COMP3411/9814 24T3
Artificial Intelligence

Assignment 2 - Reinforcement Learning with Teacher Interaction

Due: Week 9, Friday, 8 November 2024, 5 pm AEDT

1 Problem Overview

In this assignment, you will implement and compare two reinforcement learning algorithms: **Q-learning** and **SARSA**, within a static grid world environment. The grid world consists of a 10×10 grid in which the agent must navigate from a random starting position to a designated goal while avoiding fixed obstacles.

You will first develop the Q-learning and SARSA algorithms, implementing action selection policies that allow the agent to choose actions using either an **ϵ-greedy** or **softmax** approach to balance exploration and exploitation.

After training the agents, you will simulate **interactive reinforcement learning (IntRL)** [1] by introducing a teacher-student framework. In this setup, a pre-trained agent (the teacher) provides advice to a new agent (the student) during its training.

The teacher's advice will be configurable in terms of its **availability** (probability of offering advice) and **accuracy** (probability that the advice is correct). You will evaluate the impact of teacher feedback on the student's learning performance by running experiments with varying levels of availability and accuracy. This involves comparing how the student agent learns with and without teacher guidance.

The goal is to understand how teacher interaction influences the learning efficiency of the new agent, and to observe how the Q-learning and SARSA algorithms differ in their adaptation to feedback-based learning.

2 Environment

For detailed information about the environment including setup instructions, key functions, agent movement and actions, movement constraints, and the reward structure please refer to the **Environment User Guide** provided separately as a PDF file along with the `env.py` file. Ensure you familiarise yourself with the environment before proceeding with the assignment.

Important: Make sure that `env.py`, `utily.py`, and the images folder are placed in the same directory as your solution. These files are required to import the environment, use the utility function, and render the environment visuals properly.

3 Parameter Settings

In this assignment, you have the flexibility to choose the learning rate (α), discount factor (γ), action selection method (e.g., ϵ-greedy, softmax) and exploration rate (e.g., fixed ϵ, initial ϵ and ϵ-decay rate, or temperature in softmax), and the number of episodes for training. However, the maximum number of steps per episode must be limited to **100**.

To ensure fair comparisons, the parameters used in **Task 1** must be reused in **Task 3**, and the parameters chosen for **Task 2** must also be applied in **Task 4**.

4 Task 1: Implement Q-learning

In this task, you are required to implement the Q-learning algorithm and train an agent in the provided environment. You should experiment with different hyperparameters such as the learning rate (α), discount factor (γ), and exploration rate to optimise the agent's learning performance.

During training, you must track the following metrics:

- **Total Rewards per Episode**: The cumulative reward accumulated by the agent during each episode.
- **Steps per Episode**: The number of steps the agent takes to complete each episode.
- **Number of Successful Episodes**: The number of times the agent successfully reaches the goal during the training process.

Tracking these metrics allows you to calculate the cumulative reward over episodes and generate valuable insights. After completing the training, you are required to:

- Generate a plot that displays the cumulative reward over the episodes.
- Report the **Success Rate**, **Average Reward per Episode**, and **Average Learning Speed**, using the following formulas:
  - **Success Rate**:

$$\text{Success Rate} = \left(\frac{\text{Number of Successful Episodes}}{N} \right) \times 100\% \tag{1}$$

    where N is the total number of episodes.

- **Average Reward per Episode**:

$$\text{Average Reward} = \frac{\sum_{i=1}^{N} R_i}{N} \tag{2}$$

where R_i is the total reward in the i-th episode.

