AE598 Homework 1 : Dynamic Programming

Manav Vora*
University of Illinois Urbana Champaign, Urbana, Illinois, 61801

This document contains the final results obtained by implementing different reinforcement learning algorithms, in a tabular setting, on two different environments namely, Gridworld and Pendulum.

I. Introduction

We apply five reinforcement learning algorithms in a tabular setting i.e. small finite state and action spaces. The algorithms are policy iteration, value iteration, SARSA, Q-learning and TD(0). Policy iteration and value iteration are model-based algorithms and can only be applied to problems where we know the model of the underlying Markov Decision Process (MDP). SARSA, Q-Learning and TD(0) are model-free algorithms and are useful when we don't know the model of the underlying MDP. Out of the three model-free algorithms, SARSA and Q-Learning are used to calculate the optimal Q-function and hence the optimal policy whereas TD(0) is used to compute the value function associated with a given policy.

In this homework, we consider two environments:

- Gridworld : The model for the MDP is known
- Pendulum : The model for the MDP is unknown

II. Problem Statement

In this section, we briefly describe the enivronments on which we apply the reinforcement learning algorithms.

A. Gridworld

This environment is based on an example given in Sutton and Barto [1]. Figure 1 (left) shows a rectangular gridworld representation of a simple finite MDP. The MDP has 25 states, each being a cell of the grid. There are 4 possible actions at each cell : up, down, left, right and these actions result in deterministic transitions to certain cells. The agent receives a negative reward of -1 for actions which make it leave the grid (however the location is still unchanged). Other actions result in a reward of 0 except for those which move the agent out of special states A and B (1 (left)). All 4 actions yield reward of +10 from A and from state B, all 4 actions yield a reward of +5. Also, we have an explicit model of the MDP. Hence, we will apply all four of policy iteration, value iteration, Q-learning and SARSA to compute the optimal policy for the gridworld. We will also apply TD(0) to compute the approximate value function associated with SARSA and Q-Learning. Theoretically, all these algorithms should generate approximately the same value functions and optimal policies.

Based on the above description, intuitively, the optimal policy should be such that the agent keeps transitioning between states A and A'.

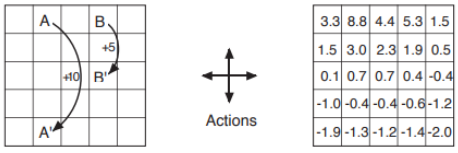


Fig. 1 Gridworld Environment

*Department of Aerospace Engineering, mkvora2@illinois.edu

B. Discrete Pendulum

This environment is based on the simple pendulum chapter in [2]. The MDP represents a simple pendulum with discretized state and action spaces. The explicit model for the MDP is not available and hence we can only apply model-free methods to compute the optimal policy. The state vector at each time instant consists of two components : the angle of the pendulum θ and the angular velocity $\dot{\theta}$. The actions are different values of torque to be applied to the pendulum. The agent gets a reward of 1 if the pendulum is upright and 0 otherwise. Also, it incurs a large negative reward for exceeding the maximum specified angular velocity. Hence the goal of the agent is to stabilize the pendulum in the upright position. We will apply SARSA and Q-Learning algorithms to this problem. We will also apply TD(0) to compute the approximate value function associated with SARSA and Q-Learning. Intuitively, the angle θ should converge to 0 after sometime and the angular velocity $\dot{\theta}$ should never exceed the limit $\dot{\theta}_{max} = 15$.

III. Results

In this section, we will show the different plots obtained for the various reinforcement learning algorithms applied on gridworld and discrete pendulum environments.

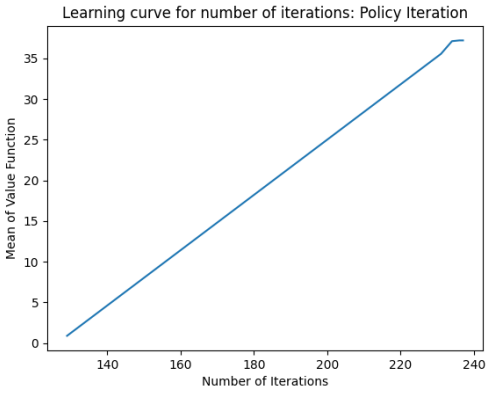
A. Gridworld

The value of discount factor $\gamma = 0.95$ for all the algorithms.

