Tabular Planning

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

Goal

Goal of this assignment is to study tabular planning algorithms through Cliff Walking problem described in Chapter 6 and Dyna Maze problem (Example 8.1) of textbook [1]). We implement the tabular Dyna Planning versions of TD learning algorithms: SARSA and QLearning implemented in previous homework and study how their performance compares to corresponding counterparts in 2 different environmental settings.

Problem Description

Environment1: 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

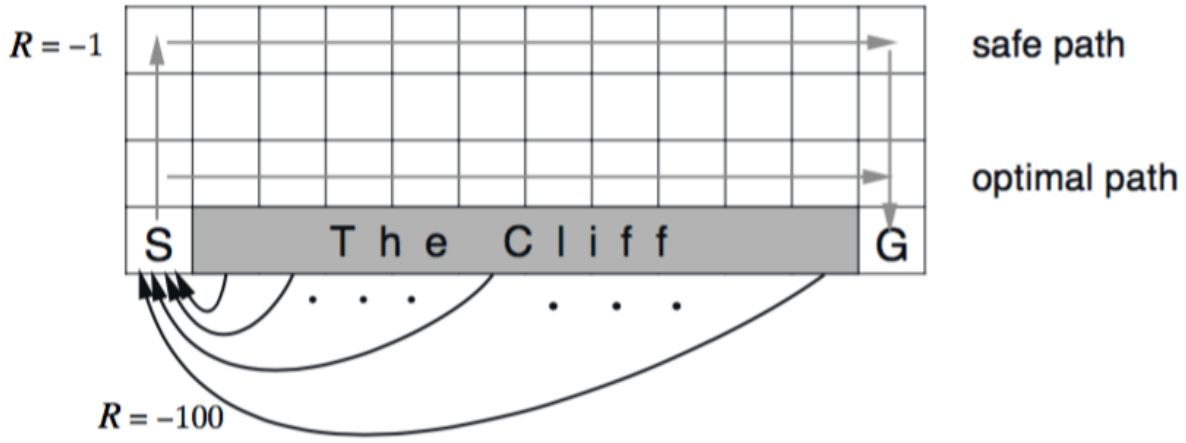


Fig 1 Cliff Walking Environment (Credits [1])

Environment2: Maze shown in the Figure 2. This is a discounted, 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 0 on all transitions and +1 on reaching the Goal.

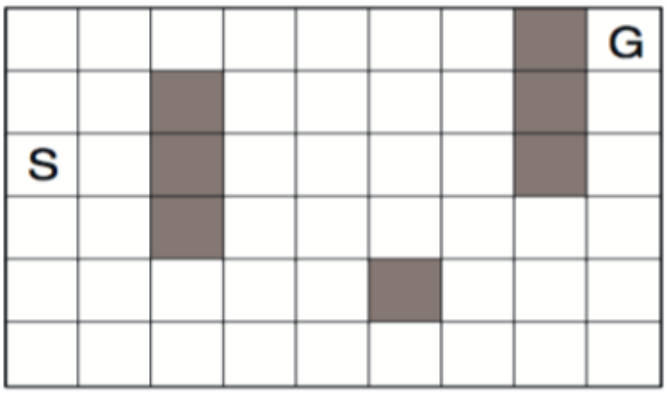


Fig 2 Maze (Credits [1])

Methodology

Approach

I used SARSA and QLearning algorithms (Chapter 6) and their tabular planning versions (Chapter 8) of [1] for solving this problem. Sarsa & QLearning differ only in the way they apply temporal difference update. Their corresponding planning versions use the Dyna architecture shown in Fig 3. Fig 4, 5 and 6 show pseudo code for these algorithms.

