AE 598-RL HW1 Report

Gokul Puthumanaillam (gokulp2)
Department of Aerospace Engineering, University of Illinois Urbana-Champaign

I. Problem Description

A. Grid World

The setting for this environment is a grid layout, comprising of 5 columns and 5 rows. The entire grid comprises of 25 different states, each representing a unique location. There are four feasible actions that can be taken at each state: right, up, left, and down. The environment's transition model is defined, and all algorithms have access to it. The objective of the algorithms is to learn how to navigate through the environment and find a path that maximizes the episode reward. This task will require the algorithms to identify the optimal sequence of actions to take in order to achieve the highest reward possible.

B. Inverted Pendulum

This environment comprises a simple pendulum with a discretized state and action space. The transition model for this environment is not explicitly available. The state of the pendulum is determined by the angle θ displacement from the vertical axis. The maximum allowed angular velocity of the pendulum is fixed (exceeding which, a large negative reward is incurred). The objective of the algorithm is to maintain the pendulum's state within a specific goal range, which represents an upright position, and rewards a score of +1. Any state outside the goal range earns a score of 0. Therefore, the algorithms will be trained to maintain the pendulum inverted for as many time-steps as possible during an episode. This will require the algorithm to balance the pendulum by selecting the appropriate actions at each time step.

II. Preliminaries

This report documents the generated graphs for the following algorithms in the Grid World and the Discrete Pendulum (in a tabular setting):

1) Value Iteration: Grid World
2) Policy Iteration: Grid World
3) SARSA: Grid World and Discrete Pendulum
4) QLearning: Grid World and Discrete Pendulum

Algorithm 1: Value Iteration

Input: MDP $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$
Output: Optimal value function V^*
Initialization: $V_0(s) \leftarrow 0, \forall s \in \mathcal{S}$;
for $i = 1$ **to** ∞ **do**
  for $s \in \mathcal{S}$ **do**
    $V_i(s) \leftarrow \max_{a \in \mathcal{A}} \left(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V_{i-1}(s')\right)$;
  end
  if $\|V_i - V_{i-1}\|_\infty < \epsilon$ **then**
    Break;
  end
end
return V_i

Algorithm 2: Policy Iteration

Input: MDP $M = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$
Output: Optimal policy π^*
Initialization: $\pi_0(s) \leftarrow$ arbitrary, $\forall s \in \mathcal{S}$;
for $i = 0$ **to** ∞ **do**
  Policy Evaluation:;
    $V^{\pi_i}(s) \leftarrow \sum_{a\in\mathcal{A}} \pi_i(a|s) \left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)V^{\pi_i}(s')\right), \forall s \in \mathcal{S}$;
  Policy Improvement:;
    $\pi_{i+1}(s) \leftarrow \arg\max_{a\in\mathcal{A}} \left(R(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)V^{\pi_i}(s')\right), \forall s \in \mathcal{S}$;
  if $\pi_{i+1} = \pi_i$ **then**
    Break;
  end
end
return π_i

Algorithm 3: SARSA

Input: Environment with state-action pairs Q, initial state s, exploration rate ϵ, learning rate α, discount factor γ
Output: Learned action-value function Q
Initialization: $Q(s,a) \leftarrow$ arbitrary, $\forall a$;
Repeat for each episode:;
  Initialize s;
  Choose a using ϵ-greedy policy based on Q;
  Repeat for each step:;
    Take action a, observe r and s';
    Choose a' using ϵ-greedy policy based on Q;
    $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$;
    $s \leftarrow s'$, $a \leftarrow a'$;
  Until s is terminal;
return Q

Algorithm 4: Q-learning

Input: Environment with state-action pairs Q, initial state s, exploration rate ϵ, learning rate α, discount factor γ
Output: Learned action-value function Q
Initialization: $Q(s,a) \leftarrow$ arbitrary, $\forall a$;
Repeat for each episode:;
  Initialize s;
  Repeat for each step:;
    Choose a using ϵ-greedy policy based on Q;
    Take action a, observe r and s';
    $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$;
    $s \leftarrow s'$;
  Until s is terminal;
return Q

III. Hyperparameters:

A. Grid World

1. Value Iteration and Policy Iteration

1) Max Number of episodes: 5000
2) γ: 0.95
3) θ=1e-6

2. *SARSA and QLearning*
   1) Max Number of episodes: 5000
   2) γ: 0.95
   3) θ: 1e-6
   4) ϵ: 0.1

