COMP9414 23T2
Artificial Intelligence

Assignment 1 - Reward-based learning agents

Due: Week 5, Friday, 30 June 2023, 11:55 PM

1 Activities

In this assignment, you are asked to implement a modified version of the temporal-difference method Q-learning and SARSA. Additionally, you are asked to implement a modified version of the action selection methods soft-max and ϵ-greedy.

To run your experiments and test your code you should make use of the example gridworld used for Tutorial 3 (see Fig. 1). The modification of the method includes the following two aspects:

- Random numbers will be obtained sequentially from a file.
- The initial Q-values will be obtained from a file as well.

The random numbers are available in the file `random_numbers.txt`. The file contains 100k random numbers between 0 and 1 with seed = 9999 created with numpy.random.random as follows:

```
import numpy as np

np.random.seed(9999)
random_numbers=np.random.random(100000)
np.savetxt("random_numbers.txt", random_numbers)
```

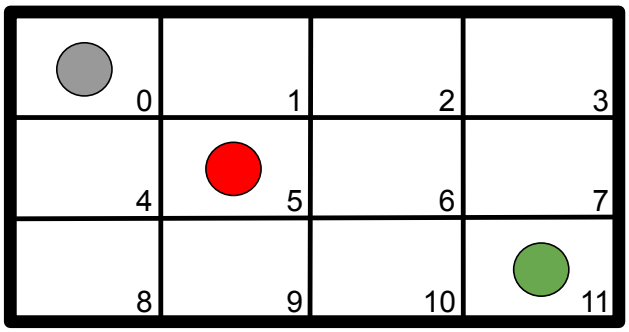


Figure 1: 3×4 gridworld with one goal state and one fear state..

1.1 Implementing modified SARSA and ϵ-greedy

For the modified SARSA you must use the code review during Tutorial 3 as a base. Consider the following:

- The method will use a given set of initial Q-values, i.e., instead of initialising them using random values the initial Q-values should be obtained from the file `initial_Q_values.txt`. You must load the values using `np.loadtxt(initial_Q_values.txt)`.
- The initial state for the agent before the training will be always 0.

For the modified ϵ-greedy, create an action selection method that receives the state as an argument and returns the action. Consider the following:

- The method must use sequentially one random number from the provided file each time, i.e., a random number is used only once.
- In case of a random number $rnd <= \epsilon$ the method returns an exploratory action. We will use the next random number to decide what action to return, as shown in Table 1.
- You should keep a counter for the random numbers, as you will need it to access the numbers sequentially, i.e., you should increase the counter every time after using a random number.

Random number (r)	Action	Action code
$r <= 0.25$	down	0
$0.25 < r <= 0.5$	up	1
$0.5 < r <= 0.75$	right	2
$0.75 < r <= 1$	left	3

Table 1: Exploratory action selection given the random number.

1.2 Implementing Q-learning and softmax

You should implement the temporal-difference method Q-learning. Consider the following for the implementation:

- For Q-learning the same set of initial Q-values will be used (provided in the file `initial_Q_values.txt`).
- Update the Q-values according to the method. Remember this is an off-policy method.
- As in the previous case, the initial state before training is also 0.

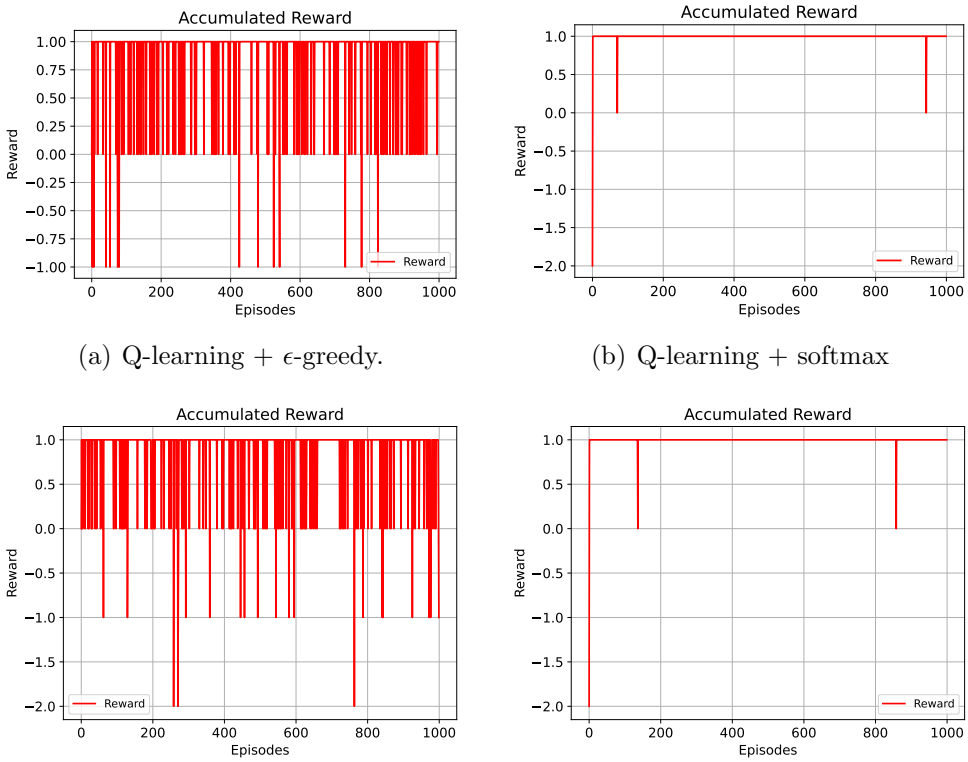
For the softmax action selection method, consider the following:

- Use a temperature parameter $\tau = 0.1$.
- Use a random number from the provided file to compare it with the cumulative probabilities to select an action. **Hint:** `np.searchsorted` returns the position where a number should be inserted in a sorted array to keep it sorted, this is equivalent to the action selected by softmax.
- Remember to use and increase a counter every time you use a random number.

