Assignment 3

EE675: Introduction to Reinforcement Learning

March 26, 2025

Instructions

- **Only Part A will be graded. Part B is for your practice and will not be graded.**
- Kindly name your submission files as ‘`RollNo_Name_A3_PartA/B.ipynb`‘, based on the part you are submitting. Marks will be deducted for all submissions that do not follow the naming guidelines.
- You are required to work out your answers and submit only the iPython Notebook. The code should be well commented and easy to understand as there are marks for this.
- Submissions are to be made through HelloIITK portal. Submissions made through mail will not be graded.
- Answers to the theory questions, if any, should be included in the notebook itself. While using special symbols use the *LaTeX* mode
- Make sure your plots are clear and have title, legends and clear lines, etc.
- Plagiarism of any form will not be tolerated. If your solutions are found to match with other students or from other uncited sources, there will be heavy penalties and the incident will be reported to the disciplinary authorities.
- In case you have any doubts, feel free to reach out to TAs for help.

Part-A (Deadline - 5th April 2025)

(Cliff Walking) `[20 Marks]` Through this grid world exercise we will compare SARSA and Q-learning algorithms, highlighting the difference between them. Consider the grid world shown in the Figure below. This is a standard undiscounted, episodic task, with start and goal states, and the usual actions causing movement up, down, right, and left. Reward is -1 on all transitions except those into the the region marked “The Cliff.” Stepping into this region incurs a reward of -100 and sends the agent instantly back to the start. The episode ends when the agent reaches the goal state.

Given the template notebook skeleton code, Implement and answer the following questions in the notebook:

1. Implement and compare the SARSA and Q-Learning methods `[8+8 Marks]`
2. Why is Q-learning considered an off-policy control method? `[3 Marks]`

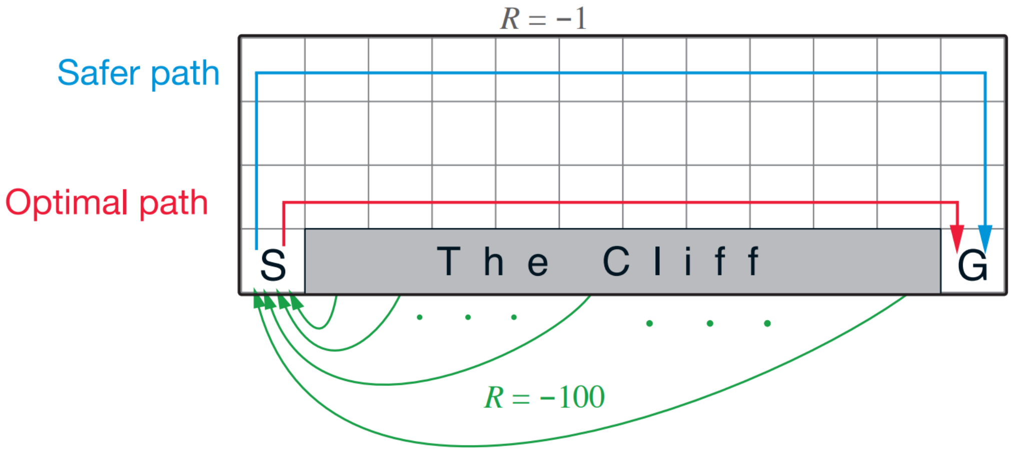
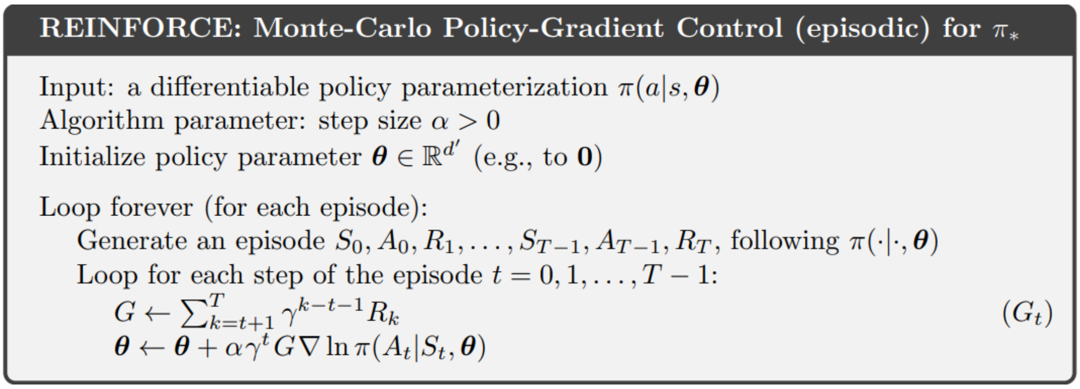


