Reinforcement Learning MVA 2018/2019 TP1

Louis GUO

Exercise 1: Dynamic Programming

Question 1

The discrete MDP model is implemented in the class dMDP in dMDP.py in folder Part1_dynamic_progamming. Examples are ran in jupyter notebook "Part 1 - Dynamic Progamming". As for the MDP represented in Figure 1, the reward given for action 2 at state 2 is 0.9, and particularly after such action, we stay at state 2 with probability 1. From state 2 and action 1, the reward is of 1 which looks appealing at first but can transition to state 1 where all actions have 0 reward. As well, state 0 's actions have very low rewards.

In all configurations, if we are at state 1, we would be better off transitioning to the state with highest value which is state 2, but it is different for the other states depending on γ:

- Intuitively, for high γ, we would guess that the optimal policy would be to join state 2 and loops infinitely on state 2 with action 2. Optimal policy would be : $[1, 1, 2]$
- However, if γ is low such that discount of rewards becomes very heavy with time, if we are at state 0, we may want to loop on state 0 because the cost of moving to state 2 is too high. As well for state 2, with higher discount, we will want to target action 1 which has a higher reward on the moment compared to action 2. Optimal policy would be : $[2, 1, 1]$ or $[2, 1, 2]$

Question 2

We implement and run value iteration to identify a 0.01-optimal value function. We use stopping criterion $\epsilon = \frac{0.01}{2}\frac{1-\gamma}{\gamma}$ with $||v^{k+1} - v^k||_\infty \leq \epsilon$ such that we have $||v^{\pi^+_{k+1}} - v^*||_\infty \leq \frac{2\epsilon\gamma}{1-\gamma} = 0.01$

We plot in Figure 1 $||v^k - v^*||_\infty$ as a function of number of iterations. It is a strictly decreasing function. In fact, $||v^k - v^*||_\infty = ||Tv^{k-1} - Tv^*||_\infty \leq \gamma||v^{k-1} - v^*||$ and $\gamma < 1$. Also, we see that with at most 160 iterations (in practice, it was a lot less), we can identify a 0.01-optimal value function. By policy extraction, and policy evaluation, for $\gamma = 0.95$ we find optimal policy: $\boxed{\pi^* = [1, 1, 2]}$ and $\boxed{v^* = [15.39, 16.55, 18.00]}$

Question 3

We implement and run policy iteration with initial policy $\pi_0 = [0, 0, 0]$. The stopping criterion is when two consecutive policies have same value function.

We plot in Figure 1 $||v^k - v^*||_\infty$ and it reaches the exact optimal policy in 4 iterations with the same policy as found in Q2.

Policy iteration is based on the idea that policy reaches optimal policy in a finite number of iterations (4 steps here) when value iteration convergence is only asymptotic: policy converges faster than value function. In value iteration, policy may have converge long before the value function which means a lot of effort is wasted.

However, at each step of policy iteration, policy evaluation then policy improvement are needed. Evaluation in the implementation inverses the Bellman linear equations which is done in time complexity $O(N^3)$. Improvement is of time complexity $O(N|A|)$. Value iteration is less computationally expensive at each step, with a time complexity of $O(N^2|A|)$ at each step.

Exercise 2: Reinforcement Learning

The code can be ran using whether with main_tp1_ex2.py script or with jupyter notebook "Part 2 - Reinforcement Learning"

Question 4

We want to evaluate to policy that selects the action right when available, otherwise up. For policy evaluation, we use Monte-Carlo estimator to estimate to value function (seen as an expectation of trajectory discounted reward) where each sample is a trajectory from start state drawn from the distribution. We only implement first-visit Monte-Carlo estimator: each trajectory only updates the value of the start state.

