

**Name:** Sungjoon Park

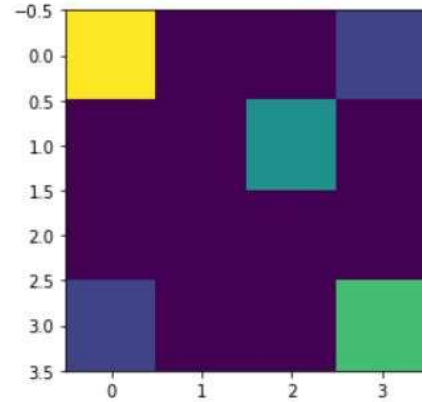
**Major (Track):** Engineering Science MS (Data Science)

**UB Person Number:** 50419347

## CSE 546 Assignment 1 Checkpoint

1. Environments (Deterministic and Stochastic Environments in common)
  - a. set of actions:  $\{up, down, left, right\}$
  - b. the number of states: 16 (a 4 by 4 gridworld)
  - c. rewards:  $\{0, -0.5, 0.5, 1\}$
  - d. main objective: to arrive at the terminal state within a given time limit (15)
2. Visualizations of Each Environment

0 (starting)	1	2	3 (-0.5)
4	5	6 (+0.5)	7
8	9	10	11
12 (-0.5)	13	14	15 (terminal , +1.0)



### Deterministic Environment

$p(1, 0|0, right) = 1, p(4, 0|0, down) = 1, p(0, 0|0, up) = 1, p(0, 0|0, left) = 1,$   
 $p(2, 0|1, right) = 1, p(5, 0|1, down) = 1, p(1, 0|1, up) = 1, p(0, 0|1, left) = 1,$   
 $p(3, -0.5|2, right) = 1, p(6, 0.5|2, down) = 1, p(2, 0|2, up) = 1, p(1, 0|2, left) = 1,$   
 $p(3, -0.5|3, right) = 1, p(7, 0|3, down) = 1, p(3, -0.5|3, up) = 1, p(2, 0|3, left) =$   
 $1,$   
 $p(5, 0|4, right) = 1, p(8, 0|4, down) = 1, p(0, 0|4, up) = 1, p(4, 0|4, left) = 1,$   
 $p(6, 0.5|5, right) = 1, p(9, 0|5, down) = 1, p(1, 0|5, up) = 1, p(4, 0|5, left) = 1,$   
 $p(7, 0|6, right) = 1, p(10, 0|6, down) = 1, p(2, 0|6, up) = 1, p(5, 0|6, left) = 1,$   
 $p(7, 0|7, right) = 1, p(11, 0|7, down) = 1, p(4, -0.5|7, up) = 1, p(6, 0.5|7, left) =$   
 $1,$

$$\begin{aligned}
&p(9, 0|8, right) = 1, p(12, -0.5|8, down) = 1, p(4, 0|8, up) = 1, p(8, 0|8, left) = 1, \\
&p(10, 0|9, right) = 1, p(13, 0|9, down) = 1, p(5, 0|9, up) = 1, p(8, 0|9, left) = 1, \\
&p(11, 0|10, right) = 1, p(14, 0|10, down) = 1, p(6, 0.5|10, up) = 1, \\
&p(9, 0|10, left) = 1, \\
&p(11, 0|11, right) = 1, p(15, 1|11, down) = 1, p(7, 0|11, up) = 1, \\
&p(10, 0|11, left) = 1, \\
&p(13, 0|12, right) = 1, p(12, -0.5|12, down) = 1, p(8, 0|12, up) = 1, \\
&p(12, -0.5|12, left) = 1, \\
&p(14, 0|13, right) = 1, p(13, 0|13, down) = 1, p(9, 0|13, up) = 1, \\
&p(12, -0.5|13, left) = 1, \\
&p(15, 1|14, right) = 1, p(14, 0|14, down) = 1, p(10, 0|14, up) = 1, \\
&p(13, 0|14, left) = 1
\end{aligned}$$

### 3. Stochastic Environment

In the 1.2 Stochastic Environment, for each state and action, the next state is determined by a high probability (0.85) as intended. However, the agent moves to an unintended state with the remaining 0.15 probability. Specifically, it moves to the three other directions respectively with probability of 0.05.