B. Discrete Pendulum

1. *SARSA and QLearning*
   1) Max Number of episodes: 700
   2) γ: 0.95
   3) θ: 1e-6
   4) ϵ: 0.1

IV. Plots

A. Grid World

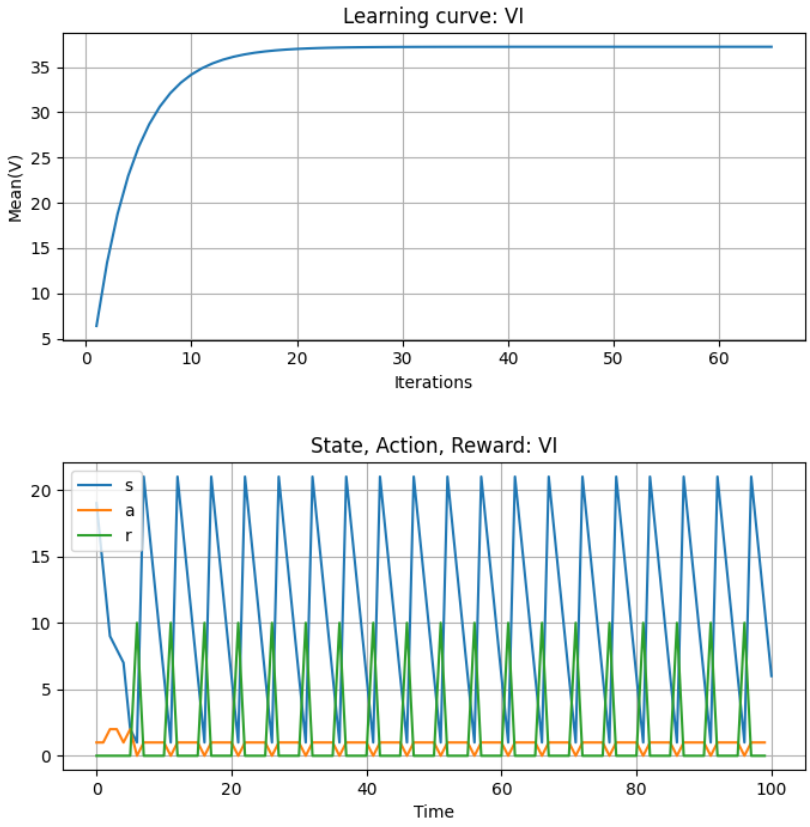

Fig. 1 Plots showing the learning curve for Value Iteration and the State, Action, Rewards

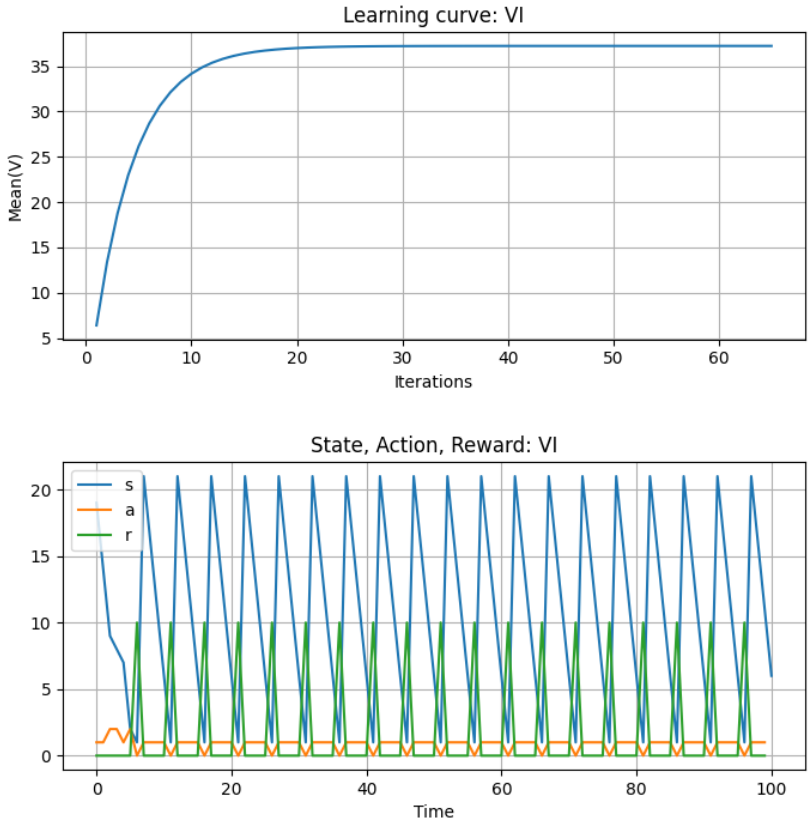

Fig. 2 Plots showing the learning curve for Policy Iteration and the State, Action, Rewards

