National Taiwan University
Department of Electrical Engineering
Name: LEI WA KIN
ID: R11922001

112-1 (Fall 2023)
Reinforcement Learning

Assignment 2 Report

Q1. Discuss and plot learning curves under ϵ values of $(0.1, 0.2, 0.3, 0.4)$ on MC, SARSA, and Q-Learning

Figure 1 plots the learning curves for different ϵ values under MC, SARSA and Q-Learning. With the constant ϵ-greedy policy, $\epsilon = 0.1$ performs highest non-discounted episode reward among 4 candidates ϵ values after the episode reward is saturated. This is because the agent tends to do random walk, which leads lower episode reward, when ϵ is large.

It is worth noted that SARSA has the highest reward among three algorithms, then Q-Learning, finally MC. See Figure 2 for the comparison.

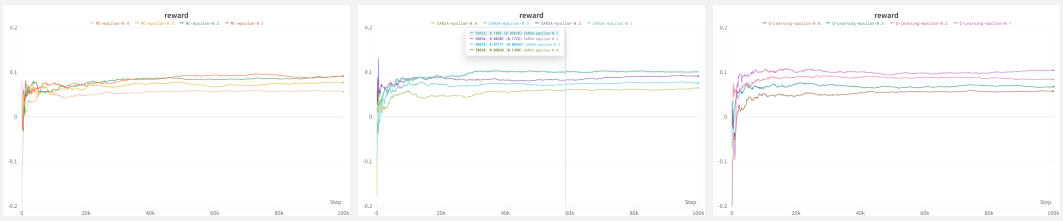

Figure 1: Running average non-discounted episode reward in MC, SARSA, and Q-Learning. The curves are smoothed by EMA($\alpha = 0.999$) for better readability.

Figure 2: Comparing the running average non-discounted episode reward in MC, SARSA, and Q-Learning. Takes $\epsilon = 0.2$. The curves are smoothed by EMA($\alpha = 0.999$) for better readability.

Q2. Discuss and plot loss curves under ϵ values of $(0.1, 0.2, 0.3, 0.4)$ on MC, SARSA, and Q-Learning

Estimation loss refers to time-to-convergent. In MC and SARSA, when ϵ is large implies that the agent has larger estimation error. In contrast, the estimation loss is smallest when $\epsilon = 0.4$ in case of Q-Learning.

Furthermore, It is worth noted that MC has the largest estimation error among three algorithms given the same number of episodes. This is because the definition of estimation loss between MC and SARSA are different. It is defined as difference to total return $(G_t - Q(s_t, a_t))$ in MC, but one-step error $(r + Q(s', a') - Q(s, a))$ in SARSA. See Figure 4 for the comparison.

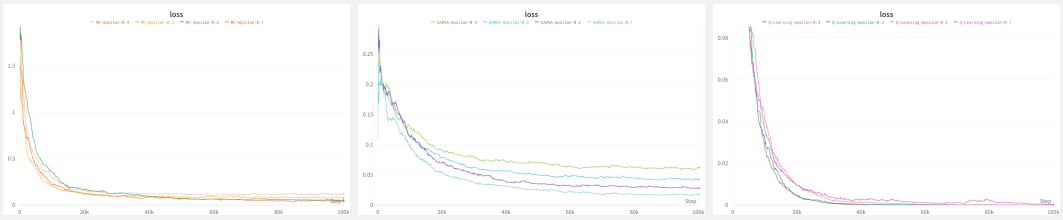

Figure 3: Running average absolute estimation loss in MC, SARSA, and Q-Learning. The curves are smoothed by EMA($\alpha = 0.95$) for better readability.

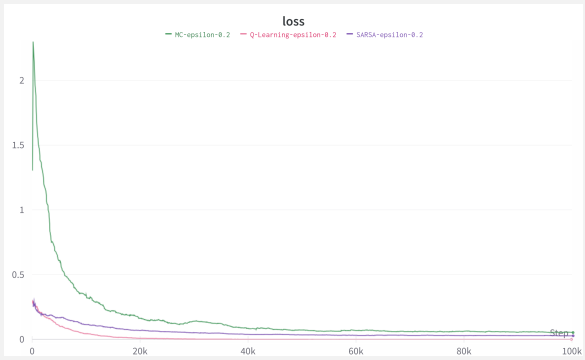

Figure 4: Comparing the running average absolute estimation loss in MC, SARSA, and Q-Learning. The curves are smoothed by EMA($\alpha = 0.95$) for better readability.