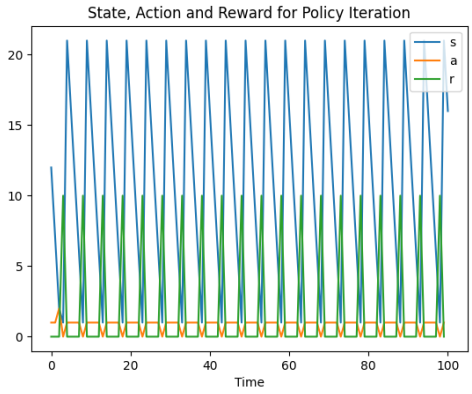
1. Policy Iteration

For policy iteration, we have two plots (Figure 2):

- A learning curve with the mean value function on the y-axis and the number of iterations on the x-axis (Figure 2a). The number of iterations here are the ones spent on policy evaluation.
- A plot for an example trajectory and the optimal policy represented by the sequence of states, actions and reward (Figure 2b).



(a) Learning curve for Policy Iteration



(b) Example trajectory and optimal policy for Policy Iteration

Fig. 2 Plots for Policy Iteration

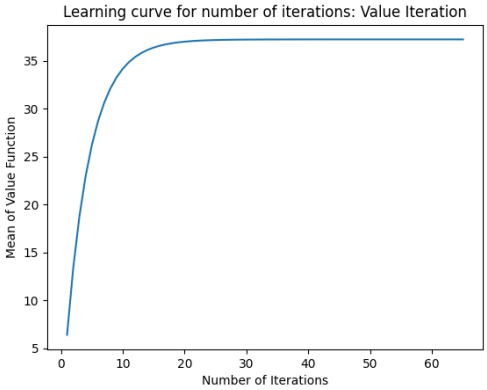
As can be clearly observed, the optimal policy results in the agent switching between states A and A' because that results in the highest return. Also, the mean value function increases with the number of iterations and after a certain number of iterations, saturates to the optimal value.

2. Value Iteration

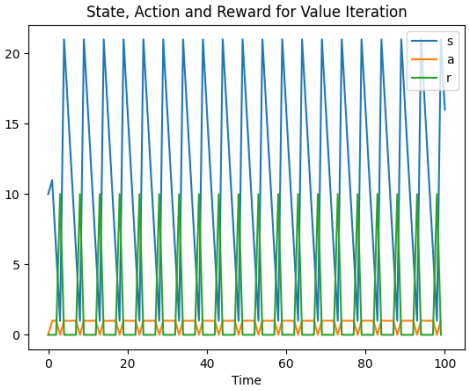
For value iteration, we have two plots (Figure 3):

- A learning curve with the mean value function on the y-axis and the number of iterations on the x-axis (Figure 3a).

- A plot for an example trajectory and the optimal policy represented by the sequence of states, actions and reward (Figure 3b).



(a) Learning curve for Value Iteration



(b) Example trajectory and optimal policy for Value Iteration

Fig. 3 Plots for Value Iteration

These plots are similar to the ones obtained for policy iteration except for the fact that the mean value function converges to the optimal value much faster.

3. SARSA

For SARSA, we have four plots (Figure 4):

- A learning curve with the return on the y-axis and the number of episodes on the x-axis (Figure 4a).
- A plot for an example trajectory and the optimal policy represented by the sequence of states, actions and reward (Figure 4b).
- A plot of learning curves for different values of parameter α (Figure 4c).
- A plot of learning curves for different values of parameter ϵ (Figure 4d).

For generating the optimal policy, the parameters used for SARSA are:

- Number of episodes = 5000
- $\alpha = 0.5$
- $\epsilon = 0.1$

As can be clearly observed, the optimal trajectory and policy is similar to that obtained for policy and value iteration. Furthermore, the return increases and fluctuates around the optimal value after a certain number of episodes. Also $0 < \epsilon < 0.3$ results in higher relative return. Similarly, α values around 0.5 result in higher relative return.