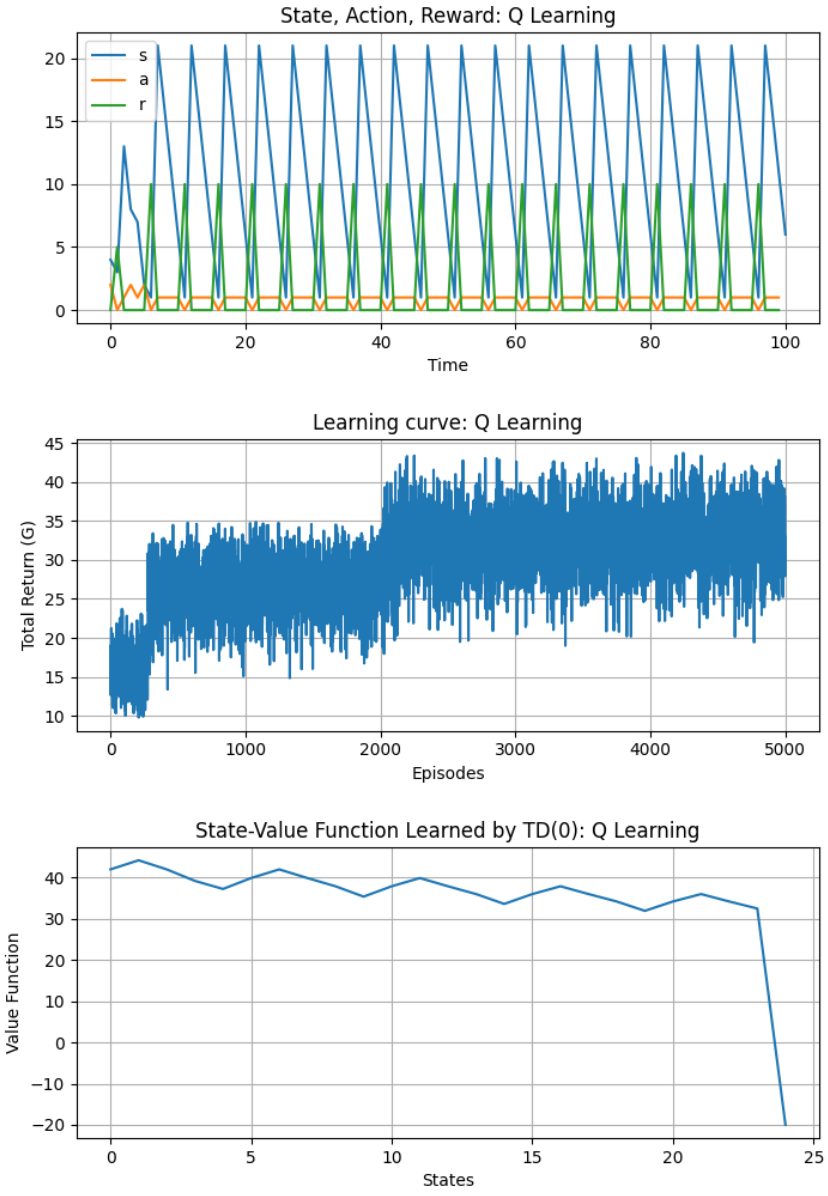

Fig. 3 Plots for QLearning: (a)State, Action, Reward (b) Learning Curve (c) State Value Function Estimation using TD(0)