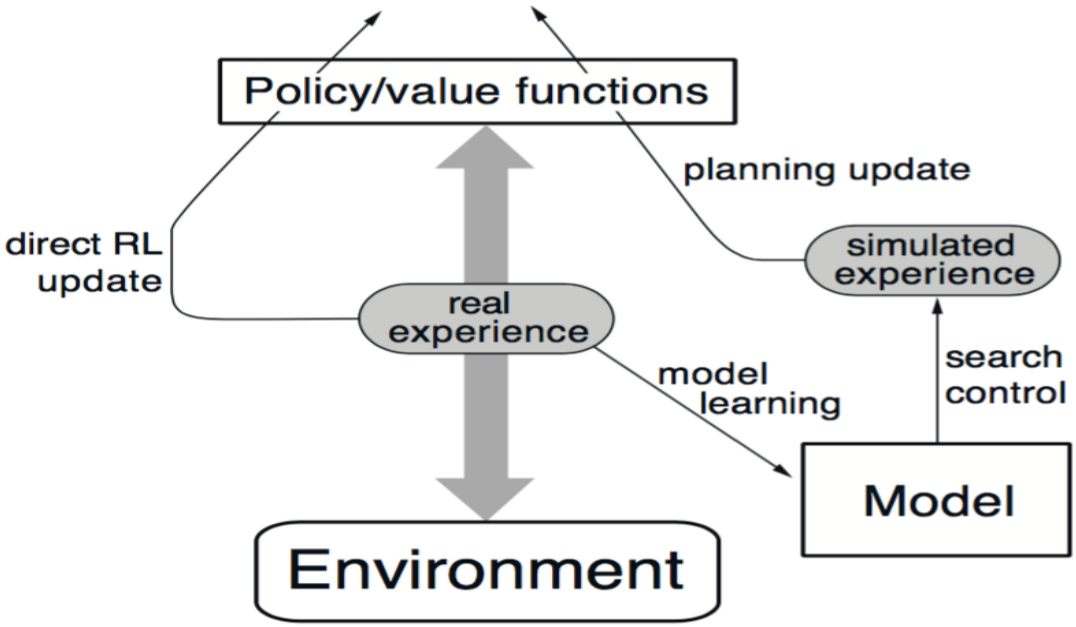


Fig 3 Dyna Architecture (Credits [1])

Implementation

I reused the code for Sarsa and QLearning from HW3 and implemented their tabular planning versions in ipython notebook. All the code and results for various experiments which were performed can be found at my github profile under the repository tabular planning.

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 4 Sarsa Pseudo Code (Credits [1])

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'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Fig5 QLearning Pseudo Code (Credits [1])

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
  (a) $S \leftarrow$ current (nonterminal) state
  (b) $A \leftarrow \epsilon$-greedy(S,Q)
  (c) Execute action A; observe resultant reward, R, and state, S'
  (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
  (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
  (f) Repeat n times:
    $S \leftarrow$ random previously observed state
    $A \leftarrow$ random action previously taken in S
    $R, S' \leftarrow Model(S,A)$
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Fig 6 Tabular QLearning Pseudo Code (Credits [1])

Experiments & Results

In this section we discuss the various experiments conducted for studying Dyna Planning versions of TD Learning approaches Sarsa and QLearning for undiscounted episodic task of cliff walking and discounted episodic task of Dyna Maze. We use the following parameter values (unless mentioned otherwise):

Parameters	Cliff Walking	Dyna Maze
ϵ	0.1	0.1
α	0.1	0.1
γ	1	0.95
Planning Steps	10	10

E1: Cliff Walking (TD vs Dyna)

In this experiment we compared the policies learnt by Sarsa and QLearning against their corresponding tabular planning versions. We also investigate how their learning process differs from each other. Fig 7 shows the learnt policies. We observe that Dyna Sarsa policy is mid way between Sarsa and QLearning policy, whereas DynaQ policy is same as QLearning. Thus the additional simulated experience obtained through planning helps the SARSA move from the safe policy towards the optimal policy. As mentioned in textbook [1], reducing ϵ slowly over time also has same effect but planning helps in faster convergence.