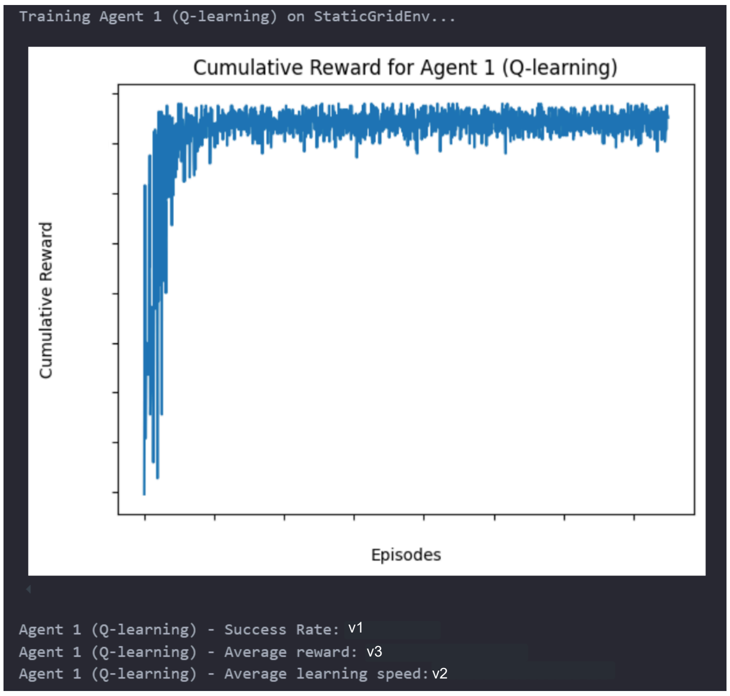
- **Average Learning Speed**:

$$\text{ALS} = \frac{1}{\frac{1}{N}\sum_{i=1}^{N} S_i} \tag{3}$$

where S_i is the number of steps taken in the i-th episode.

- Save these calculated values locally, as you will need to submit them along with your solution and upload them during the discussion.

An example plot is shown below:



```
Training Agent 1 (Q-learning) on StaticGridEnv...
```

```
Agent 1 (Q-learning) - Success Rate: v1
Agent 1 (Q-learning) - Average reward: v3
Agent 1 (Q-learning) - Average learning speed: v2
```

Figure 1: Example of cumulative reward plot and a list of key metrics for Q-learning agent.

5 Task 2: Implement SARSA

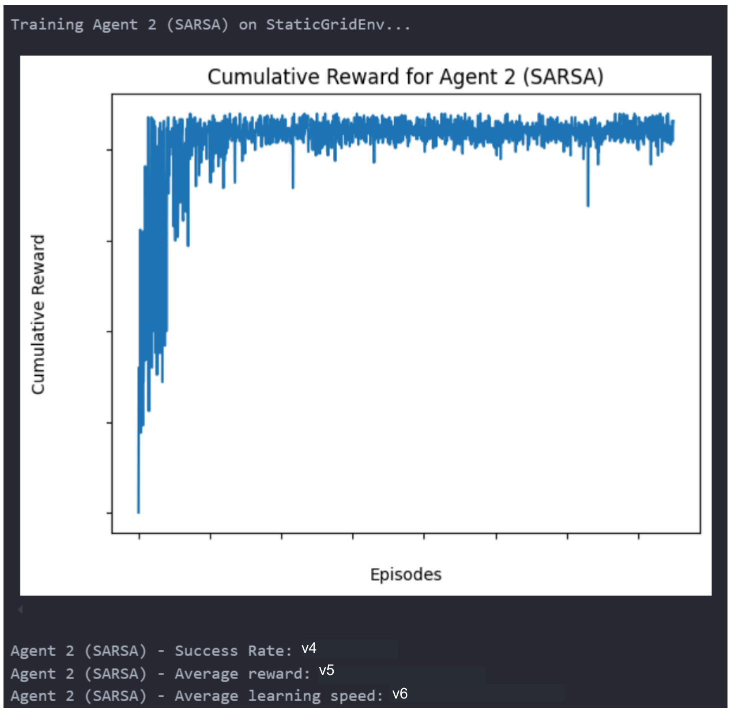
Task 2 follows the same structure as Task 1, but you will implement the SARSA algorithm instead of Q-learning. You are required to track the same metrics during training, including total rewards per episode, steps per episode, and the number of successful episodes.

After training, calculate and report the key performance metrics; Success Rate, Average Reward per Episode, and Average Learning Speed using the same formulas as in Task 1 (Equations (1), (2), and (3), respectively).

After completing the training, you are required to:

- Generate a plot that displays the cumulative reward over the episodes.
- Report the **Success Rate**, **Average Reward per Episode**, and **Average Learning Speed**.
- Save these calculated values locally, as you will need to submit them along with your solution and upload them during the discussion.

An example plot is shown below:


```
Training Agent 2 (SARSA) on StaticGridEnv...
```

```
Agent 2 (SARSA) - Success Rate: v4
Agent 2 (SARSA) - Average reward: v5
Agent 2 (SARSA) - Average learning speed: v6
```

Figure 2: Example plot of cumulative reward and a list of key metrics for SARSA agent.