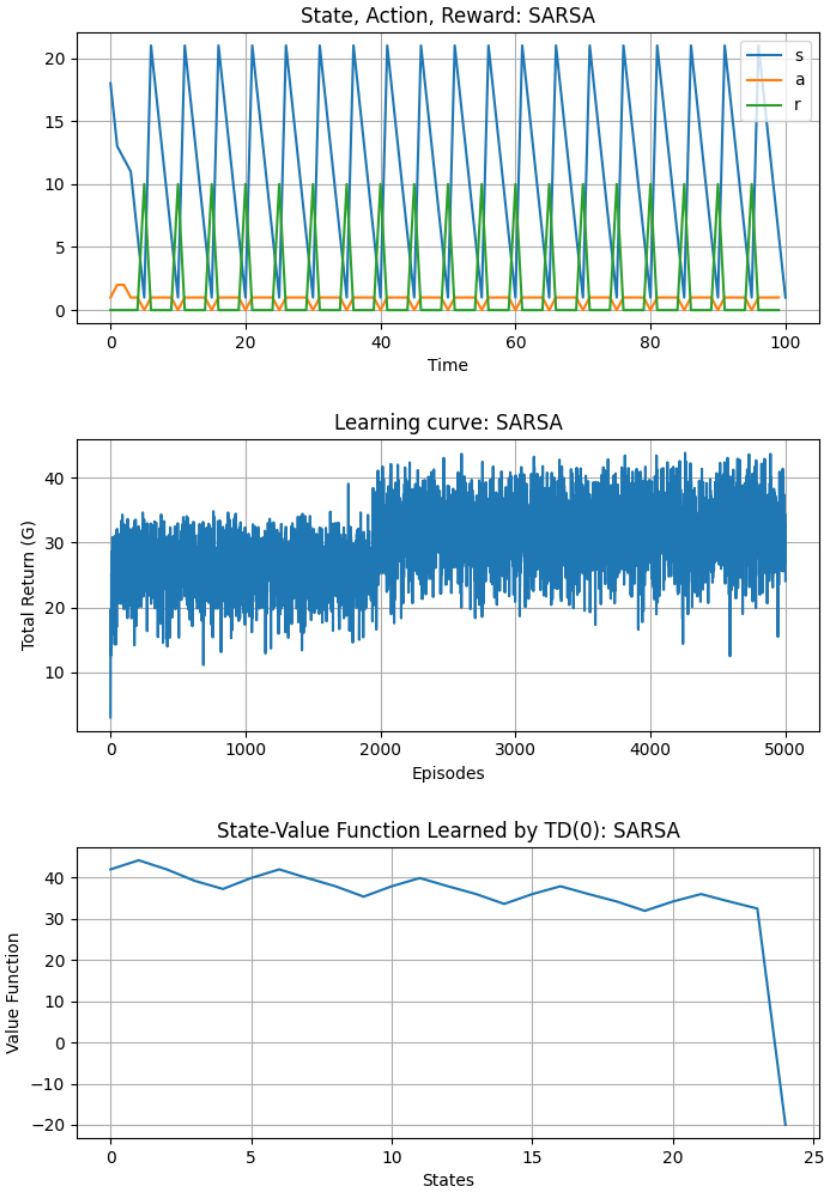

Fig. 4 Plots for SARSA: (a)State, Action, Reward (b) Learning Curve (c) State Value Function Estimation using TD(0)