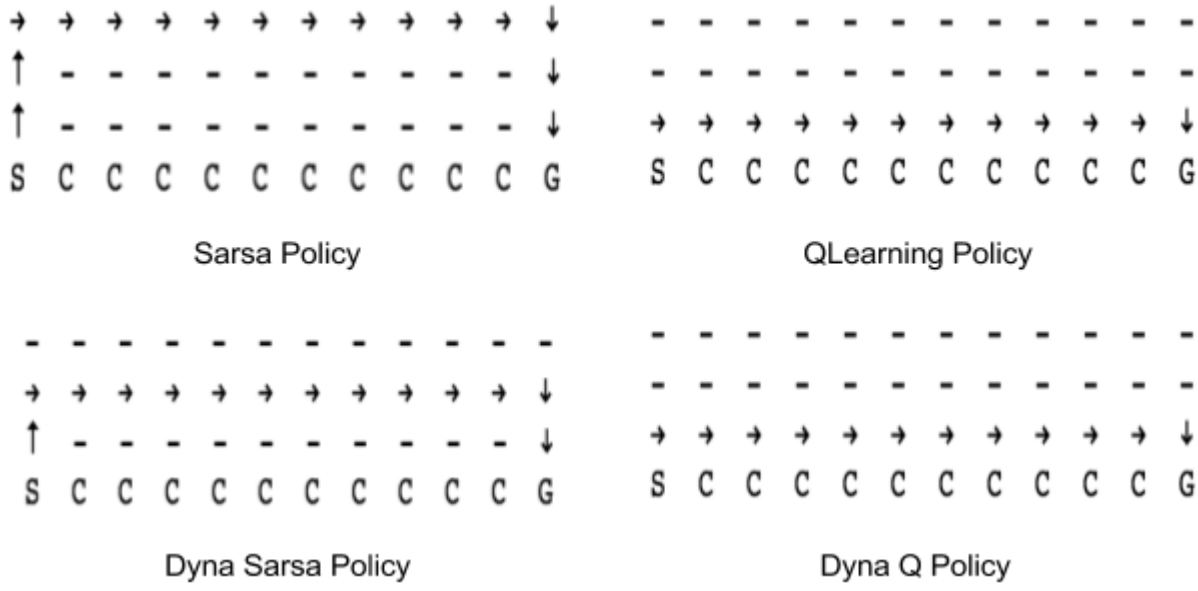


Fig 7 Learnt Policies

Fig 8 and 9 compare the online performance of TD algorithms against their planning versions. We observe that the Dyna versions (green curve) of both SARSA and QLearning dominate their TD versions (blue curves) at the start in Fig 8 and 9. This shows that they converge faster both in terms of rewards and number of steps per episode.

