Cliff Walking

Name: Kunal Lad
UT Eid: KL28697
Email: kunal.lad@utexas.edu

Goal

Goal of this assignment is to study temporal difference learning algorithms through Cliff Walking problem described in Chapter 6 of textbook [1]). We implement the 2 basic TD learning algorithms: SARSA and QLearning and study how their performance varies with various parameters and environment settings.

Problem Description

Environment: Gridworld shown in the Figure 1. This is a standard undiscounted, episodic task, with start and goal states.

Actions: Usual actions in grid world {up, down, right, and left} causing movement in respective direction.

Rewards: Reward is 1 on all transitions except those into the region marked “The Cliff” Stepping into this region incurs a reward of 100 and sends the agent instantly back to the star

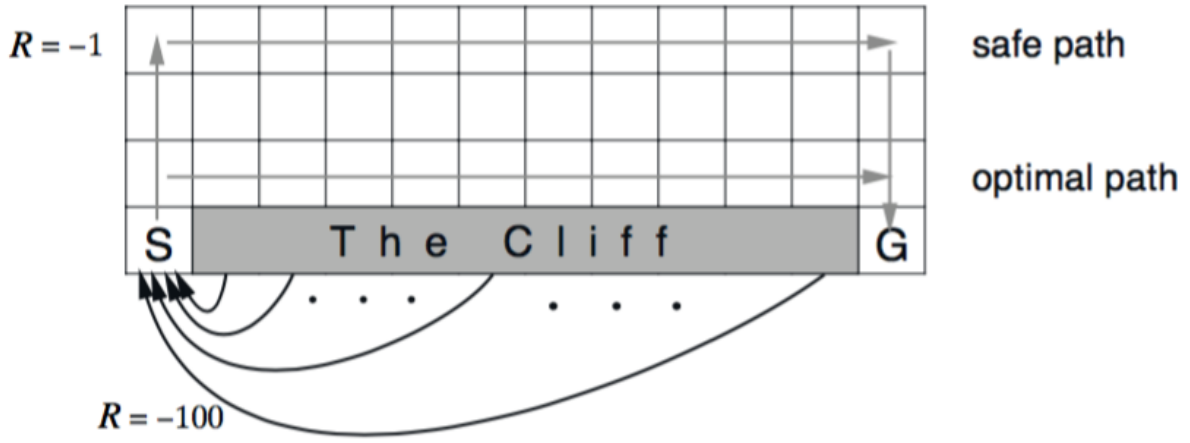

Fig 1 Cliff Walking Environment

Methodology

Approach

I used 2 temporal learning algorithms (SARSA and QLearning) described in Chapter 6 of the textbook [1] for solving this problem. These approaches differ only in the way they apply temporal difference update. Fig 2 & 3 show pseudo code for these algorithms.

Implementation

I implemented the above algorithms in ipython notebook. All the code and results for various experiments which were performed can be found at my github profile under the repository [cliff_walking]

Sarsa: An on-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Fig 2 Sarsa Pseudo Code

Q-learning: An off-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
    $S \leftarrow S'$
  until S is terminal

Fig 3 QLearning Pseudo Code (Credits [1])

Experiments & Results

In this section we discuss the various experiments conducted for studying Temporal Difference Learning approaches for episodic task of cliff walking. We use the following parameter values (unless mentioned otherwise):

- ϵ = 0.1
- α = 0.1
- γ = 1

E1: SARSA v/s QLearning

In this experiment we compared the policies learnt by Sarsa and QLearning and also investigate how their learning process differs from each other. Fig 4 shows the results. As discussed in [1] we observe that Sarsa learns the safe policy which prefers to walk away from cliff whereas QLearning learns the optimal policy which is to walk along the cliff. Policies differ from each other because of the way two algorithms perform the update. Since QLearning considers max action from next state while performing the updates, it knows that it is safe to walk along the cliff if correct action is taken, whereas Sarsa considers walking along the cliff bad since it performs the update based on action selected by ϵ greedy policy and there is high penalty for falling off the cliff.

