Reinforcement learning - MVA 2017/2018
Homework 2

Rémi Lespinet

1 On-Policy Reinforcement Learning with Parametric Policy

- Q1: If we condider the gaussian policy

$$\pi(a|s) = \frac{1}{\sigma_w(s)\sqrt{2\pi}} e^{-\frac{(a-\mu_\theta)^2}{2\sigma_w^2(s)}}$$

we can also write the derivative of the logarithm w.r.t w as

$$\nabla_w \log \pi(a|s) = \frac{(a-\mu_\theta(s))^2 - \sigma_w^2}{2\sigma_w^4(s)} \nabla_w \sigma_w^2(s)$$

This is the version that I use in my code (I define $\sigma^2(s)$ and $\nabla_w \sigma_w^2(s)$ directly).

1.1 Constant step

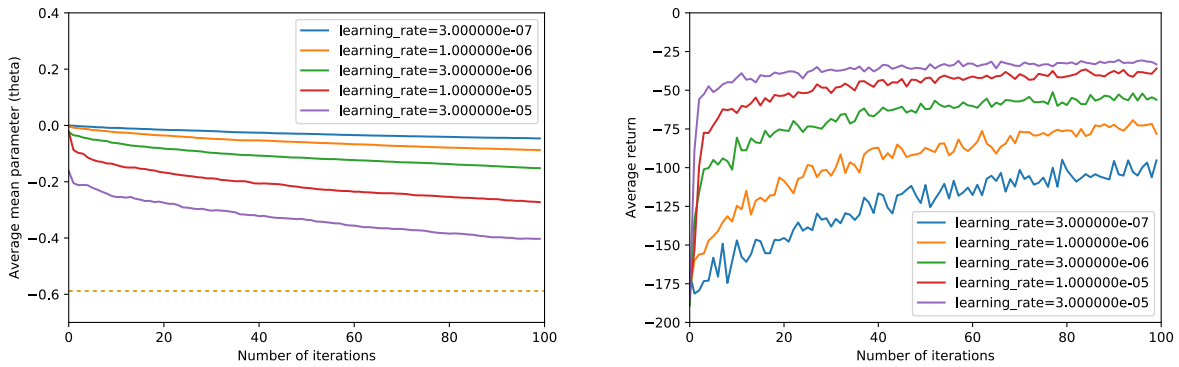
In this part, we use a constant step α. As we can see, high values of α make the computation of the estimated expected discount reward

$$\hat{\nabla}_\theta(J) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t^n | s_t^n) R(\tau^n)$$

diverge. Hence α must be chosen very carefully. The figure 1 presents the evolution of the mean parameter θ and the estimated policy performance as a function of the number of iteration for the REINFORCE algorithm with constant step for different values of the learning rate. In practice the highest values of the learning rate do not converge every time and I've restarted the simulation in order for them to work. These curves are averaged over 5 runs of the algorithm for $N = 100$ samples per estimation of the gradient of the policy performance, $T = 100$ (length of the trajectory) and the $\gamma = 0.9$.

In our case, a value of $\alpha = 10^{-5}$ seems to work well. This is not surprising to have such order for α, in our example, we are only interested in the gradient along theta, ($\sigma_w(s)$) is constant). If we look at an order of a single MC sample in our estimation of the expected discount reward, we see that is is expressed as

$$R(\tau) \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t | s_t) = \frac{a_t - \theta s_t}{\sigma_w(s)^2} R(\tau) s_t$$



(a) Average mean parameter θ

(b) Average return

Figure 1: Evolution of the mean parameter θ and the estimated policy performance as a function of the number of iteration for the REINFORCE algorithm with constant step for different values of the learning rate. These curves are averaged over 5 runs of the algorithm for $N = 100$, $T = 100$, $\gamma = 0.9$. The golden dashed line represents the optimal mean parameter value $\theta^* = -0.5884$

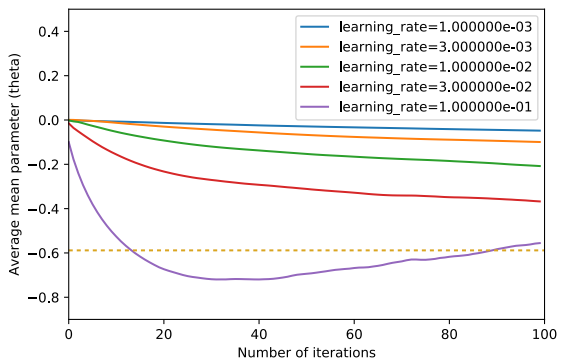
We know that a_t and s_t are on the order of 10^2, $\sigma_w(s)^2 = 0.25$, and $r_t = -\frac{s_t^2+a_t^2}{2}$ which is of the order of 10^4. Hence a single MC sample in the estimation of the expected discount reward should be of the order of 10^6. Since we expect values of θ to be on the order of 1, we expect the gradient ascend update to be of the order of 1