For updating the value function at each new episode, we use recursive updates:

- $V_{n+1}(s) = \frac{1}{N(s)+1}(N(s)V_n(s) + R_{N(s)+1})$ if at n+1 episode, we draw start state s.
- For $\tilde{s} \neq s$, we keep previous value: $V_{n+1}(\tilde{s}) = V_n(\tilde{s})$

This estimator is unbiased and we could check its consistency by computing asymptotic confidence interval with central limit theorem or non-asymptotic one with concentration inequalities. Here we just compute the estimator several times to have a catch at its consistency. As a check of convergence, we plot $J_n - J^\pi$ as a function of n, where J can be seen as the value of the game, taking into account the randomness of starting state as well (we estimate start state distribution also with MC estimation). The plot (Figure 2) confirms the convergence induced by the law of large numbers. As the number of iterations is high, the estimate is less and less noisy as we can see that the difference estimations are closer and closer to each other.

Question 5

Several parameters have to be chosen such as:

- T_{max} max length of episode: chosen as $T_{max} = O(-log(\delta)/(1-\gamma))$
- ϵ parameter of the greedy exploration: For large ϵ, we prioritize exploration. But when the Q-function starts to be well-estimated, it can be intersting to decrease the exploration to favour exploitation.
- Learning rates α_i: decreasing speed parametrized by a power law $\alpha_i(s,a) = \frac{1}{N(s,a)^k}$. For large powers k, the decreasing of the learning rates happens very fast such that the marginal contribution of new episode becomes less important which tends to slow down the learning (we still have convergence by theorem). We can choose slowly-decreasing learning rates which still satisfy the Robbins-Monro conditions for Q-function convergence), for example $k \in]0.5, 1]$

In Figure 4 and 5, we see that as ϵ is small, the learning of the optimal policy is slowed. Likely, when the power law is slowly decreasing (high value of k), the learning is also slowed. In Figure 6, we see that the choice of high ϵ disfavours exploitation as the average level of reward reached for each trajectory converges to an underestimate of the optimal global value (plotted as the red dotted line) of the game.

This illustrates the dilemma between exploration and exploitation. One solution would be to decrease ϵ with time when the Q-function is already learned well.

Question 6

The optimal policy of an MDP is not affected by the change of the initial distribution μ_0. The optimal policy of an MDP and its optimal value function are defined by optimal Bellman equation in which the initial distribution μ_0 isn't used. However, the initial distribution can have an effect on the process of learning the optimal Q-function. The states for which start distribution probability is low, it may take a longer time to estimate the Q-function on this state.