Q3.

In the following discussion, I will first describe the effect of the parameters designed for Q-Learning algorithm, which include 1. buffer size, 2. update frequency, and 3. sample batch size in Q3.1. After that, I will describe those are generic to the setting of MDP/RL problems, which include 1. step reward, 2. discount factor, and 3. learning rate.

To keep the discussion concise, all the following discussions and experiments are based on the context of Q-Learning algorithm. Also check Table 1 for the details.

Q3.1 Discuss (and plot) different settings in terms of buffer size, update frequency, and sample batch size

The concept of Q-Learning is based on bootstrapping from memory buffer. By alternating these three parameters, I found that they mainly affected the estimation loss. I suggest that except the extreme cases (e.g., 1. Extreme small memory buffer, which damages the variance of batch, or 2. Extreme slow updating frequency, which only explore but not exploit), the Q-Learning agent can learn and perform well in GridWorld with a broad set of hyper-parameters.

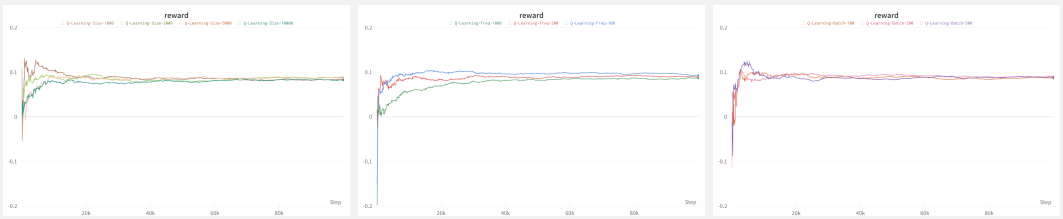

Figure 5: Running average non-discounted episode reward in Q-Learning. The curves are smoothed by EMA($\alpha = 0.999$) for better readability.

Figure 6: Running average absolute estimation error in Q-Learning. The curves are smoothed by EMA($\alpha = 0.95$) for better readability.

Q3.2 Discuss (and plot) different settings in terms of step reward, discount factor, learning rate

To solve the maze-like problem with MDP/RL, the reward should be carefully tuned. Basically user should define a large reward for arriving the goal, a large penalty for being trapped, and a small penalty for traversing in maze. This encourages agent to escape the maze earlier, and also avoid the traps. In contrast, agent will decide to fall in trap if the goal reward is smaller than the trap penalty plus total step penalty from current state to goal state.

Discount factor represents the uncertainty to long-term goal. In the deterministic environment, using $\gamma \to 1$ is considered reasonable decision.

Learning rate affects both time-to-convergence and stability of algorithm. Set a high learning rate $\alpha \to 1$ leads model to forget the previous evaluation and adapt to new observation earlier, which might push agent to have unstable state estimation. In contrast, setting a low learning rate leads more time to converge. Last but not least, I suggest that user should tune learning rate with considering the randomness of policy (e.g., using ϵ-greedy with constant ϵ).

Table 1 concludes the parameters I have tried in the experiments. For each experiments, only a parameter is modified, others are kept as default. Also, the default value are highlighted by bold font, which is the same as the parameters provided in initial code base and slides.

Parameter	Range
Algorithm	Q-Learning
Number of Episodes	5×10^4
ϵ	0.2
γ	$(0.9, \mathbf{0.99}, 0.999)$
α	$(10^{-1}, \mathbf{10^{-2}}, 10^{-3})$
Step reward	$(-0.05, \mathbf{-0.10}, -0.15, -0.20)$
Goal reward	1.00
Trap reward	-1.00
Buffer size	$(1000, 2000, 5000, \mathbf{10000})$
Update frequency	$(100, \mathbf{200}, 500)$
Sample batch size	$(100, 200, \mathbf{500})$

Table 1: Parameters and settings applied in Q3.

Appendix

1. I decided to smooth the curves before plotting it, because the initial state of each episode is different, which affects the optimal return for the episode. Take average from last 10 episodes is not enough to cancel out the variance of optimal non-discounted return, thus using a strong smoothing function is considered.

2. Accidentally, I log the episode reward and loss in separate two function calls, which WandB treats these logs are from different epoch. Therefore, to interpret the relation between the metrics and steps, the scale of x-axis should be scaled down by a factor of 0.5×.