Figure 1: Cliff Walking Environment



```
REINFORCE: Monte-Carlo Policy-Gradient Control (episodic) for $\pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot,\boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                    $(G_t)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$
```

Figure 2: REINFORCE

3. Which algorithm takes a safer path? Why? `[1 Marks]`

Part-B (OPTIONAL - WILL NOT BE GRADED)

(Cart Pole Balancing using Policy Gradient: REINFORCE) `[WILL NOT BE GRADED]` Through this Cart Pole balancing exercise we will learn policy gradient algorithms. Consider the cart-pole problem where a pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The pendulum is placed upright on the cart and the goal is to balance the pole by applying forces in the left and right direction on the cart. For information about the observation_space, action_space and rewards of the environment refer the cart pole documentation.

For this part implement the policy gradient algorithm REINFORCE as shown below in the figure. You are required to use a **linear policy** (say parameterized by $\boldsymbol{\theta} = [\boldsymbol{\theta}_1 \ \boldsymbol{\theta}_2]^\top$, a 2x4 matrix for the cart pole problem) of state s passing through a softmax i.e.

$$\pi(a|s,\boldsymbol{\theta}) = \text{softmax}(\boldsymbol{\theta} \cdot s) = [\exp\{\boldsymbol{\theta}_1^\top s\} \ \exp\{\boldsymbol{\theta}_2^\top s\}]^\top / \exp\{\boldsymbol{\theta}_1^\top s\} + \exp\{\boldsymbol{\theta}_2^\top s\}$$

1. Write the expression for the gradient $\nabla \ln \pi(A_t|S_t,\boldsymbol{\theta})$ for the policy shown above (write in LaTeX in the jupyter-notebook submission) `[2 Marks]`

2. Implement REINFORCE algorithm with appropriate choice of algorithm parameters [5 Marks]

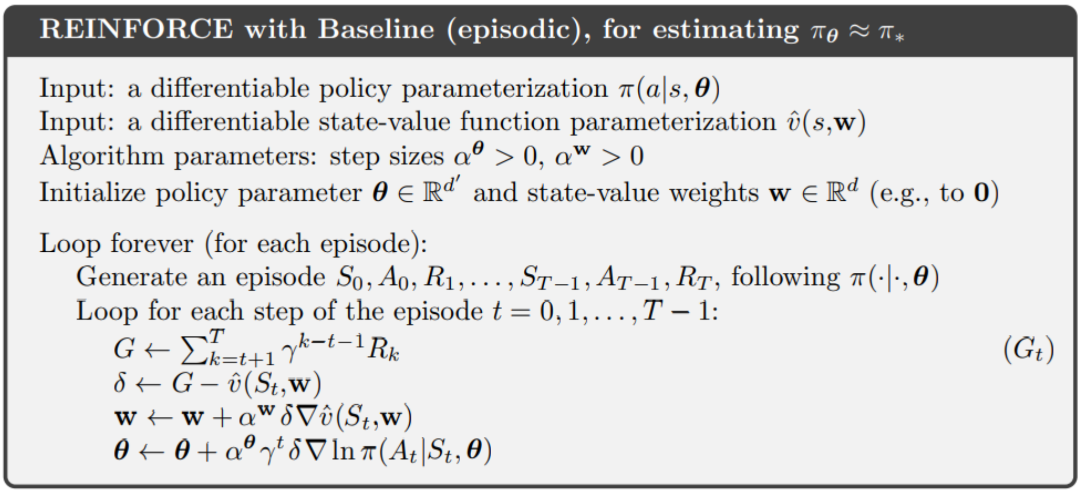
3. Plot the training rewards over 1000 episodes [2 Marks]

4. Test the trained policy and compute the average reward over 5 episodes [1 Marks]

5. Implement REINFORCE algorithm with baseline as shown in Figure 2 with appropriate choice of algorithm parameters. Take the state value function as a linear function of the state [7 Marks]

$$\hat{v}(s, w) = \mathbf{w}^\top s$$

6. Compare and plot the training performance for both the algorithms [3 Marks]



```
REINFORCE with Baseline (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)

Loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    Loop for each step of the episode $t = 0, 1, \ldots, T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$                                        $(G_t)$
        $\delta \leftarrow G - \hat{v}(S_t,\mathbf{w})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \delta \nabla \hat{v}(S_t,\mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
```

Figure 3: REINFORCE with baseline

Note: Take each episode to have a maximum of 500 steps