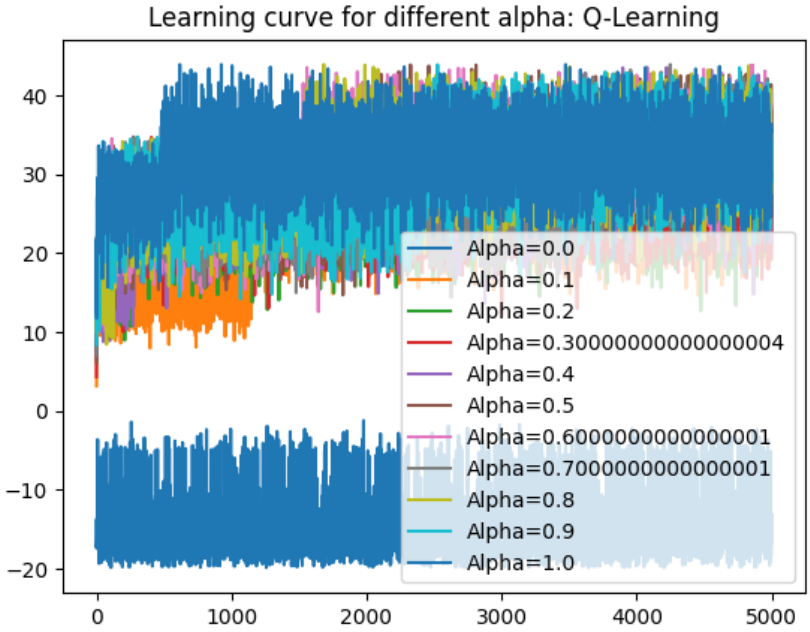

Fig. 5 Learning Curve for different Alpha: QLearning

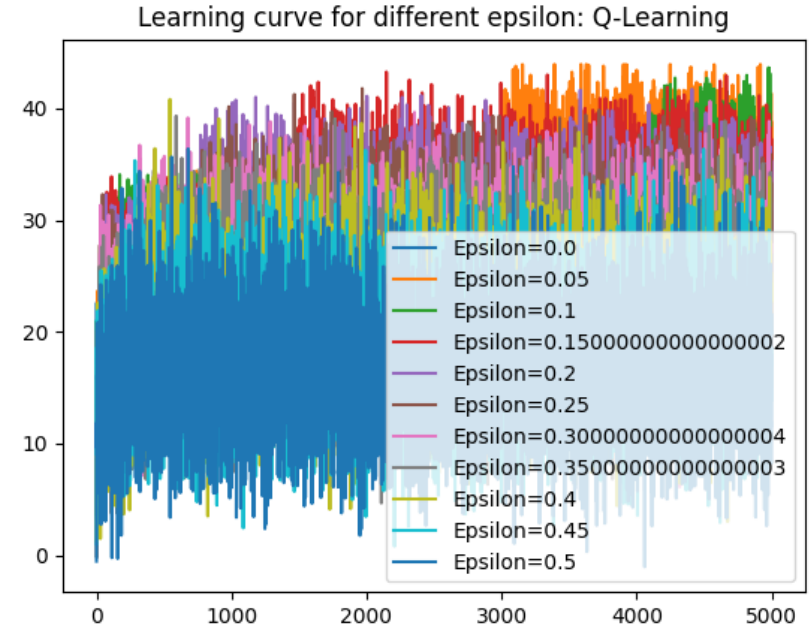

Fig. 6 Learning Curve for different Epsilon: QLearning

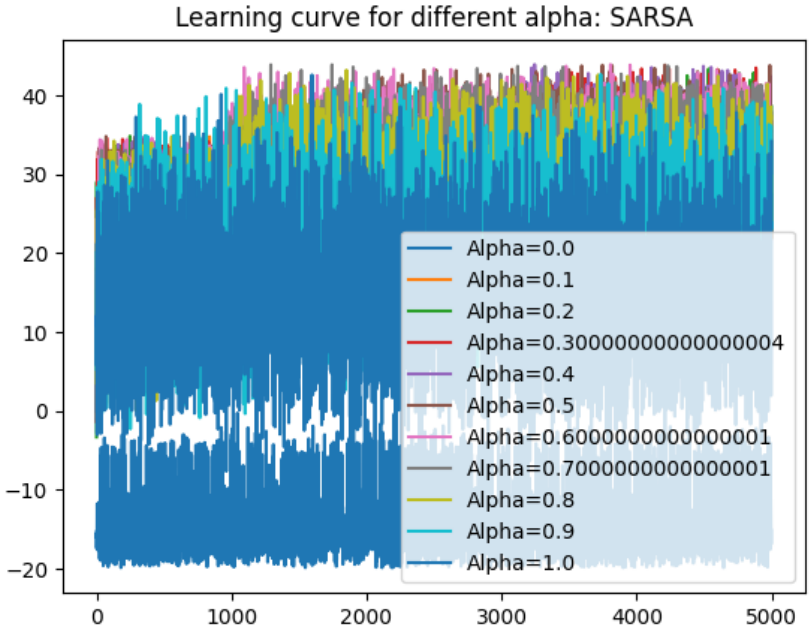

Fig. 7 Learning Curve for different Alpha: SARSA