#### a. Example 1

$$- p(1, 0|0, right) = 0.85, p(0, 0|0, right) = 0.1, p(4, 0|0, right) = 0.05$$

#### b. Example 2

$$\begin{aligned}
&- p(6, 0.5|10, up) = 0.85, p(9, 0|10, up) = 0.05, p(11, 0|10, up) = 0.05, \\
&p(14, 0|10, up) = 1
\end{aligned}$$

4. Unlike deterministic environments in which the next state and reward are clearly predicted given the present state and action, basically, either is randomly determined with some probability in stochastic environments even though an agent has already decided to select a move for the next state.

### 5. Safety in AI

- a. If an action that has the agent go outside from the edges of the grid is taken, it is enforced to stay on the edges of the grid in order to allow the simulation to be repeated only within the given environment.
- b. This measure is reflected in the GridEnvironment class using numpy's clip function.

## CSE 546 Assignment 1 Part 2

Please read below.

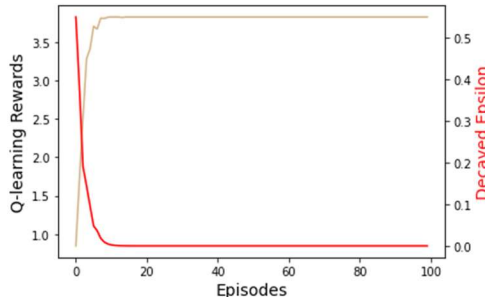
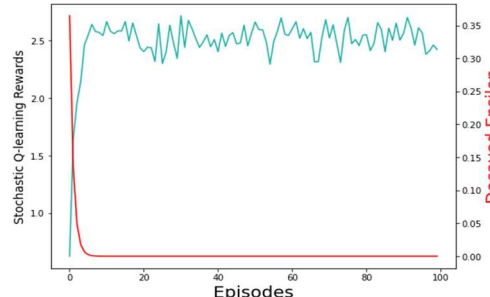
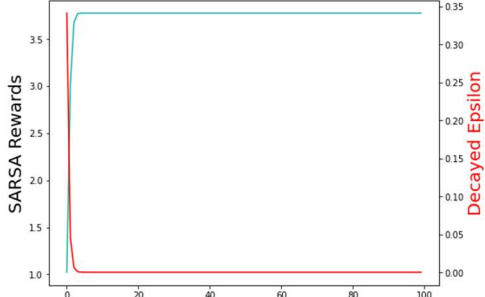
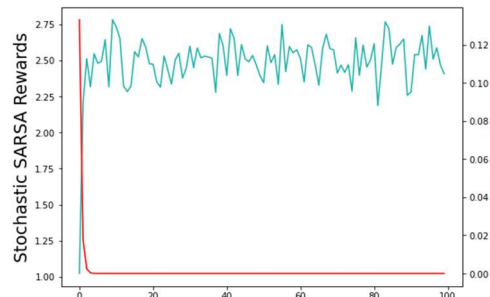
□ Revision made from the previous checkpoint

- ✓ I deleted the cell using google.colab module for visualization. Instead, I use matplotlib.pyplot ax.imshow(). This is because it is inconvenient for me to work on Google colab. I failed to install it for several times. I prefer to work on my local PC. (I have tried to resolve the problem using CCR. So I consulted the person in charge but it has not been resolved in the end.) For this reason, every cell which depended on google.colab module has been changed.
- ✓ I replace the original reward in the GridEnvironment class. This is because the rewards are too similar to each other to be compared (the performance of each hyperparameter setting).