6 Teacher Feedback Mechanism

A **teacher feedback system** is a valuable addition to the training process of agents using Q-learning or SARSA. In this system, a pre-trained agent (the teacher) assists a new agent by offering advice during training. The advice provided by the teacher is based on two key probabilities:

- The **availability** factor determines whether advice is offered by the teacher at any step.
- The **accuracy** factor dictates whether the advice given is correct or incorrect.

6.1 Pseudo-code for Teacher Feedback

This pseudo-code illustrates how the teacher's advice can be implemented:

```
function provide_teacher_advice(teacher_q_table, current_state, availability, accuracy):
    if random_value_1 < availability:  // Random chance based on availability
        if random_value_2 < accuracy:  // Random chance based on accuracy
            correct_action = action with the highest value in teacher_q_table at current_state
            return correct_action  // Return correct advice
        else:
            correct_action = action with the highest value in teacher_q_table at current_state
            possible_actions = all actions excluding the correct_action
            return a randomly chosen action from possible_actions  // Return incorrect advice
    return None  // No advice given
```

Figure 3: Pseudo-code for teacher feedback mechanism.

Two separate random value variables (each a number between 0 and 1) are generated in this process. First, `random_value_1` is compared to `availability` to determine whether the teacher provides advice. If advice is given, a second `random_value_2` is used to check if the advice is correct by comparing it to `accuracy`. These two checks ensure that advice is provided probabilistically and may not always be accurate.

The agent responds to the teacher's advice as follows:

- If the advice is **correct**, the agent takes the action with the highest Q-value, as advised by the teacher.
- If the advice is **incorrect**, the agent takes a random action, excluding the correct action.
- If **no advice** is given, the agent continues its independent learning using its exploration strategy (e.g., ϵ-greedy or softmax).

7 Task 3: Teacher Advice Using Q-learning Agent

In this task, you will use the agent trained in **Task 1** using Q-learning as the **teacher**. This teacher will provide advice to a new agent, which will also be trained using **Q-learning**. The new agent will be trained in the same environment and with the same parameters as in Task 1.

Your task is to write a function to train the new agent by incorporating a range of availability and accuracy values. You can achieve this, by using two nested loops that account for all pairwise combinations of the following values:

- **Availability**: [0.2, 0.4, 0.6, 0.8, 1.0]
- **Accuracy**: [0.2, 0.4, 0.6, 0.8, 1.0]

During the training process, you are required to track the following metrics, as you did in Task 1:

- **Total Rewards per Episode**: The cumulative reward accumulated by the agent during each episode.
- **Steps per Episode**: The number of steps the agent takes to complete each episode.

- **Number of Successful Episodes**: The number of times the agent successfully reaches the goal during the training process.

For each combination of availability and accuracy, you will calculate the following performance metrics:

- **Success Rate** (Equation (1))
- **Average Reward per Episode** (Equation (2))
- **Average Learning Speed** (Equation (3))

You must store these metrics for every combination in a DataFrame and use the average reward to plot a heatmap. The heatmap should represent all combinations of availability and accuracy, where:

- The **x-axis** represents the availability values.
- The **y-axis** represents the accuracy values.

Below is an illustration of the desired heatmap:

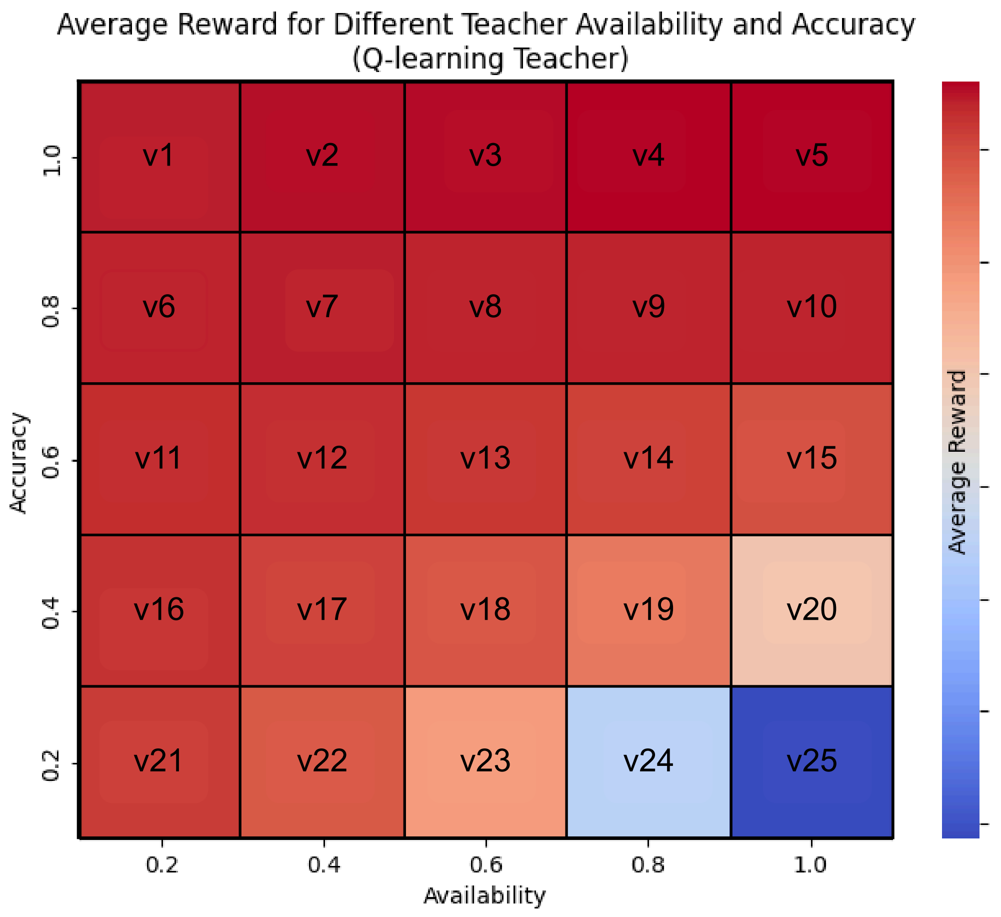


Figure 4: Illustration of heatmap showing the average reward scored for each combination of availability and accuracy for Q-learning.

8 Task 4: Teacher Advice Using SARSA Agent

Task 4 is similar to Task 3 in terms of tracking metrics, calculating performance, and storing the results in a DataFrame. However, in this task, the **teacher** is the agent trained in **Task 2** using SARSA. This teacher will provide advice to a new agent, which will also be trained using **SARSA** in the same environment and with the same parameters as in Task 2.

Your task is to write a function to train the new agent by incorporating a range of availability and accuracy values. You can achieve this, by using two nested loops that account for all pariwise combinations of the following value:

– **Availability**: [0.2, 0.4, 0.6, 0.8, 1.0]

– **Accuracy**: [0.2, 0.4, 0.6, 0.8, 1.0]

During the training process, you must track the following metrics (as in Task 3):

– **Total Rewards per Episode**: The cumulative reward accumulated by the agent during each episode.

– **Steps per Episode**: The number of steps the agent takes to complete each episode.

– **Number of Successful Episodes**: The number of times the agent successfully reaches the goal during the training process.

For each combination of availability and accuracy, you will calculate the following performance metrics:

– **Success Rate** (Equation (1))

– **Average Reward per Episode** (Equation (2))

– **Average Learning Speed** (Equation (3))

As in Task 3, you will store these metrics in a DataFrame and generate a heatmap where:

– The **x-axis** represents the availability values.

– The **y-axis** represents the accuracy values.

This heatmap will show the performance of the new agent trained with advice from the SARSA-trained teacher in terms of the average reward.