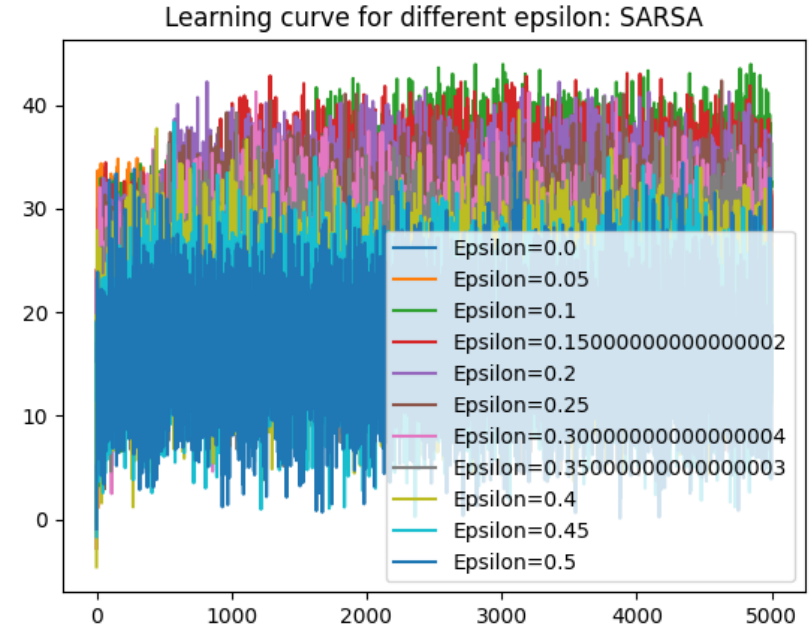

Fig. 8 Learning Curve for different Epsilon: SARSA

B. Discrete Pendulum

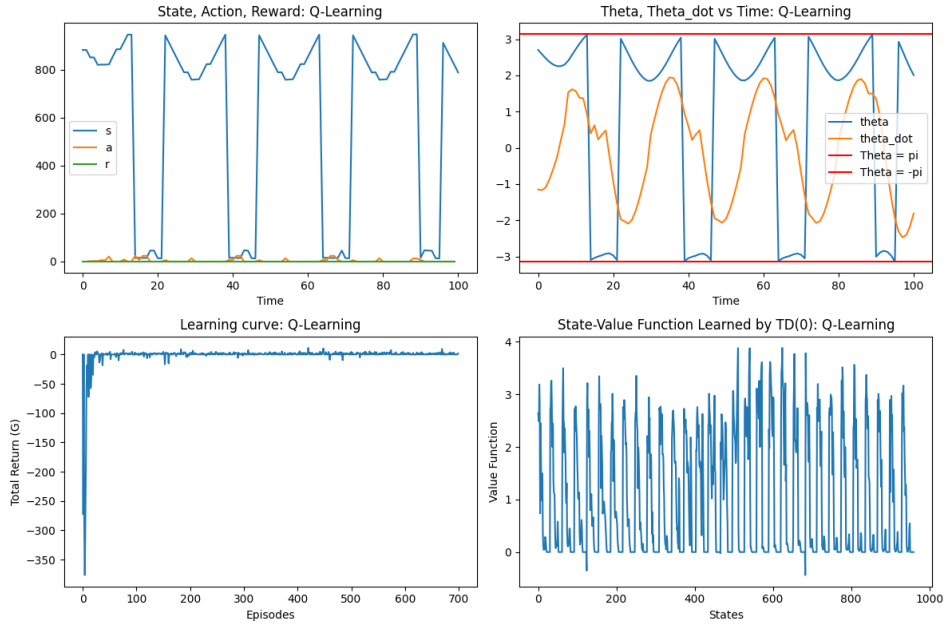

Fig. 9 Plots for QLearning: (a)State, Action, Reward (b) Theta, Theta dot vs time(c) Learning Curve (d) State Value Function Estimation using TD(0)