$$\theta_{t+1} = \theta_t + \alpha\hat{\nabla}_\theta(J)$$

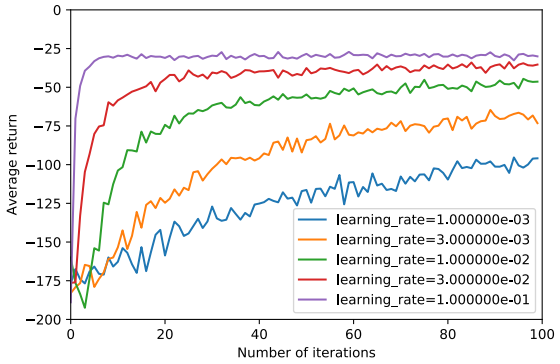
and we must then choose α of the order of 10^{-6}. In practice since the rewards are all negative, the goal of the algorithm is to maximize the rewards, which minimizes their absolute values and the order of the reward of a trajectory is around 10^3 (which explains why we can take sligly larger values for α).

1.2 Adam step

The figure 2 presents evolution of the mean parameter θ and the estimated policy performance as a function of the number of iteration for the REINFORCE algorithm with Adam step for different values of the learning rate (with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ as recommended). We see that this stepping strategy works really well in practice when compared to the other tested methods.



(a) Average mean parameter θ



(b) Average return

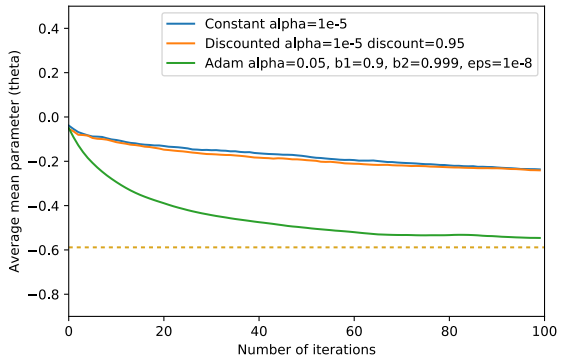
Figure 2: Evolution of the mean parameter θ and the estimated policy performance as a function of the number of iteration for the REINFORCE algorithm using the **Adam** stepping algorithm for different values of the learning rate. These curves are averaged over 5 runs of the algorithm for $N = 100$, $T = 100$, $\gamma = 0.9$. The golden dashed line represents the optimal mean parameter value $\theta^* = -0.5884$

1.3 Other strategies

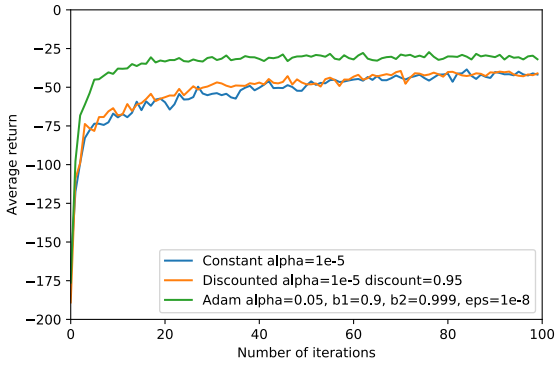
I've also implemented a discounted strategy, where

$$\alpha_t = \gamma^t \alpha_0$$

The figure 3 presents the different methods that we have seen.



(a) Average mean parameter θ

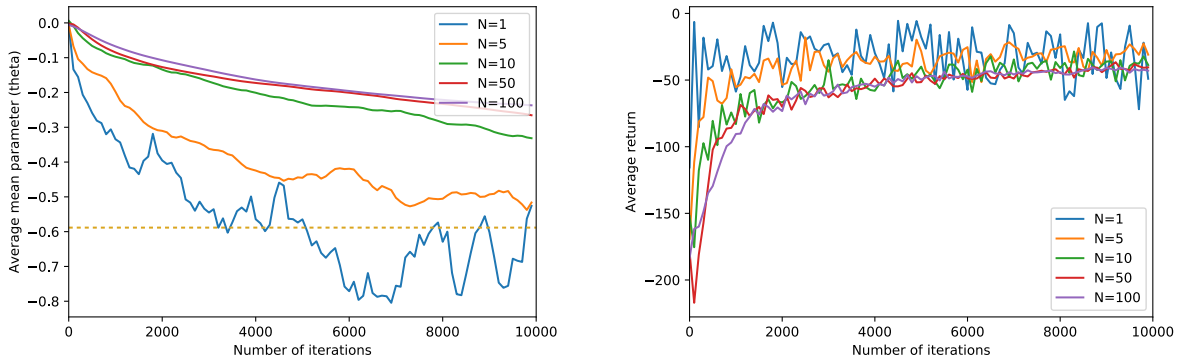


(b) Average return

Figure 3: Evolution of the mean parameter θ and the estimated policy performance as a function of the number of iteration for the REINFORCE algorithm for different stepping strategies. These curves are averaged over 5 runs of the algorithm for $N = 100$, $T = 100$, $\gamma = 0.9$

A lot of stepping strategy rely on the ability to compute the function to be optimized in several points (for example backtracking line search). In this case, we cannot compute the function easily, and it needs to be estimated by Monte Carlo sampling which make these stepping strategies relatively costly.