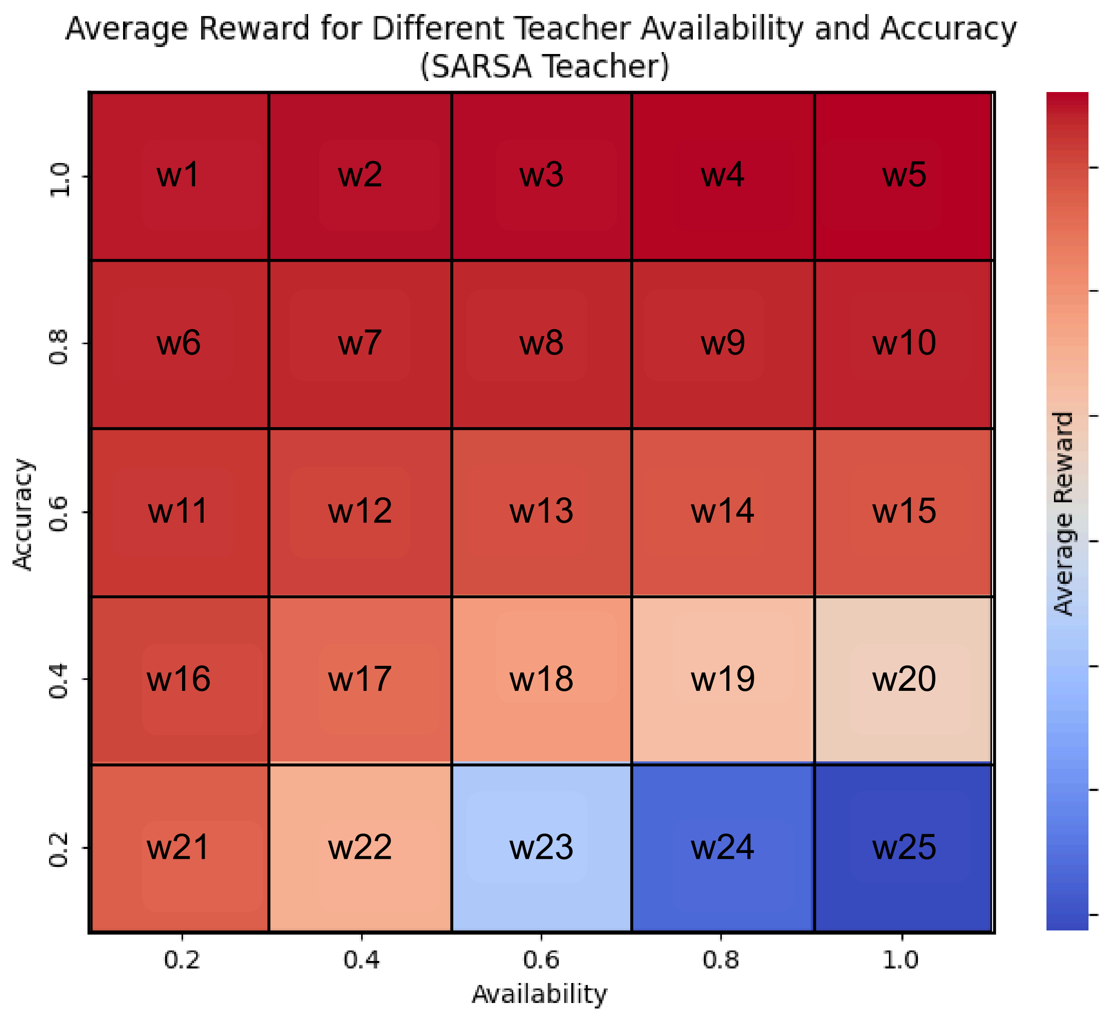


Figure 5: Illustration of heatmap showing the average reward scored for each combination of availability and accuracy for SARSA.

9 Testing and Discussing Your Code

In this section, you will load the results from **Tasks 1**, **2**, **3**, and **4** to compare the performance of agents trained with teacher advice against the baseline agents (trained without advice).

9.1 Loading Results

- For **Task 1 (Q-learning)** and **Task 2 (SARSA)**, load the results as tuples in the following format:

  $$(\text{average\_reward},\ \text{success\_rate},\ \text{learning\_speed})$$

  These represent the baseline results for Q-learning and SARSA.

- For **Task 3** and **Task 4**, load the saved DataFrames, which should have the following columns:

```
columns = [
    "Availability",          # Teacher availability level
    "Accuracy",              # Teacher accuracy level
    "Avg Reward",            # Average reward of the agent
    "Success Rate (%)",      # Success rate of the agent (in
       percentage)
    "Avg Learning Speed",    # Average speed of learning
]
```

You will use the loaded data to compare the baseline agents (from Tasks 1 and 2) and the agents trained with teacher advice (from Tasks 3 and 4). To do this, use the provided function `plot_comparison_with_baseline` from the `utils.py` module. The user guide for this function is provided separately as a PDF file along with the `utils.py` file.