1. I chose SARSA in addition to Q-learning.

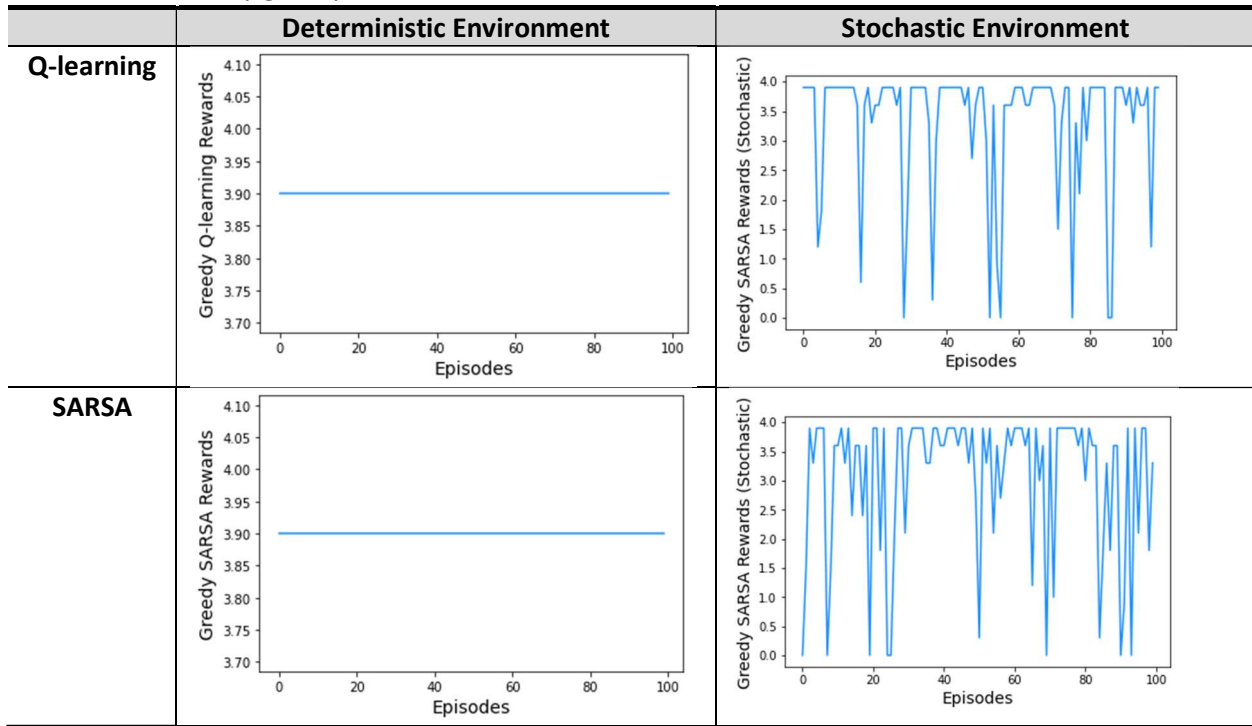
a. Applying decaying epsilon

When 100 episodes averaged (each episode's length 100), for each setting

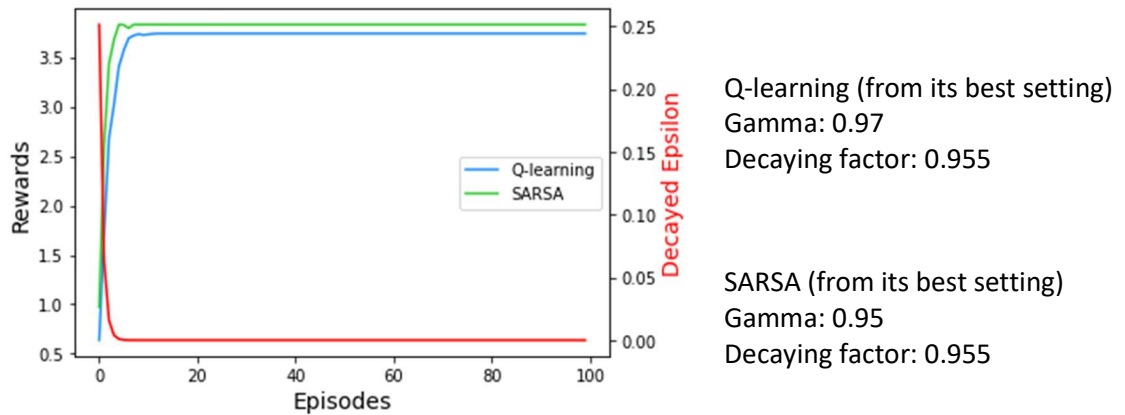
	Deterministic Environment	Stochastic Environment
<b>Q-learning</b>		
	Hyperparameter Tuning - Best setting <sup>1</sup> Gamma: 0.97 Decaying factor: 0.955	No significant difference among various settings (Gamma: 0.99, Decaying factor: 0.935)
<b>SARSA</b>		
	Hyperparameter Tuning - Best setting Gamma: 0.95 Decaying factor: 0.955	No significant difference among various settings (Gamma: 0.99, Decaying factor: 0.935)