4. Q-Learning

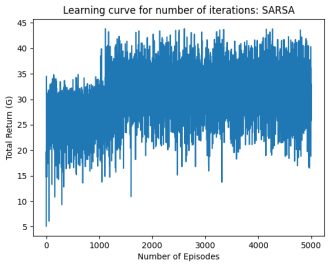
For Q-Learning, we have four plots (Figure 5):

- A learning curve with the return on the y-axis and the number of episodes on the x-axis (Figure 5a).
- A plot for an example trajectory and the optimal policy represented by the sequence of states, actions and reward (Figure 5b).
- A plot of learning curves for different values of parameter α (Figure 5c).
- A plot of learning curves for different values of parameter ϵ (Figure 5d).

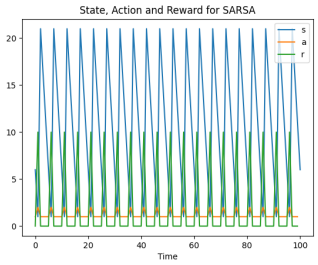
For generating the optimal policy, the parameters used for Q-Learning are:

- Number of episodes = 5000
- $\alpha = 0.5$
- $\epsilon = 0.1$

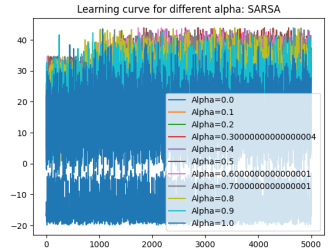
As can be clearly observed, the optimal trajectory and policy is similar to that obtained for policy and value iteration. Inferences for the learning curve as well as the α and ϵ sweeps are similar to those for SARSA.



