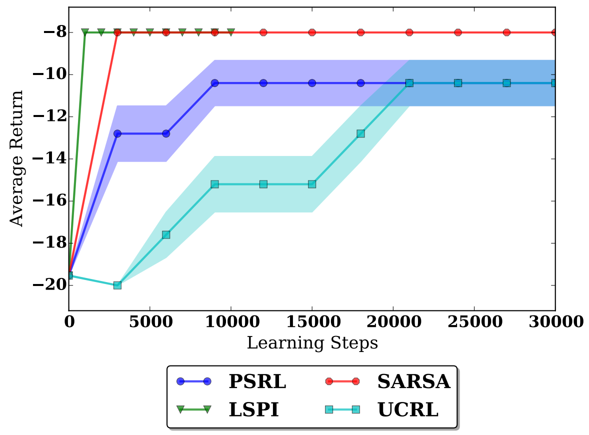

Figure 1: Average Return for the MDP chain

2 GridWorld

This experiment consists on a labyrinth with two pits and a way out. The starts at the beginning of the labyrinth and moves until the number of steps is maximum, it encounters a pit or it reaches the end. Figure 2 shows how the labyrinth looks like. The blue square corresponds to the starting point, the red ones to the pits, the gray ones to the walls and the green one to the exit.

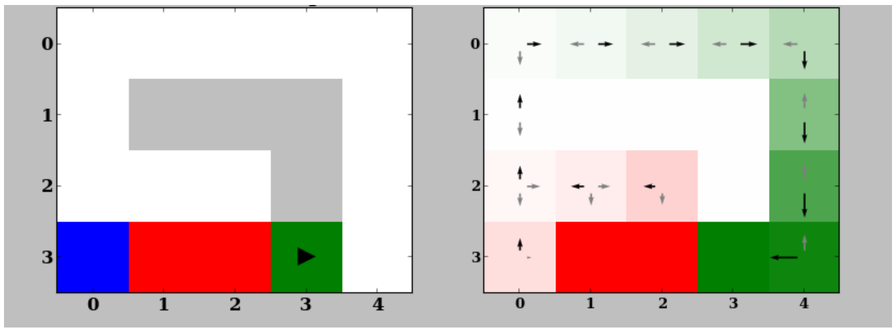

Figure 2: GridWorld labyrinth and example of policy

The adaptation of UCRL and PSRL to this setting was tricky because of the way RLPy is designed. I had to use some hacks in order to translate what the platform intended to do and what PSRL and UCRL need as input. Both UCRL and PSRL look pretty unstable in figure 3. This has to do in part with the stochasticity of the GridWorld problem

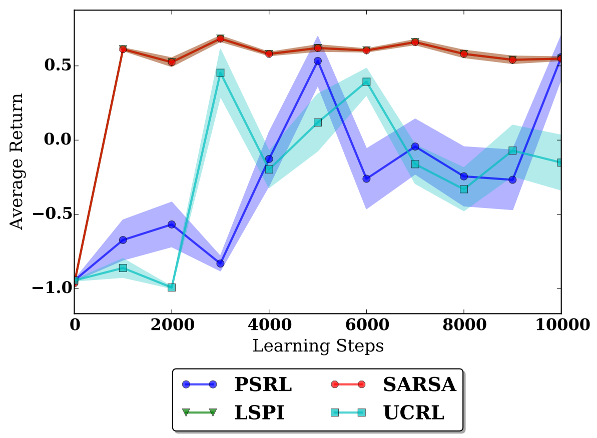

Figure 3: Average Return for GridWorld

3 Code

The code for this project can be found in `https://github.com/imanolarrieta/RL.git`. The files are located in: rlpy/Agents/UCRL and rlpy/Agents/PosteriorSampling.

4 Conclusions

These are some of the examples in which we would be able to test PSRL and UCRL. The only disadvantage and the reason I am not presenting any more is that because of Python limitations, the implementation of PSRL and UCRL can not be properly vectorized and as such runs extremely slow. However, if time is not a constraint, having this two algorithms in this platform provides great opportunity to compare them to Q-learning type algorithms.