<sup>1</sup> Gamma = (0.95, .97, 0.99), Decaying factor = (0.915, 0.935, 0.955)

b. Only greedy action

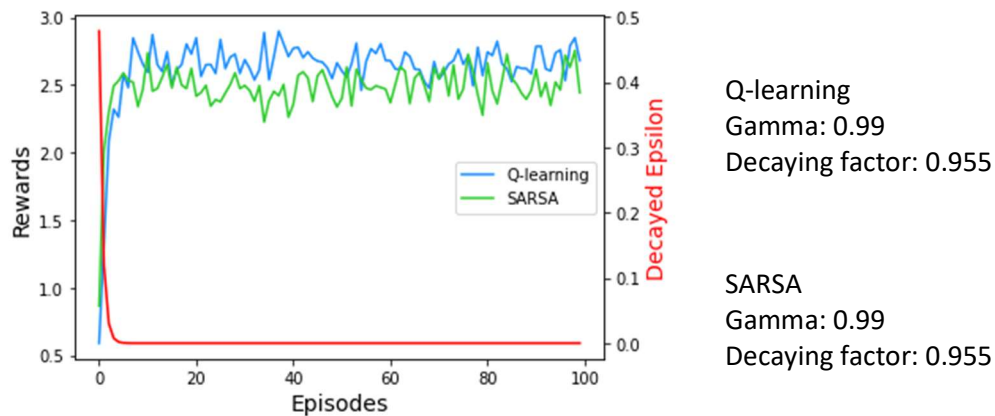


- Compare the performance of both algorithms on the same deterministic environment and give your interpretation of the results.



The SARSA algorithm shows better performance than the Q-learning in terms of size of rewards whereas two algorithms converge almost at the same time of a given episode. The SARSA could perform better than the Q-learning since the SARSA's gamma is smaller than that of the Q-learning. Note that the larger gamma, the more weight it puts on future rewards. The SARSA's larger gamma helped the algorithm do more exploration than otherwise, which in turn gave the SARSA more opportunities to improve its policy. As can be seen from the result of 'Only greedy policy' in question 1, considering that both algorithms found the same optimal policy, the difference in performance between two algorithms seems to be due to the difference in gamma.

- Compare how both algorithms perform in the same stochastic environment and give your interpretation of the results.



The Q-learning performs slightly better than the SARSA unlike the case in the deterministic environment. First of all, the cause can be found in the fact that the gamma of the two algorithms became the same, unlike the answer to question 2. In addition, SARSA algorithms take the next action and reward into account to improve policy, which is different from Q-learning algorithms. Therefore, it is estimated that the SARSA's performance was relatively low as it reflects random noise occurring in the current state's reward as well as that in the next state's reward for policy improvement.

- Briefly explain the tabular methods, including Q-learning, that were used to solve the problems. Provide their update functions and key features.

Q-learning and SARSA are typical model-free algorithms for Temporal Difference control. The former is for the off-policy TD control whereas the latter is for the on-policy TD control. That is, SARSA uses the current estimate for learning optimal policy, but Q-learning directly approximates the action value function. Each algorithm updates as follows.

Algorithm	Updating Functions
Q-learning	TD(0) case $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t))$
SARSA	Sample value SARSA: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$ Or Expected SARSA: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1})   S_{t+1}] - Q(S_t, A_t)]$ $= Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \sum \pi(a   S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t) \right]$