Reinforcement Learning
Homework 3

Amine KHELDOUNI

December 16, 2018

1 On-Policy Reinforcement Learning with Parametric Policy

1.1 Monte Carlo Policy Gradient

For this first section, we consider the one-dimensional LQG environment with fixed standard deviation. We use a Policy Gradient method based on Monte Carlo simulations called the REINFORCE algorithm.

Running the algorithm validates its convergence to the optimal parameter value $\theta^* = -0.6$ (cf. 1). The drawback of such a method is its high variance. Therefore, we needed many samples before converging. Furthermore, we can improve the performance of the algorithm by adjusting the learning step with ADAGRAD method for instance, which takes into account the information about the gradient's variance. We also performed some parameter tuning, by setting $T \approx -\frac{log(\delta/R_{max})}{1-\gamma}$ (as seen in the first assignment), where $R_{max} \approx 2000$ is the upper bound of our reward, $\gamma = 0.9$ is the discount factor and δ taken equal to 5%. Therefore, we used $T = 106$ over $N = 50$ episodes and 500 iterations. Concerning the learning rate stepper, we implemented three update methods : the constant learning rate, a decreasing learning rate with the iterations (as in the Stochastic Gradient Decent) and an ADAGRAD stepper. Moreover, we notice a higher variance for the constant stepper which was expected given the fact that our confidence intervals are restrained for the ADAGRAD stepper.

To explain the effect of the learning rate α_t, we acknowledge that a static learning rate does not take into account the variance of our gradient and does not "learn" from the exploration, while an ADAGRAD stepper helps the algorithm converge faster with an adaptive learning rate that depends on the gradient's variance and contributes to its reduction. The value of the performance (average rewards over the iterations) converges towards -65 but increasing the number of iterations may have improved this value.

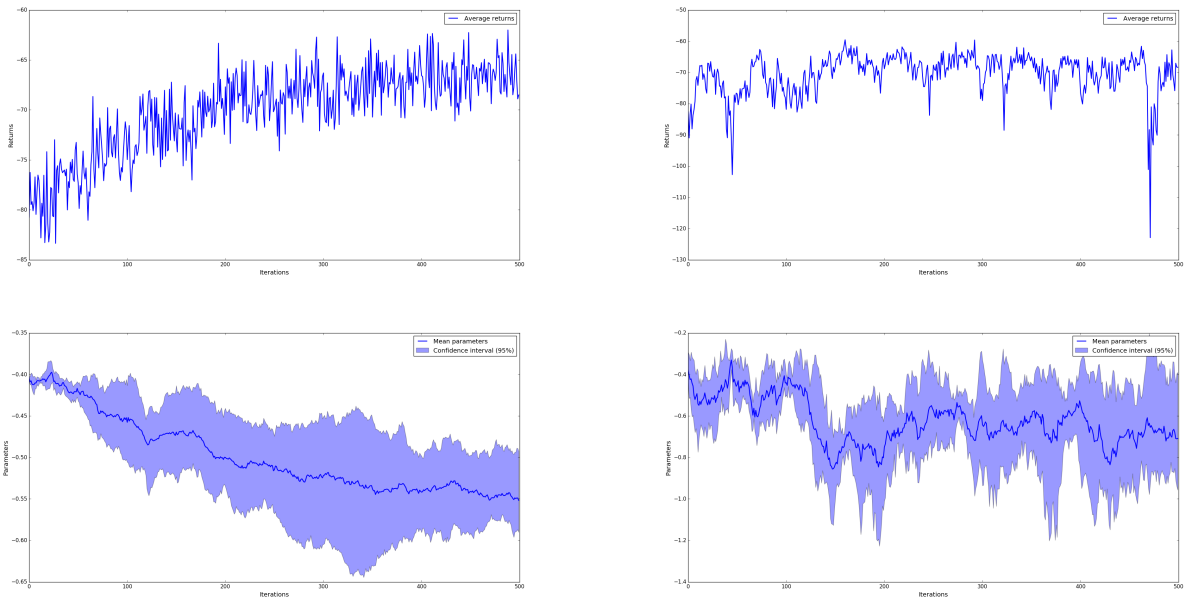


FIGURE 1 – Convergence of REINFORCE algorithm and average return over iterations for two different steppers (*left* : constant, *right* : ADAGRAD)