- Q2: We are interested in the behavior of the REINFORCE algorithm when the number of MC samples N vary. Figure 4 represents the evolution of the mean parameter θ and the estimated policy performance for different values of N as a function of the total number of trajectories computed NK (where K is the number of iteration of the algorithm e.g the number of time that the parameter θ is updated). We use the Adam stepping strategy with a learning rate of 0.1 to make this comparison.



(a) Average mean parameter θ

(b) Average return

Figure 4: Evolution of the mean parameter θ and the estimated policy performance as a function of the total number of iterations NK for the REINFORCE algorithm for different values of N. These curves are averaged over 5 runs of the algorithm an use Adam stepping strategy with $\alpha = 0.01$. $T = 100$, $\gamma = 0.9$

As we can see, when we estimate the gradient of the policy performance with a low number of sample, we obtain a very noisy approximation, but it allows to do a lot more policy parameter updates for a given amount of computation time.

2 Off-Policy Reinforcement Learning with Value Function Approximation

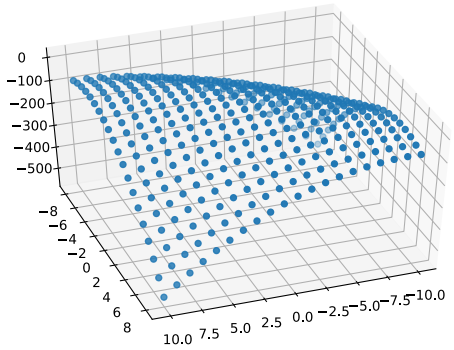
- Q3: The estimate state value function is parametrized by

$$\hat{Q}_\theta(s, a) = a\theta_1 + sa\theta_2 + (s^2 + a^2)\theta_3$$

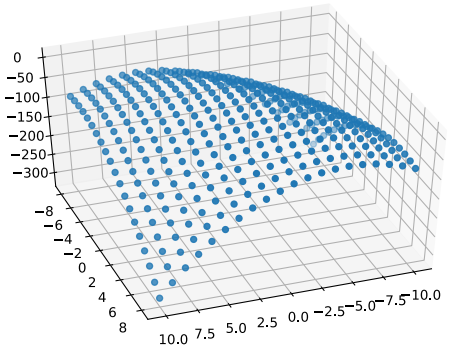
(see `LinearSecondOrderQModel` in the notebook)

As suggested, I've chosen a behavioral policy that choses actions uniformly in the action space (see `UniformPolicy`) to collect the dataset. I simulate 50 trajectories of length 50 using this policy. The figure 5 compares the state value function obtained (state and action have been discretized over a 20×20 grid), and the optimal state value function (provided). The figure 6 shows the evolution of the estimated policy performance learnt with the FQI algorithm as a function of the number of the iterations.

Note that at each iteration, we use the whole dataset collected, as the size of the dataset increases, it could be intractable, and we should use random subsets of the dataset.



(a) Representation of the Q function obtained by FQI



(b) Representation of the optimal Q function (provided)

Figure 5: Comparison of the state value function obtained with the optimal one (states and actions are discretized over a 20×20 grid

As we can see the FQI algorithm converges really fast and gives a good approximation of the state value function (in this case the optimal state value function is well described by a second order polynomial in (s, a)).

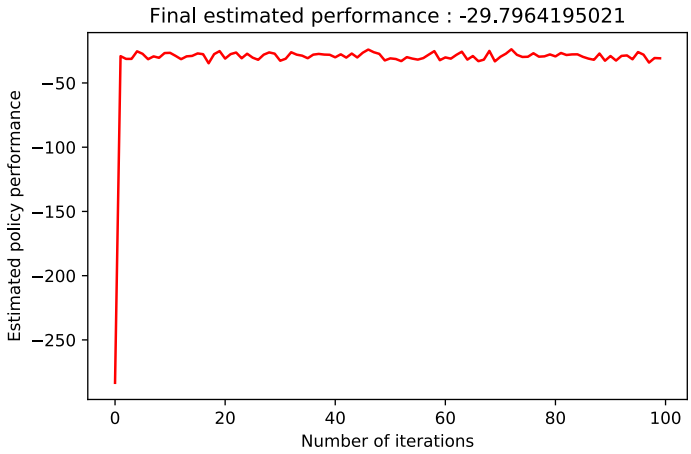


Figure 6: Representation of the performance of the policy as the number of iteration in the FQI algorithm increases.