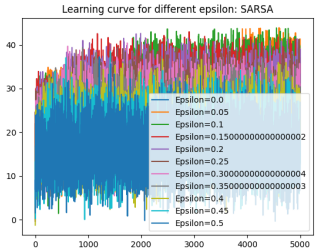
(a) Learning curve for SARSA



(b) Example trajectory and optimal policy for SARSA

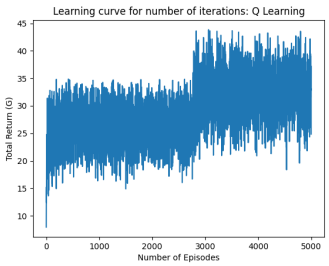


(c) Learning curves for different α

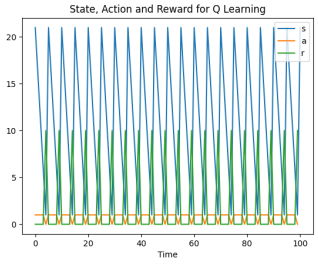


(d) Learning curves for different ϵ

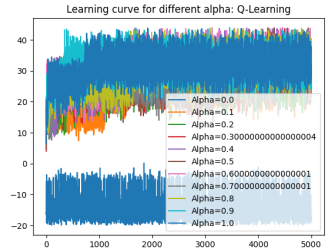
Fig. 4 Plots for SARSA



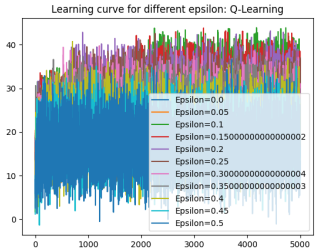
(a) Learning curve for Q-Learning



(b) Example trajectory and optimal policy for Q-Learning



(c) Learning curves for different α



(d) Learning curves for different ϵ

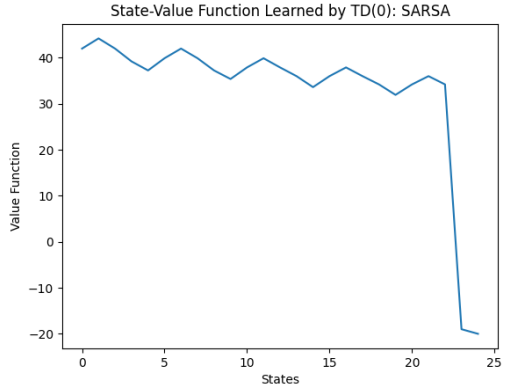
Fig. 5 Plots for Q-Learning

5. TD(0) Approximation

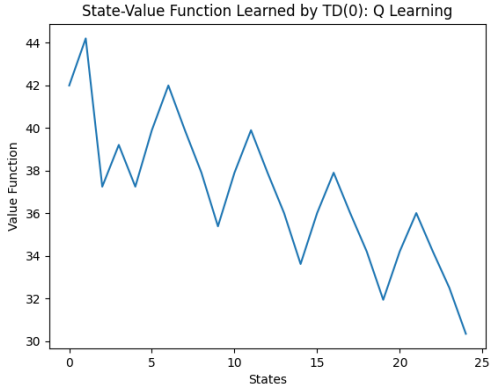
For TD(0), we have two plots (Figure 6):

- Plot of state-value function, for SARSA, with the value of the value function on y-axis and state on the x-axis (Figure 6a).
- Plot of state-value function, for Q-Learning, with the value of the value function on y-axis and state on the x-axis

(Figure 6b).



(a) State-Value Function for SARSA



(b) State-Value Function for Q-Learning

Fig. 6 State-Value Function for Q-Learning

The parameters used for TD(0) are:

- Number of episodes = 5000
- $\alpha = 0.5$

As can be clearly seen, TD(0) approximates the optimal value function pretty accurately apart from states 24 and 25 for SARSA.

B. Discrete Pendulum

The value of discount factor $\gamma = 0.95$ for all the algorithms.

1. SARSA

For SARSA, we have five plots (Figure 7):

- A learning curve with the return on the y-axis and the number of episodes on the x-axis (Figure 7a).
- A plot for an example trajectory and the optimal policy represented by the sequence of states, actions and reward (Figure 7b).
- A plot of learning curves for different values of parameter α (Figure 7c).
- A plot of learning curves for different values of parameter ϵ (Figure 7d).
- A plot of θ and $\dot{\theta}$ vs time (Figure 7e).

For generating the optimal policy, the parameters used for SARSA are:

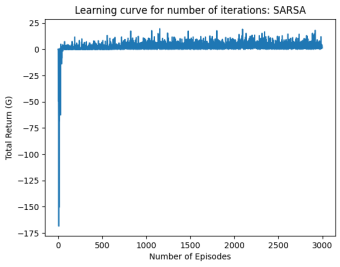
- Number of episodes = 3000
- $\alpha = 0.3$
- $\epsilon = 0.4$

As can be clearly observed, the pendulum angle θ converges close to zero with time. Furthermore, the angular velocity never crosses the maximum. Also initially the agent gets a large negative return but as number of episodes increase, the gets higher positive return.

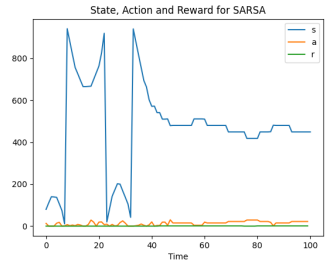
2. Q-Learning

For Q-Learning, we have five plots (Figure 8):

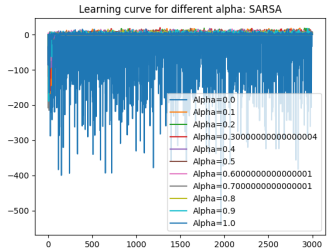
- A learning curve with the return on the y-axis and the number of episodes on the x-axis (Figure 8a).
- A plot for an example trajectory and the optimal policy represented by the sequence of states, actions and reward (Figure 8b).
- A plot of learning curves for different values of parameter α (Figure 8c).
- A plot of learning curves for different values of parameter ϵ (Figure 8d).
- A plot of θ and $\dot{\theta}$ vs time (Figure 8e).