Figures

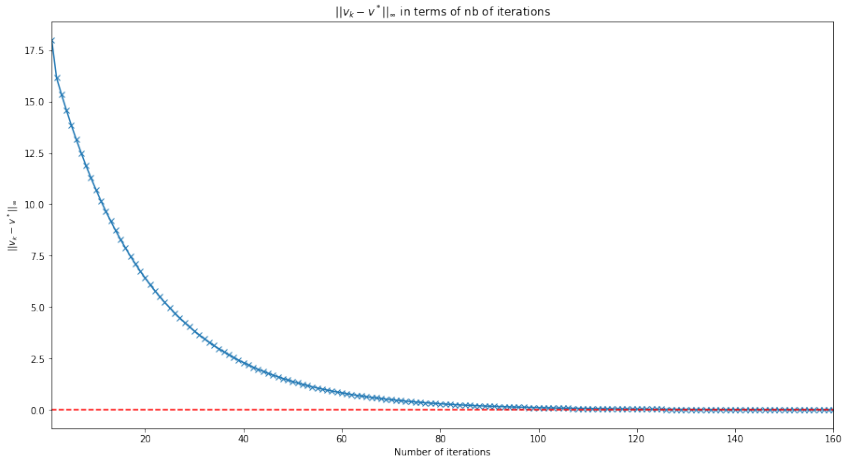

Figure 1: (Q2) L_∞ convergence of value function in Value Iteration

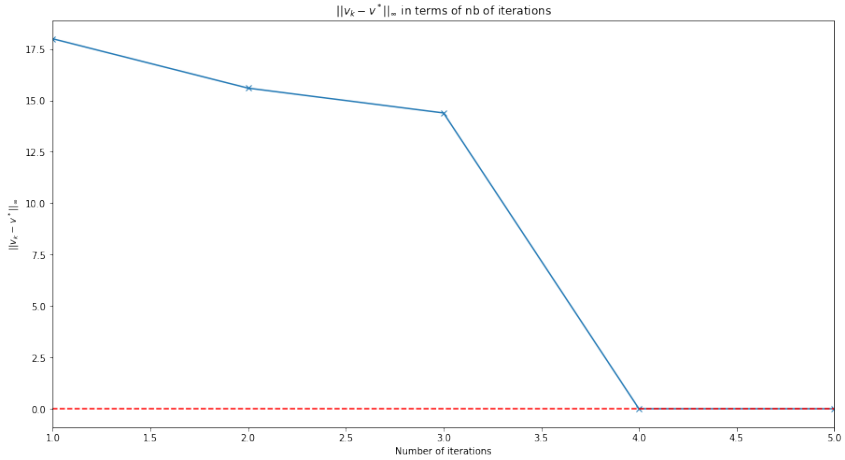

Figure 2: (Q3) L_∞ convergence of value function in Policy Iteration

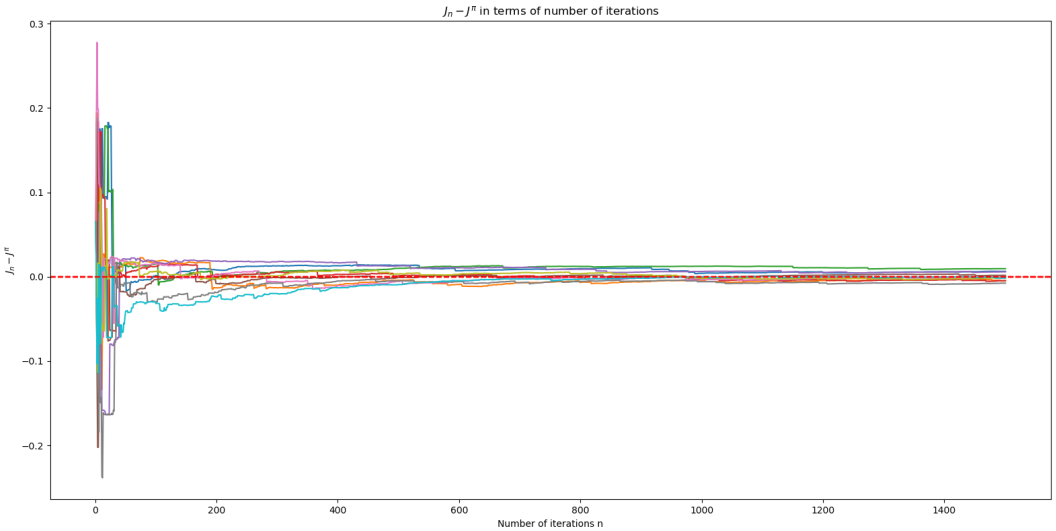

Figure 3: (Q4) 10 Policy evaluations and computation of error made in global value estimation

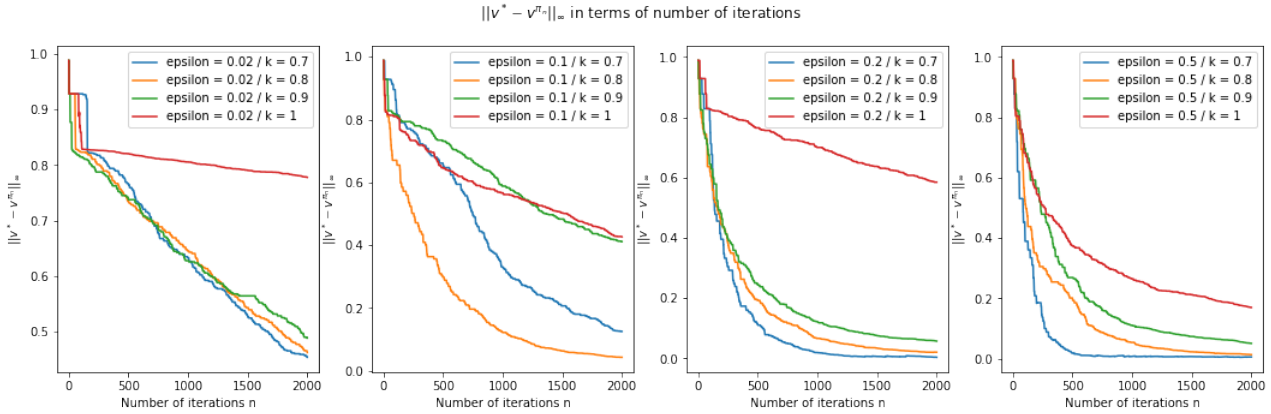

Figure 4: (Q5) L_∞ convergence of value function in policy evaluation for different ϵ and power k