In this comparison, you should observe how different combinations of availability and accuracy impact the agent's performance when trained with teacher advice. Discuss the results and the role of teacher feedback in improving or affecting the learning process.

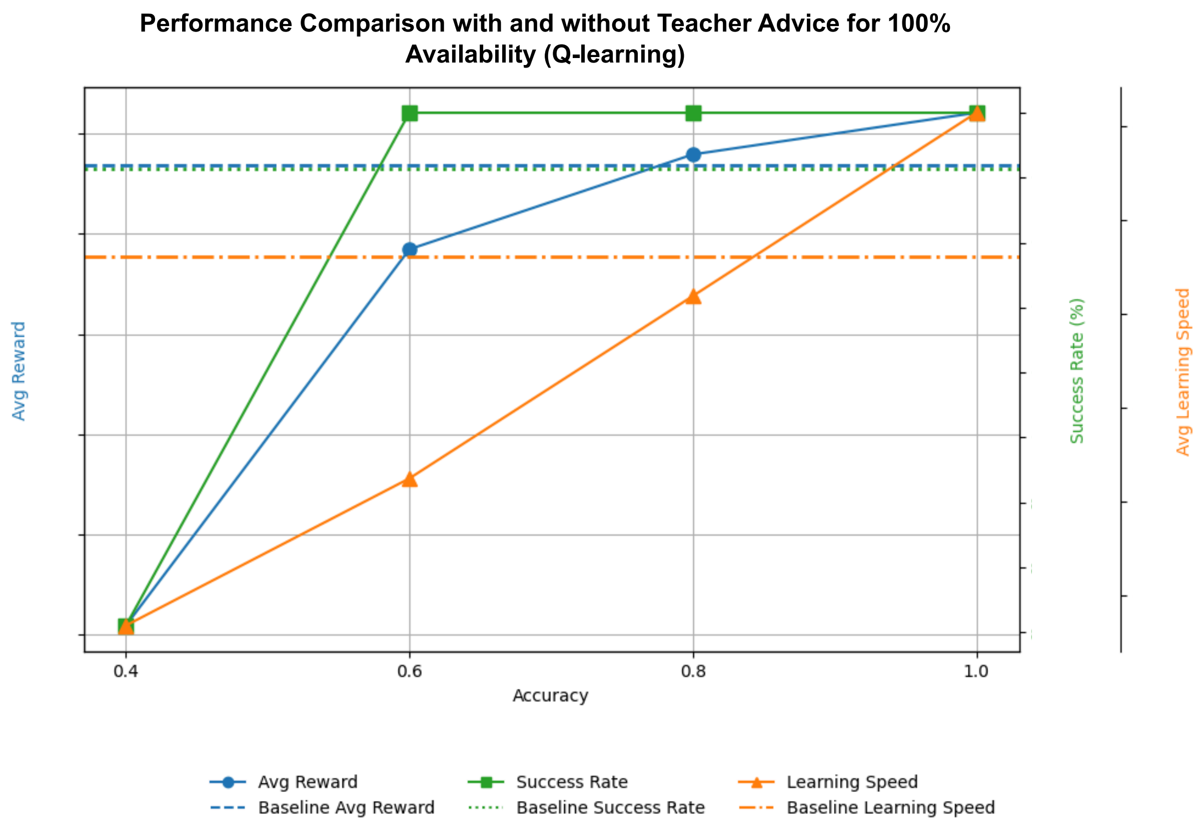


Figure 6: Comparison of the agent's average reward, success rate, and learning speed for Q-learning with teacher advice (solid lines) and without teacher advice (baseline, dashed lines). The teacher's advice is given with 100% availability, and the agent's performance is evaluated at different accuracy levels.