1.3 Testing and plotting the results

You should plot a heatmap with the final Q-values after 1,000 learning episodes. Additionally, you should plot the accumulated reward per episode and the number of steps taken by the agent in each episode.

For instance, if you want to test your code, you can use the gridworld shown in Fig. 1 and you will obtain the rewards shown in Fig. 2 and the



(a) Q-learning + ϵ-greedy.

(b) Q-learning + softmax

(c) SARSA + ϵ-greedy.

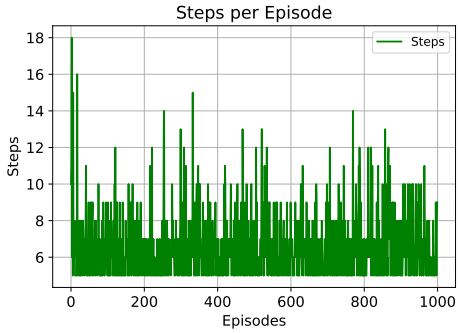
(d) SARSA + softmax.

Figure 2: Accumulated rewards.

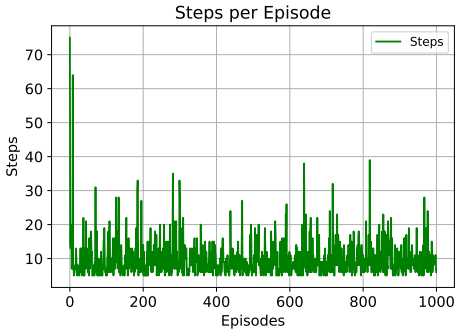
steps shown in Fig. 3. The learning parameters used are: learning rate $\alpha = 0.7$, discount factor $\gamma = 0.4$, $\epsilon = 0.25$, and $\tau = 0.1$.

In case you want to compare your results with the exact output for this example using `diff`, four files with the accumulated reward and four files with the steps per episode are provided (the combination of using Q-learning/SARSA and ϵ-greedy/softmax).

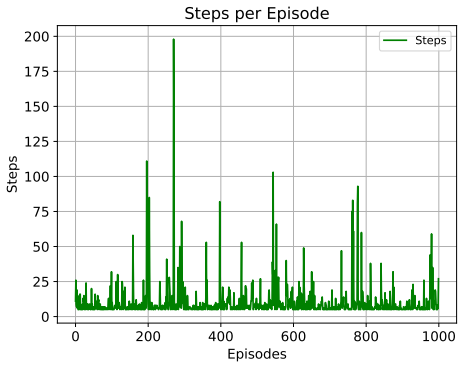
To mark your submission, different gridworlds and learning parameters will be used as test cases.



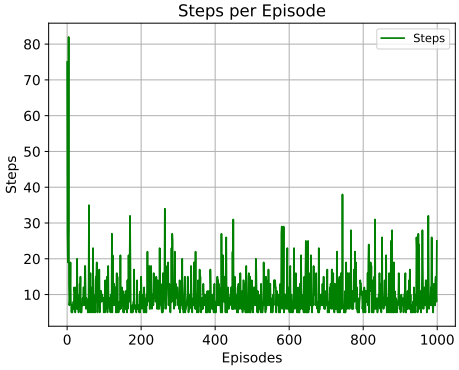
(a) Q-learning + ϵ-greedy.



(b) Q-learning + softmax.



(c) SARSA + ϵ-greedy..



(d) SARSA + softmax.

Figure 3: Steps per episode.

2 Submitting your assignment

You can do the assignment either individually or working in a couple with a classmate. If you decide to work in a couple with a classmate in another tutorial section, you need the approval of one of these two tutors, who will conduct the discussion with you and your classmate. However, the other tutor still needs to be informed by the student.

Your submission should be done by Moodle and consist of only one .py file. If you work in a couple, only one person is required to submit the file. However, the file should indicate on top as a comment the **full name and zID** of the students. It is your responsibility to indicate the names, we will not add people to a work after the deadline if you forget to include the names.

You can submit as many times as you like before the deadline – later submissions overwrite earlier ones. After submitting your file a good practice is to take a screenshot of it for future reference.

Late submission penalty: UNSW has a standard late submission penalty of 5% per day from your mark, capped at five days from the assessment deadline, after that students cannot submit the assignment.

3 Deadline and questions

Deadline: Week 5, Friday 30th of June 2023, 11:55pm. Please use the forum on Moodle to ask questions related to the project. However, you should not share your code to avoid making it public and possible plagiarism.

Although we try to answer questions as quickly as possible, we might take up to 1 or 2 business days to reply, therefore, last-moment questions might not be answered timely.

4 Plagiarism policy

Your program must be entirely your own work. Plagiarism detection software might be used to compare submissions pairwise (including submissions for any similar projects from previous years) and serious penalties will be applied, particularly in the case of repeat offences.

Do not copy from others. Do not allow anyone to see your code. Please refer to the UNSW Policy on Academic Honesty and Plagiarism if you require further clarification on this matter.