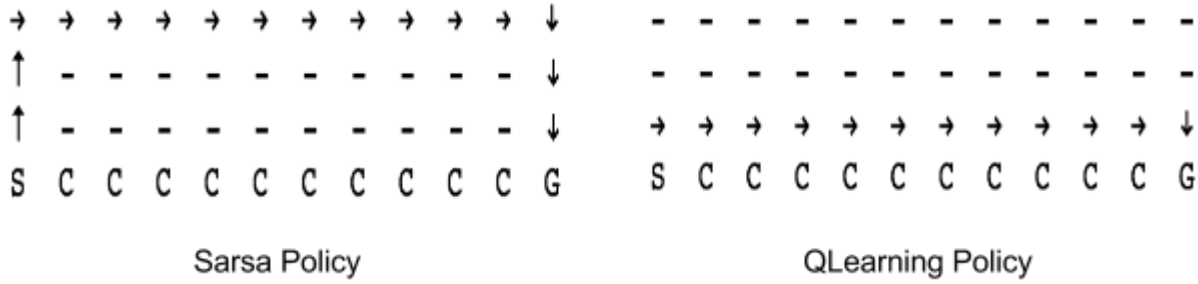

Fig 4 Learnt Optimal Policies

Fig 5 shows the online comparison of 2 approaches. We observe that even though QLearning learns optimal policy its online performance is worse than sarsa. This is due to the fact that even though updates happen according to max action, ϵ greedy policy is followed for taking next action.

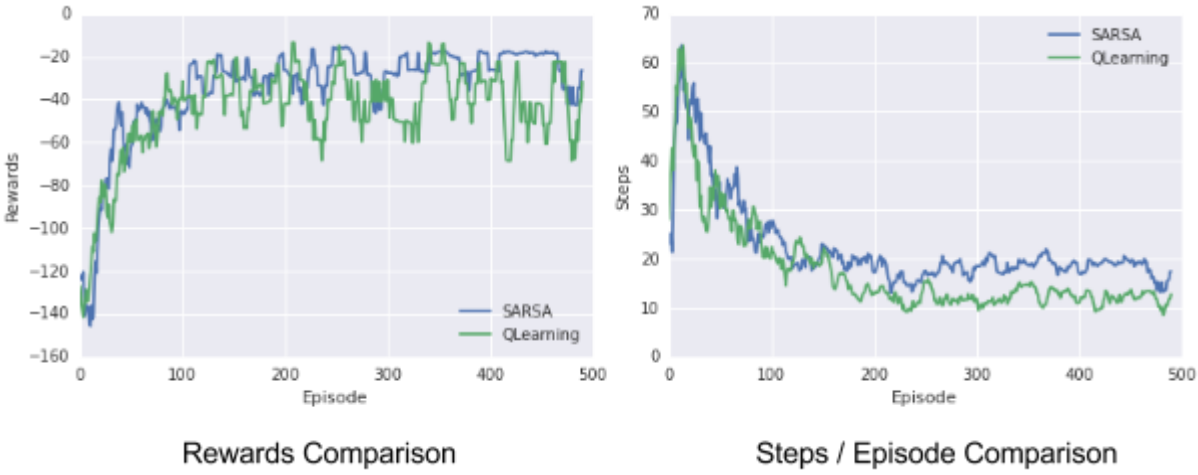

Fig 5 Online Performance

E2: Effect of Fall Penalty

In this experiment I investigated how these policies change if fall penalty is lower. Main motivation for this experiment was to see if Sarsa policy changes if fall penalty is low. Fig 6 shows the results of the experiment.

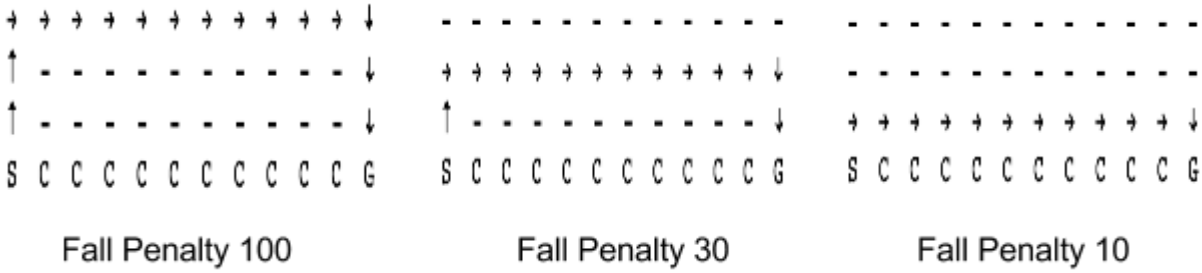

Fig 6 Sarsa Policy Variation with Fall Penalty

As expected we observe that policy learnt by Sarsa moves more and more towards the cliff as the fall penalty reduces. This also suggests that if we reduce ϵ gradually (leading to less exploration near cliff states) while learning then Sarsa will also learn the optimal policy and both algorithms will converge to correct optimal policy.

Fig 7 shows rewards and steps per episode comparison for fall penalties 30 and 10. We observe that online performance for Sarsa is much closer to QLearning in these plots as compared to the case with fall penalty 100 (Fig 5).