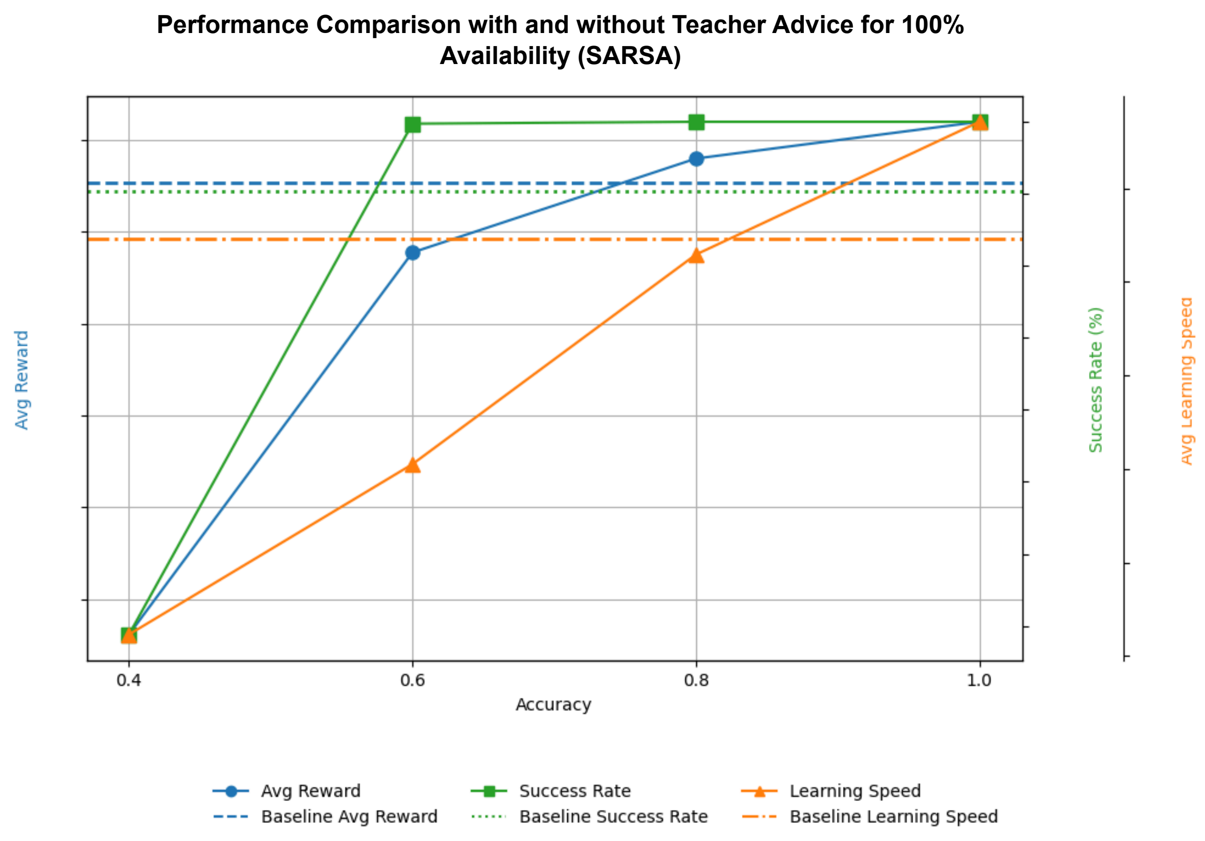


Figure 7: Comparison of the agent's average reward, success rate, and learning speed for SARSA with teacher advice (solid lines) and without teacher advice (baseline, dashed lines). The teacher's advice is given with 100% availability, and the agent's performance is evaluated at different accuracy levels.

You will receive marks for each section as shown in Table 1. You will receive 1 mark for code readability, and your tutor will also give you a maximum of 8 marks for each task depending on your level of code understanding, as follows: **8. Outstanding, 6. Great, 4. Fair, 2. Low, 0. Deficient or No answer**.

Table 1: Marking scheme for the assignment.

Criteria	**Marks**
Task 1: Implement Q-learning	
Plot of cumulative rewards versus episodes: A clear and well-labelled plot showing the cumulative rewards obtained by the agent over episodes.	1 mark
Calculation and reporting of Success Rate, Average Reward per Episode, and Average Learning Speed.	1 mark
Task 2: Implement SARSA	
Plot of cumulative rewards versus episodes: A clear and well-labelled plot showing the cumulative rewards for the SARSA agent over episodes.	1 mark
Calculation and reporting of Success Rate, Average Reward per Episode, and Average Learning Speed.	1 mark
Task 3: Teacher Advice Using Q-learning	
Implementation of teacher feedback mechanism with Q-learning and display of the generated DataFrame for this task.	1 mark
Plot of heatmap for all combinations of availability and accuracy with respect to average reward.	1 mark
Demonstrate understanding of the code and analysis during discussion.	8 marks
Task 4: Teacher Advice Using SARSA	
Implementation of teacher feedback mechanism with SARSA and display of the generated DataFrame for this task.	1 mark
Plot of heatmap for all combinations of availability and accuracy with respect to average reward.	1 mark
Demonstrate understanding of the code and analysis during discussion.	8 marks
Code Readability and Understanding	
Overall code readability, including well-commented and tidy script.	1 mark
Total marks	25 marks