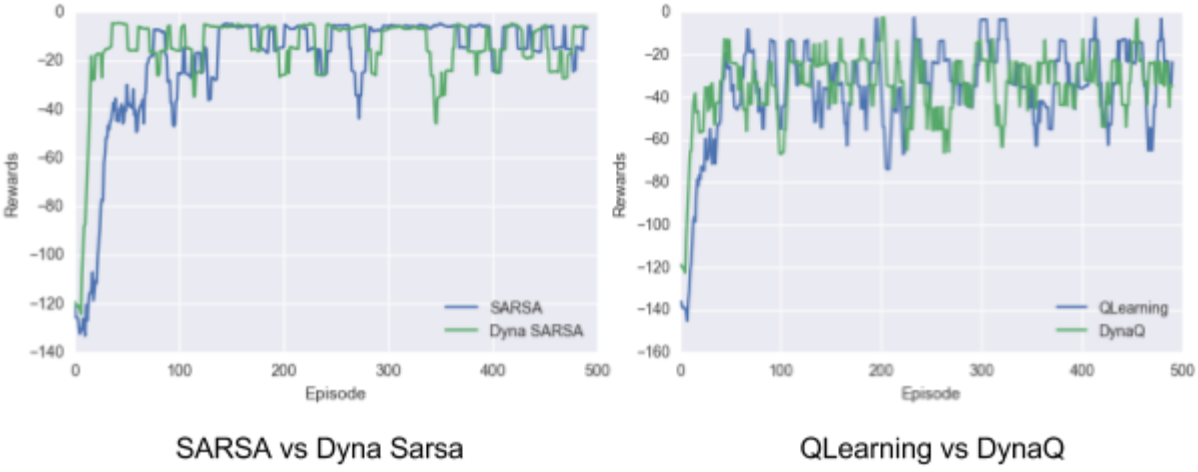
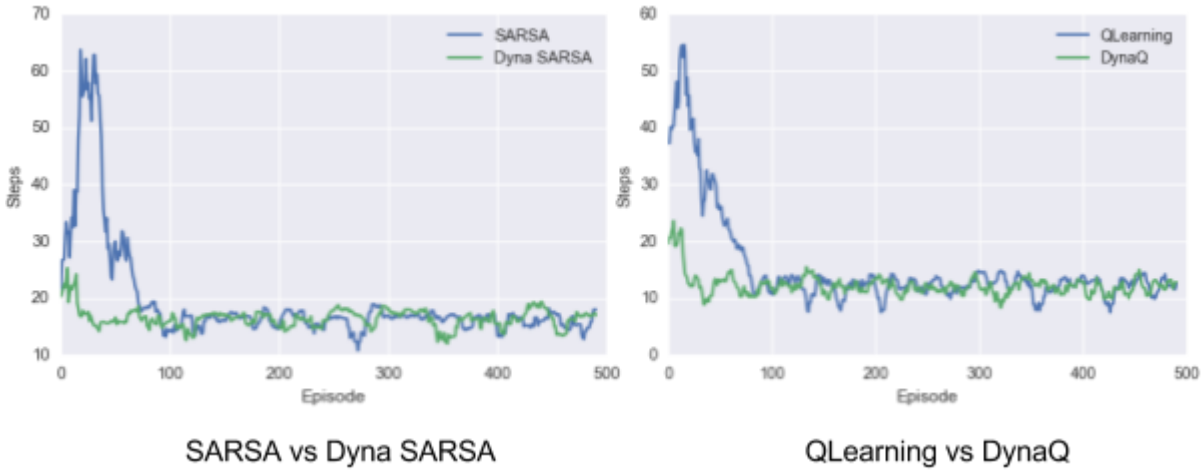
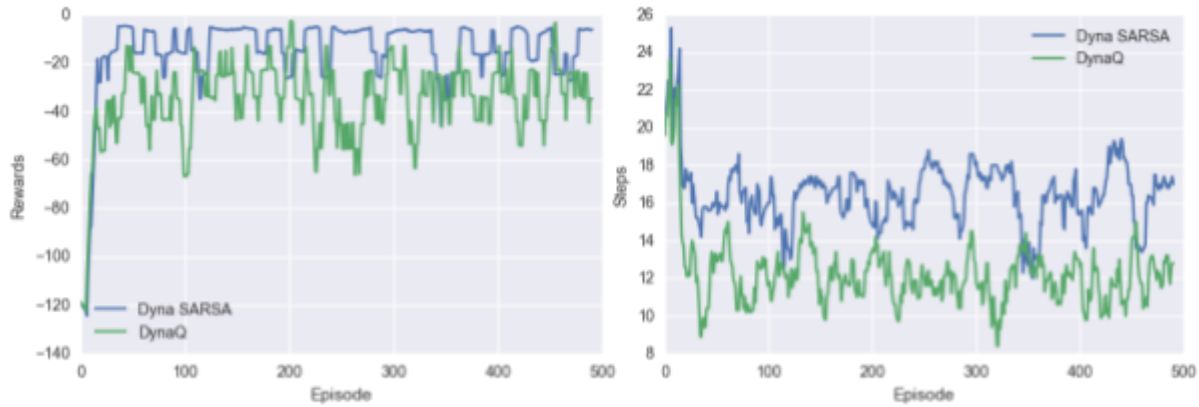


Fig 8 Rewards Comparison



SARSA vs Dyna SARSA QLearning vs DynaQ

Fig 9 Steps per Episode comparison



Rewards Comparison Steps Compariosn

Fig 10 Online performance of Dyna SARSA vs DynaQ

Finally we investigate how the two planning versions compare against each other. We observe similar trend to what we observed in HW3 for SARSA vs QLearning: DynaQ learns optimal policy but its online performance is worse than Dyna SARSA. Again the explanation is that for DynaQ updates happen according to max action but ϵ greedy policy is followed for taking next action. Hence it might sometimes fall off the cliff. If we reduce ϵ over time then performance for both will converge to same level.