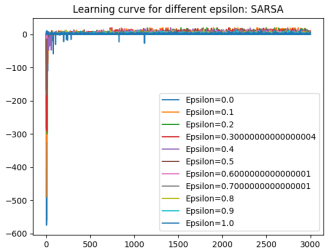
(a) Learning curve for SARSA



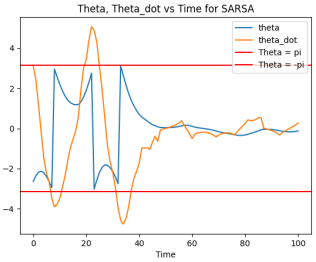
(b) Example trajectory and optimal policy for SARSA



(c) Learning curves for different α



(d) Learning curves for different ϵ



(e) θ and $\dot{\theta}$ vs Time for SARSA

Fig. 7 Plots for SARSA

For generating the optimal policy, the parameters used for Q-Learning are:

- Number of episodes = 3000
- α = 0.3
- ϵ = 0.4

As can be clearly observed, the pendulum angle θ converges close to zero with time and this convergence is faster than SARSA. Furthermore, the angular velocity never crosses the maximum. Also initially the agent gets a large negative return but as number of episodes increase, the gets higher positive return. Finally, the alpha sweep was done only for three values of α so as to save on time. As can be seen, values of α close to 0.5 will give the highest return.

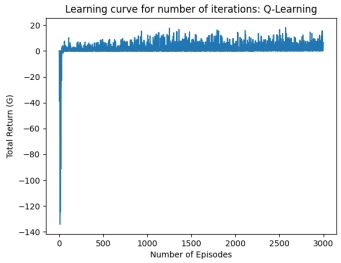
3. TD(0) Approximation

For TD(0), we have two plots (Figure 9):

- Plot of state-value function, for SARSA, with the value of the value function on y-axis and state on the x-axis (Figure 9a).
- Plot of state-value function, for Q-Learning, with the value of the value function on y-axis and state on the x-axis (Figure 9b).

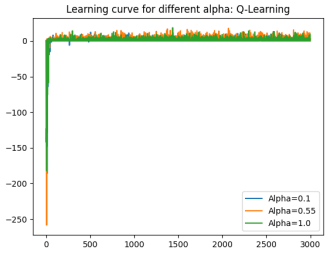
The parameters used for TD(0) are:

- Number of episodes = 3000
- α = 0.3

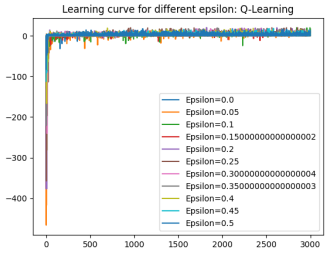


(a) Learning curve for Q-Learning

(b) Example trajectory and optimal policy for Q-Learning



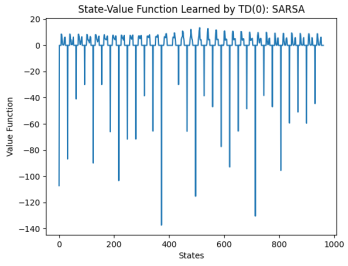
(c) Learning curves for different α



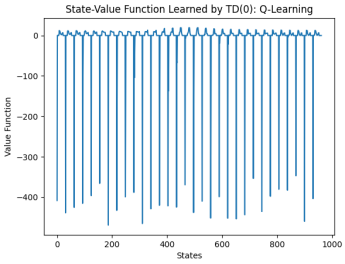
(d) Learning curves for different ϵ

(e) θ and $\dot{\theta}$ vs Time for Q-Learning

Fig. 8 Plots for Q-Learning



(a) State-Value Function for SARSA



(b) State-Value Function for Q-Learning

Fig. 9 State-Value Function TD(0)

References

[1] Sutton, R. S., and Barto, A. G., *Reinforcement Learning: An Introduction*, 2nd ed., The MIT Press, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[2] Tedrake, R., *Underactuated Robotics*, 2023. URL https://underactuated.csail.mit.edu.