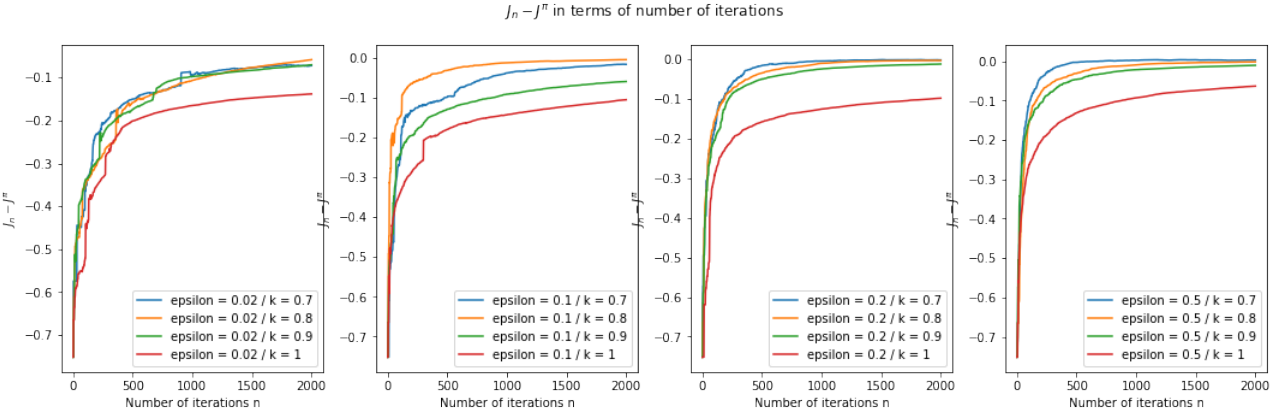

Figure 5: (Q5) Estimation error of global value for different ϵ and power k

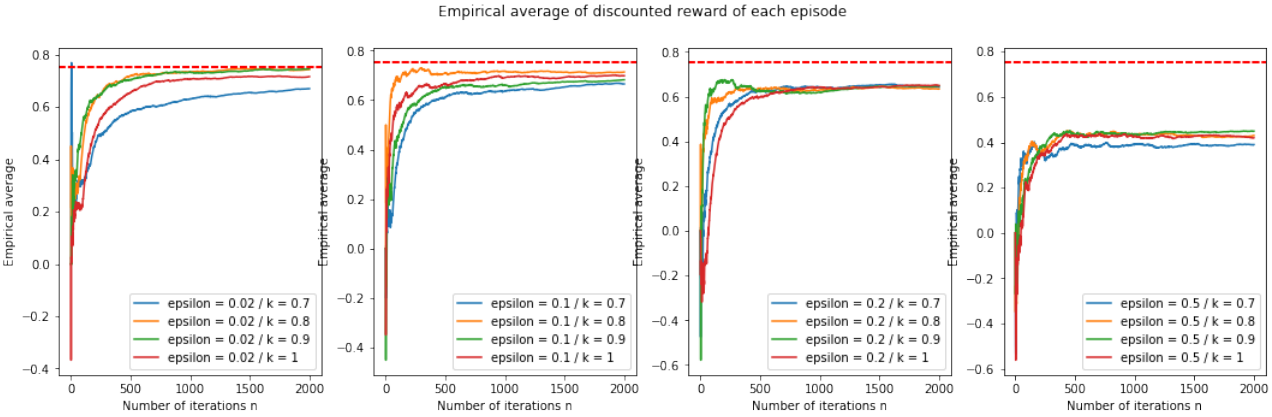

Figure 6: (Q5) Empirical average of discounted reward, compared to optimal global value for different ϵ and power k