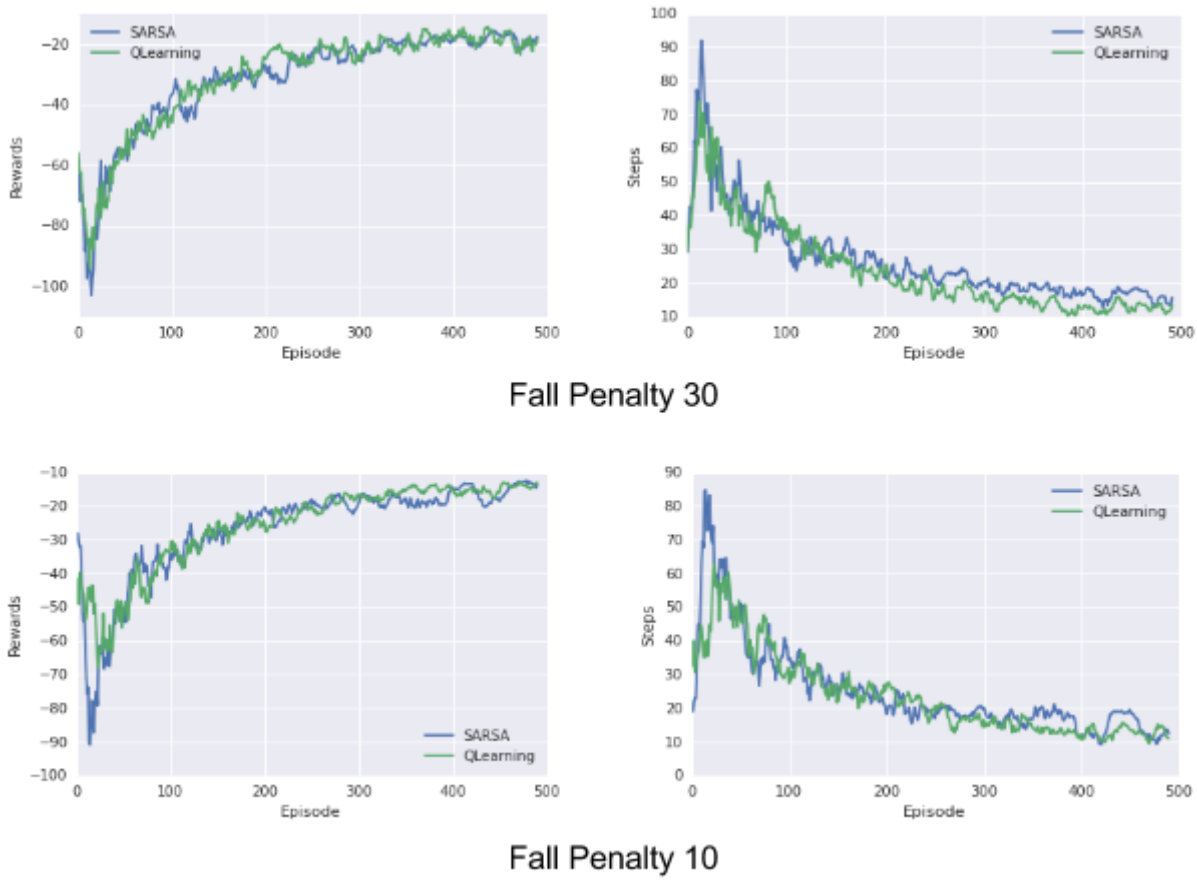

Fig 7 Online Performance for different fall penalties

E3: Multiple Goal States

In this experiment we investigate the effect of multiple goal states (G1 and G2). G1 has reward of 0 whereas G2 has reward of 10. Fig 8 shows the learnt policy by Sarsa and QLearning. In this case both learn optimal policies but Sarsa policy is less risky since it does not involve walking along the cliff.