10 Resources Provided

There are 4 files attached to this assignment document:

- `env.py`: The environment code.
- `Environment User Guide.pdf`: A detailed user guide for the environment.
- `utils.py`: A utility module containing the `plot_comparison_with_baseline` function.
- `Utils User Guide.pdf`: A user guide for the `utils.py` module.

Ensure you read the user guides and understand the provided code before proceeding with the assignment.

11 Submission Guidelines

You should submit your solution via Moodle by uploading the following:

- A Jupyter Notebook file (`.ipynb`) containing your code for all tasks.
- The saved results from **Tasks 1** and **2**, including the calculated values for Success Rate, Average Reward per Episode, and Average Learning Speed as well as the dataframes for **Task 3** and **4**. These can be in any format (e.g., text file, JSON, CSV).

Ensure that your code is well-commented and easy to read. All files should be submitted before the deadline. Missing the submission of the requested files will lead to a deduction of 2 marks from Demonstrate understanding of the code and analysis during discussion on tasks 3 and 4, respectively. 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.

12 Deadline and Questions

Deadline: Week 9, Friday, 8 November 2024, 5 pm AEDT.

Please use the forum on Moodle to ask questions related to the assignment. We will prioritise questions asked in the forum. However, you should not share your code to avoid making it public and possible plagiarism. If that's the case, use the course email `cs3411@cse.unsw.edu.au` as an alternative.

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.

For any questions regarding the discussion sessions, please contact your tutor directly. You can access your tutor's email address through Table 2.

Table 2: COMP3411/9814 24T3 Tutorials

Number	Class ID	Time	Tutor	Email
1	4106	Wed 16:00 - 18:00	Ramya Kumar	ramya.kumar1@student.unsw.edu.au
2	4107	Wed 18:00 - 20:00	Janhavi Jain	j.jain@student.unsw.edu.au
3	4101	Thu 11:00 - 13:00	Maryam Hashemi	m.hashemi@unsw.edu.au
4	4102	Thu 11:00 - 13:00	Maher Mesto	m.mesto@unsw.edu.au
5	4103	Thu 13:00 - 15:00	Maryam Hashemi	m.hashemi@unsw.edu.au
6	6138	Thu 17:00 - 19:00	Ramya Kumar	ramya.kumar1@student.unsw.edu.au
7	6139	Thu 19:00 - 21:00	Maher Mesto	m.mesto@unsw.edu.au
8	4097	Fri 11:00 - 13:00	Zhijin Meng	zhijin.meng@student.unsw.edu.au
9	6132	Fri 13:00 - 15:00	Zhijin Meng	zhijin.meng@student.unsw.edu.au
10	4099	Fri 15:00 - 17:00	Janhavi Jain	j.jain@student.unsw.edu.au
11	6134	Fri 17:00 - 19:00	Stefano Mezza	s.mezza@unsw.edu.au
12	13044	Thu 18:00 - 20:00	John Chen	xin.chen9@student.unsw.edu.au
13	12705	Wed 16:00 - 18:00	John Chen	xin.chen9@student.unsw.edu.au

13 Plagiarism Policy

Your program must be entirely your own work. Plagiarism detection software may be used to compare submissions pairwise (including submissions from previous years), and serious penalties will be applied in cases of plagiarism, particularly for repeat offences.

Do not copy from others. Do not allow anyone to see your code. We encourage you to tackle this assignment independently to maximise your learning experience. Engaging fully with the assignment tasks will enhance your understanding of reinforcement learning concepts and algorithms.

Please refer to the UNSW Policy on Academic Integrity for further clarification: `https://student.unsw.edu.au/plagiarism`.

References

[1] Bignold, Adam, Francisco Cruz, Richard Dazeley, Peter Vamplew, and Cameron Foale. ”An evaluation methodology for interactive reinforcement learning with simulated users.” Biomimetics 6, no. 1 (2021): 13.