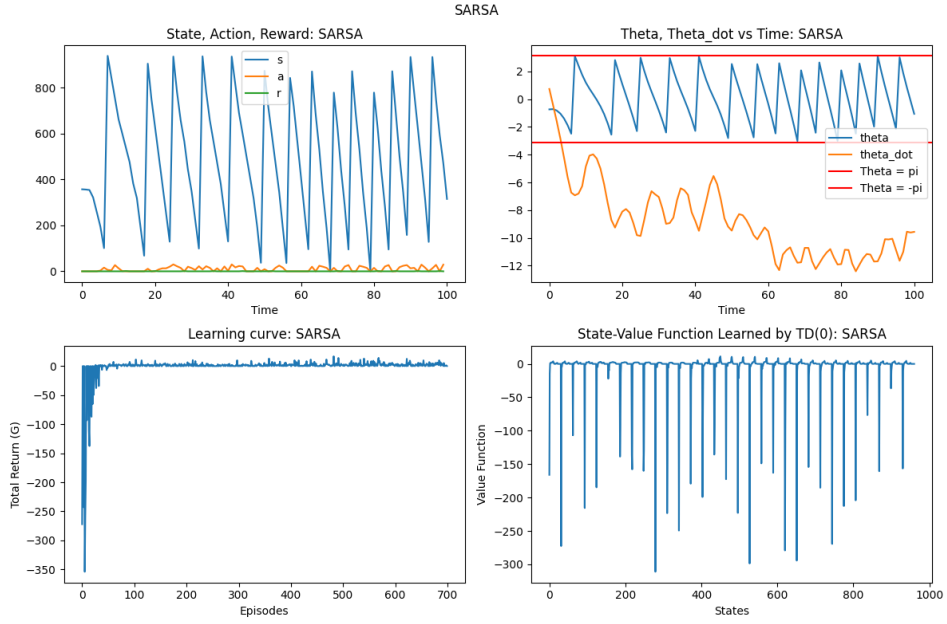

Fig. 10 Plots for SARSA: (a)State, Action, Reward (b) Theta, Theta dot vs time(c) Learning Curve (d) State Value Function Estimation using TD(0)