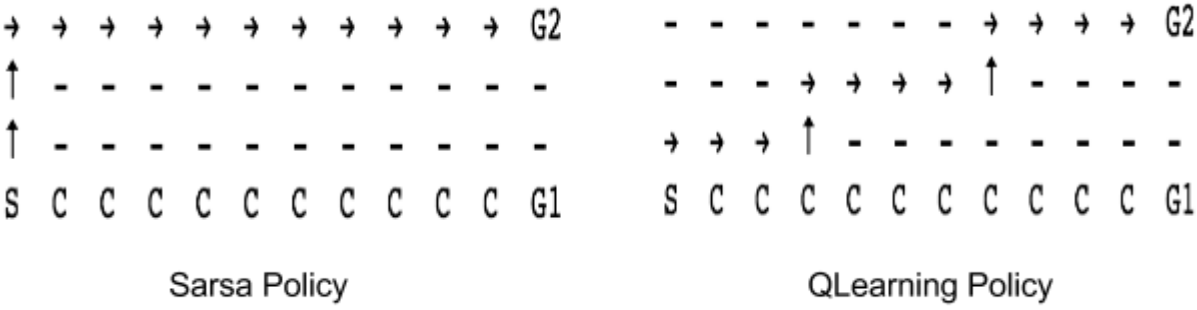

Fig 8 Learnt Optimal Policy

E4: More Actions

In this experiment, we investigate the effect of adding additional actions (movement along diagonal points). We observe that QLearning policy still moves along the cliff whereas Sarsa

still follows a safer route. This again demonstrates that Sarsa learns safe policies and QLearning's ability to identify optimal policies even in more complex environments.

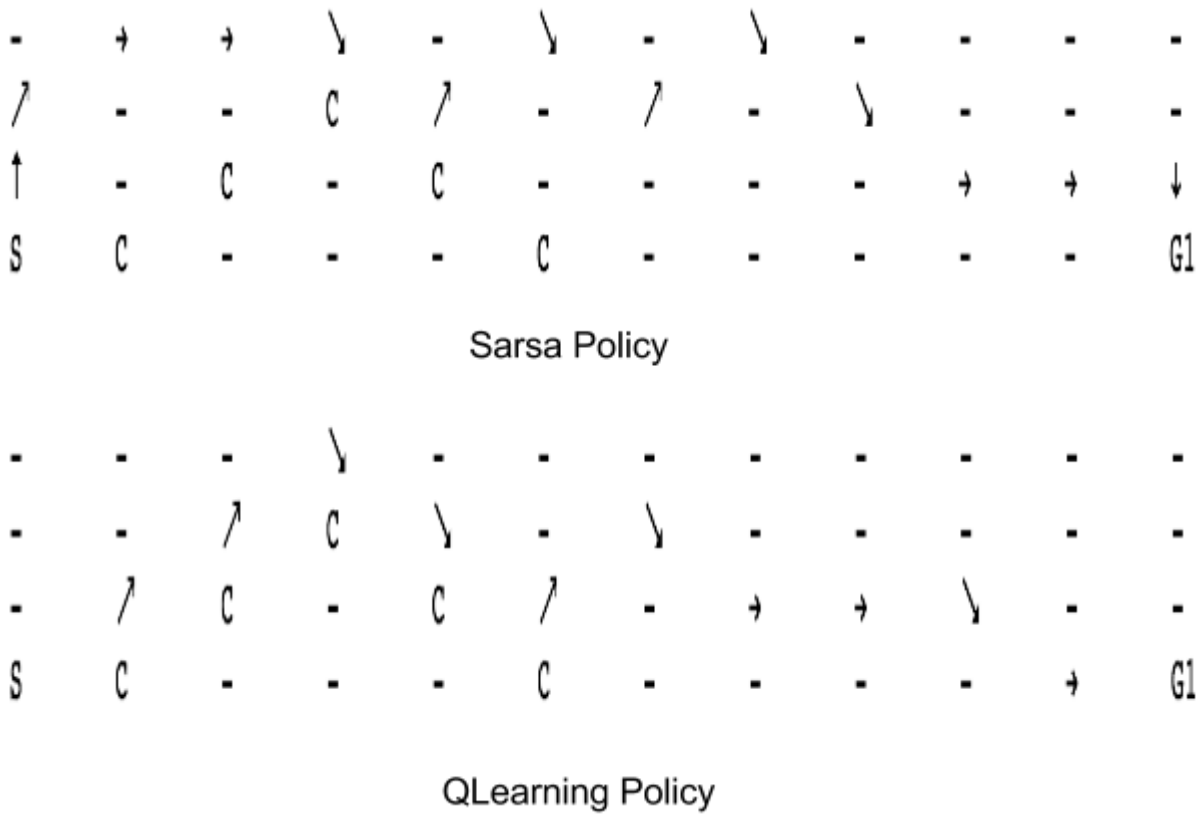

Fig 8 Optimal Policy with Diagonal Actions

Conclusion

- Sarsa learns safe policy whereas QLearning learns optimal policy.
- Sarsa and QLearning converge to same optima if exploration is reduced over time.
- Above mentioned behavior is not affected by more actions, multiple goals or different environment (cliff positions).

References

[1] Reinforcement Learning: An Introduction