1.2 Exploration bonus in REINFORCE algorithm

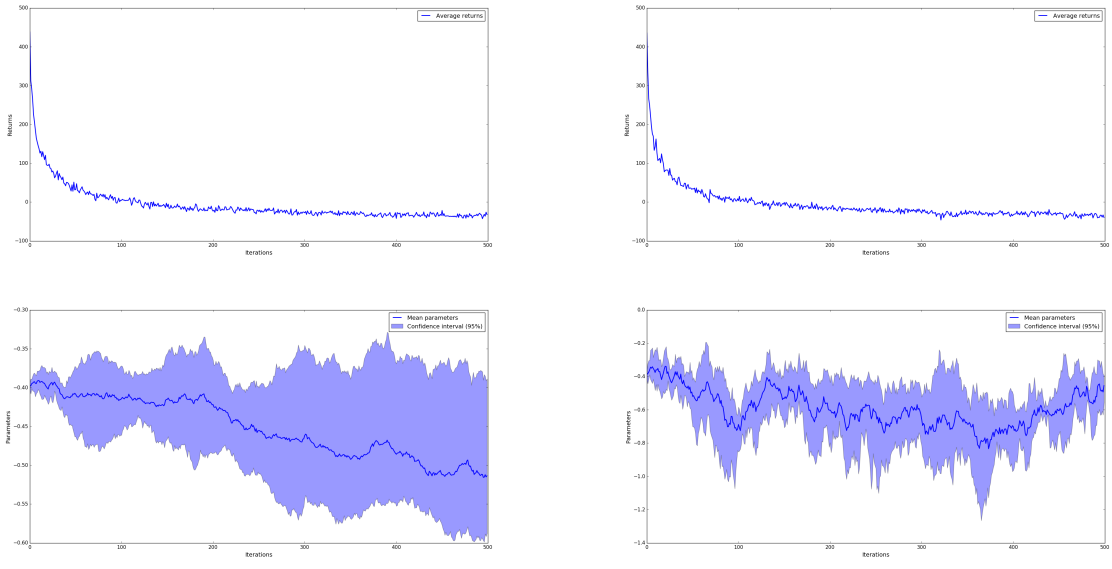


FIGURE 2 – Convergence of REINFORCE algorithm and average return over iterations when adding an exploration bonus (MBIE-BE) for two different steppers (*left* : constant, *right* : ADAGRAD)

To improve the performance of the REINFORCE algorithm, we implement the method of Strehl and

Littman (2008), adding an optimistic bonus term (MBIE-BE) to the reward (cf. 2). This bonus term enforces exploration toward unseen state and action areas. Above are the performance curves. We set $\beta = 1$ and used the same parameters as in previous section.

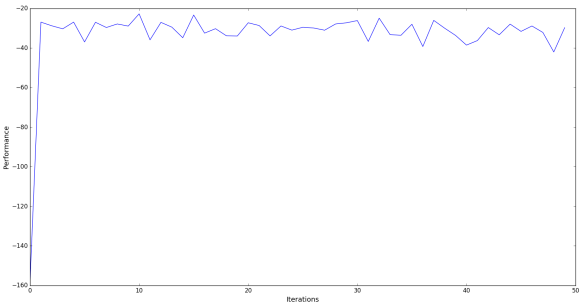
With the exploration bonus, the value of the performance (average rewards over the iterations) converges towards -30 even though it starts far away from this value (high bonus for the first iterations to favor exploration).

2 Off-Policy Reinforcement Learning with Value Function Approximation

In this second part of our report, we will tackle the *Fitted Q-Iteration* (FQI) algorithm which is an *off-policy* method that exploits samples collected in advance. After implementing the linear FQI method presented in the handout, we evaluate the performance of the algorithm by plotting the estimated performance over 50 episodes of length 50 at each iteration step.

Therefore, the implemented FQI algorithm should converge toward the optimal value of Q^* and builds a greedy policy after the iterations that creates a mapping between the discretized state set and action set.

As we expected, the performance estimation is coherent with the first part of the assignment which uses on-policy methods. Therefore, the performance value is converging toward -30 in average. We display in what follows the results of our FQI algorithm. The Q-value functions are really similar and the estimation computed with FQI iterations converges to the optimal Q-value $(state, action)$ mapping.



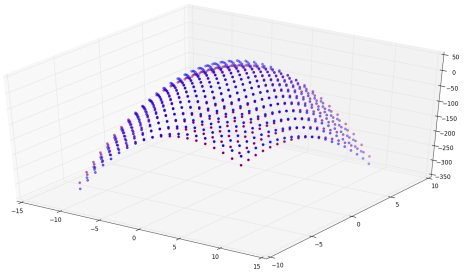


FIGURE 3 – Linear FQI algorithm for policy estimation and optimal Q-value computation