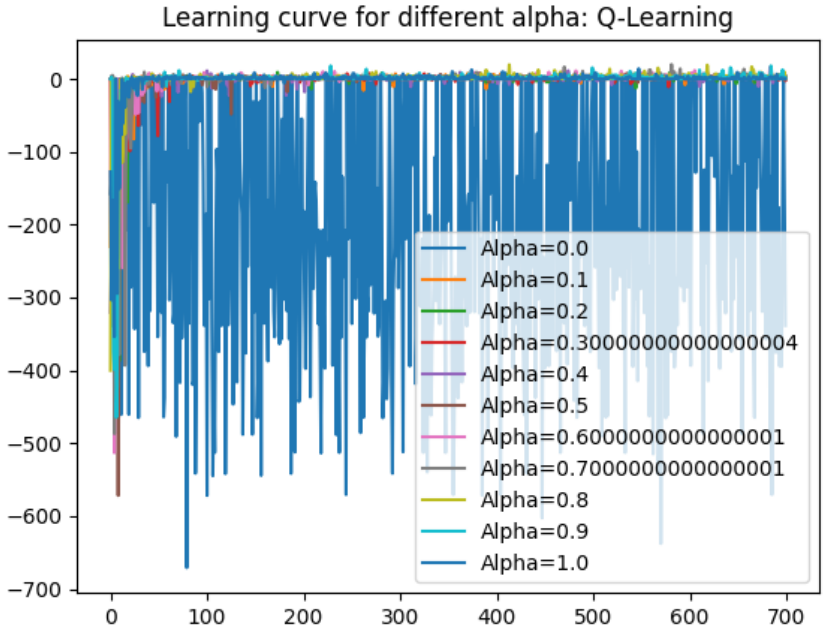

Fig. 11 Learning Curve for different Alpha: QLearning

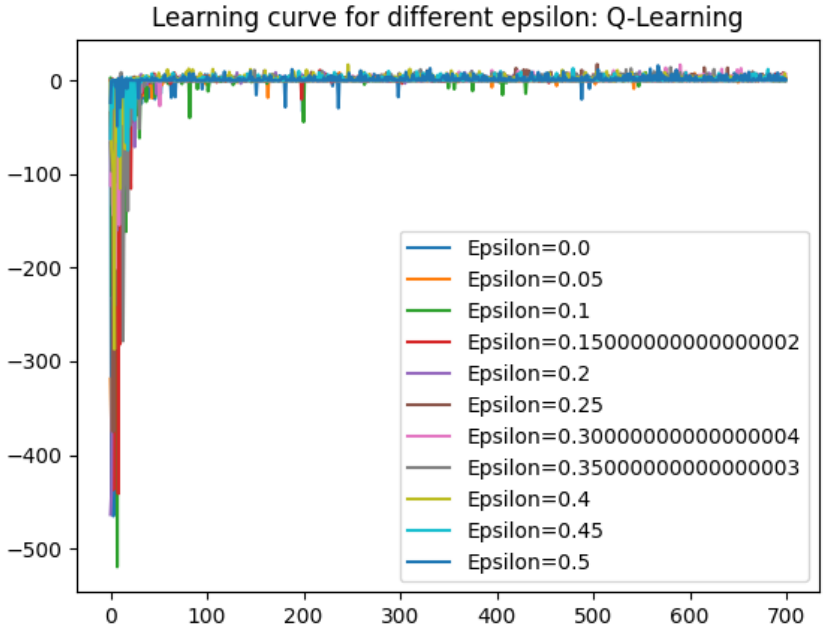

Fig. 12 Learning Curve for different Epsilon: QLearning

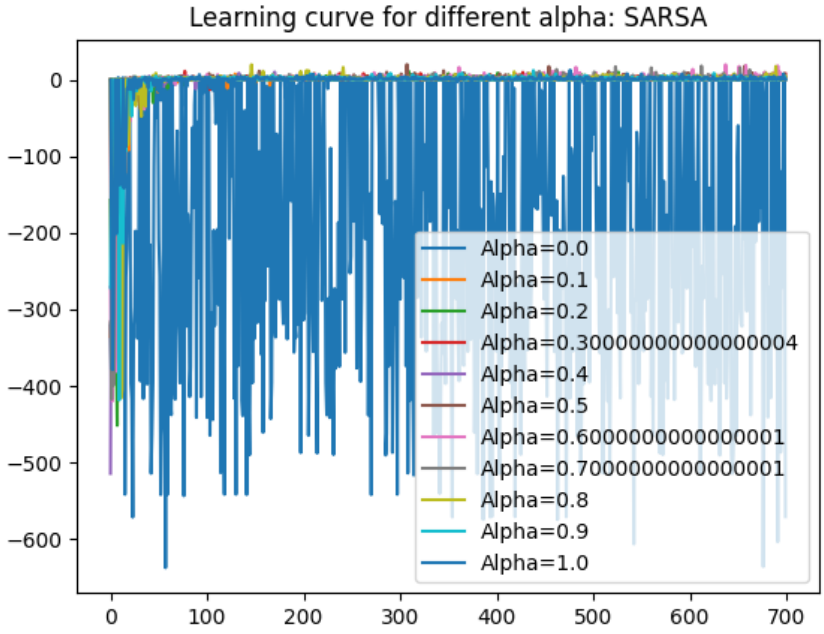

Fig. 13 Learning Curve for different Alpha: SARSA

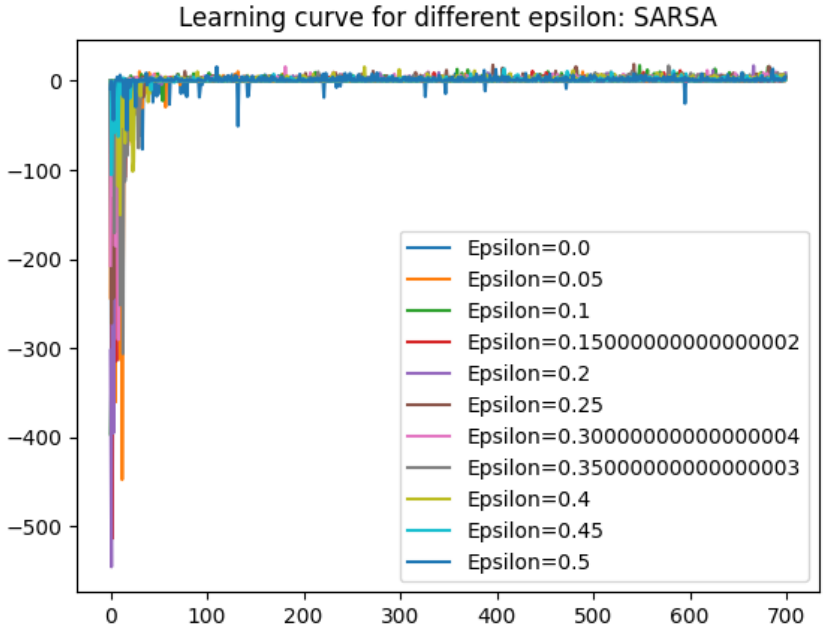

Fig. 14 Learning Curve for different Epsilon: SARSA