E2: Dyna Maze

In this experiment we investigate the effect of planning for TD Learning algorithms: SARSA and QLearning on Dyna Maze environment from Example 8.1 of textbook [1]. Fig 11 shows the optimal policy learnt by all the algorithms. Unlike Cliff Walking all algorithms learn the same policy for Dyna Maze problem.

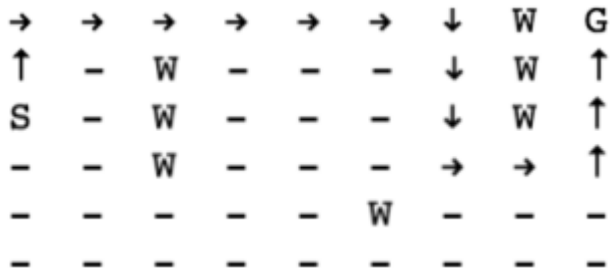


Fig 11 Optimal policy for Dyna Maze found by TD Learning and Tabular Planning algorithms

Fig 12 demonstrates the benefit of planning in the Dyna Maze environment. We observe a similar trend as Cliff Walking tasks where both TD Learning versions (SARSA and QLearning) are dominated by their corresponding planning versions. This is expected since the simulated experience helps the planning algorithms to learn faster than TD Learning algorithms.

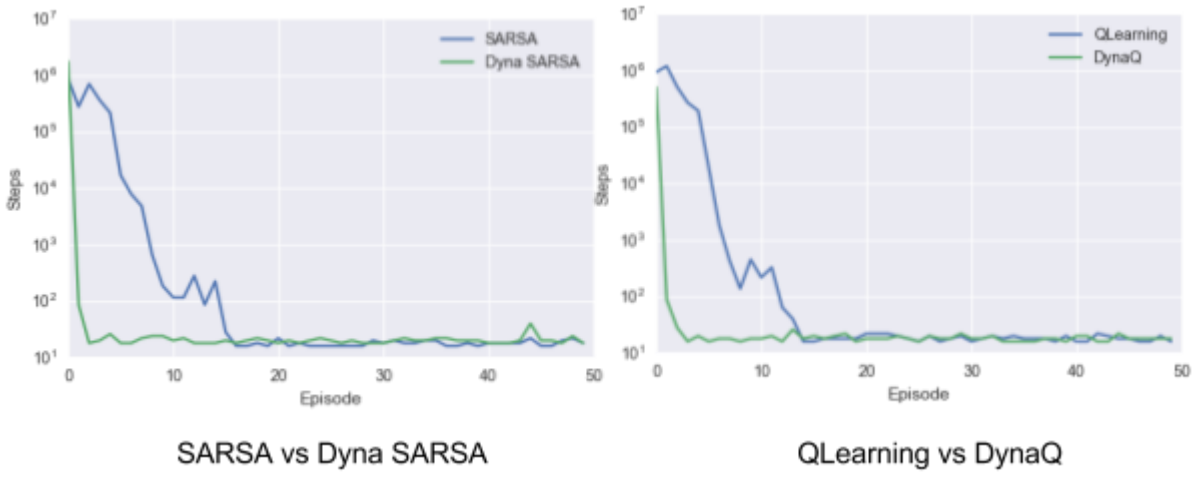


Fig 12 Steps per Episode comparison for TD Learning and Tabular Planning algotihms

Conclusions

- Dyna Planning versions of Temporal Difference Learning algorithms SARSA and QLearning converge faster
- Above mentioned behavior is not affected by environment (Cliff Walking or Dyna Maze)
- Dyna Planning versions might learn different policy than TD Learning versions (Like in case of Cliff Walking)

References

[1] Reinforcement Learning: An Introduction