Assignment 4

(svyas44@gatech.edu)

I. MDP Problems and their Description

Frozen Lake MDP: The Frozen Lake problem is a Markov Decision Process (MDP) where an agent navigates a grid world containing frozen tiles, safe tiles, a goal tile, and hole tiles. The agent's goal is to reach the goal tile while avoiding the holes. Actions include moving up, down, left, or right, but the ice makes movements stochastic. Rewards are given upon reaching the goal, with no reward for falling into a hole. Transition probabilities determine the likelihood of moving from one state to another. Reinforcement learning algorithms are applied to learn optimal policies for navigating this stochastic environment.

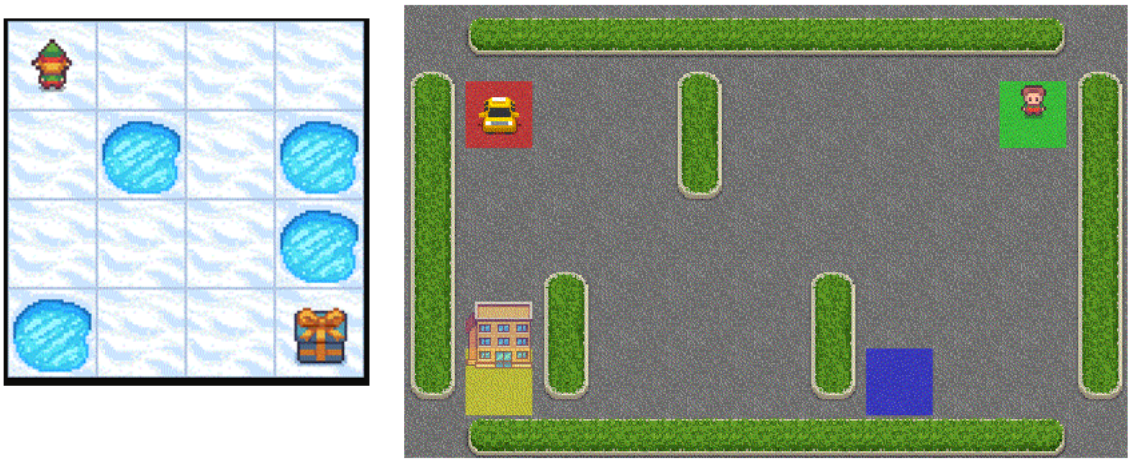


Figure 1: Frozen Lake problem (on the left) and Taxi problem (on the right)

Taxi MDP: The Taxi MDP (Markov Decision Process) is a classic problem in reinforcement learning where an agent navigates a grid-world environment as a taxi driver. The grid contains multiple locations: passenger locations (designated by letters), destination locations (also designated by letters), walls, and open spaces. The agent's objective is to pick up passengers from their current location and drop them off at their designated destination. The agent can move in four directions (up, down, left, right) and can perform actions like picking up and dropping off passengers. Rewards are given for successfully picking up and dropping off passengers, with penalties for illegal actions and for each time step. The challenge lies in learning an optimal policy to efficiently pick up and drop off passengers while minimizing the time and distance travelled.

What makes them interesting?

Both the problems varying levels of complexity (in terms of number of states and the final objectives). With most of the real-world problems being analysed in a

discretized fashion, the setup provides a quick and nice way to benchmark multiple algorithms. The environment provides sparse rewards. Personally, to me, the tasks interest me as it helps me as I am able to relate these problems closely to Robot Navigation problems, where the scenarios of *falling in hole, navigating in wrong direction, optimization of paths and respective capacity planning (of battery fuel based on optimal path)* are covered in these scenarios. The varying sizes provided a good space to experiment, where small MDP can be used to run many experiments as convergence can be achieved faster. The Taxi problem provides a inherent sequential nature of information *(pick the passenger first and then drop it at a place)* falls in the domain of continuous sequence problems, which I also find when solving structured Natural Language where sequence indicates the Rules that must be followed by a generative model one step at a time, as per legal and style guidelines.

II. Description of Algorithms and setup

I have used Value Iteration, Policy Iteration and Q-learning (as the reinforcement Learning) on the two mdp problems. Fundamentally, all the algorithms can be described their own version of Bellman's Equations:

1. Bellman Expectation Equation:

$$V(s)=\sum_a \pi(a|s) \sum_{s',r} P(s',r|s,a)[r+\gamma V(s')]$$

2. Bellman Expectation Equation:

$$Q(s,a)=\sum_{s',r} P(s',r|s,a)[r+\gamma \max_{a'} Q(s',a')]$$

Where

- V(s) represents value at state s
- Q(s,a) represents value of action a in state s
- Pi(a|s) represents probability of selecting a when in state s
- Gamma is discount factor for future rewards.
- P(S_prime, r | s, a) represents the transition probability to s_prime from s when taken action is a
- $\text{Max}_a Q(s\_prime, a\_prime)$ is maximum action value in next state s_prime under current policy

III. Experimental Results and Analysis

a. Parameter(s) Exploration

Choosing gamma: After exploring multitude of gamma values and multiple iterations, choosing more gamma helps in getting more average rewards and this helping in quick convergence by reducing Failure percentage.

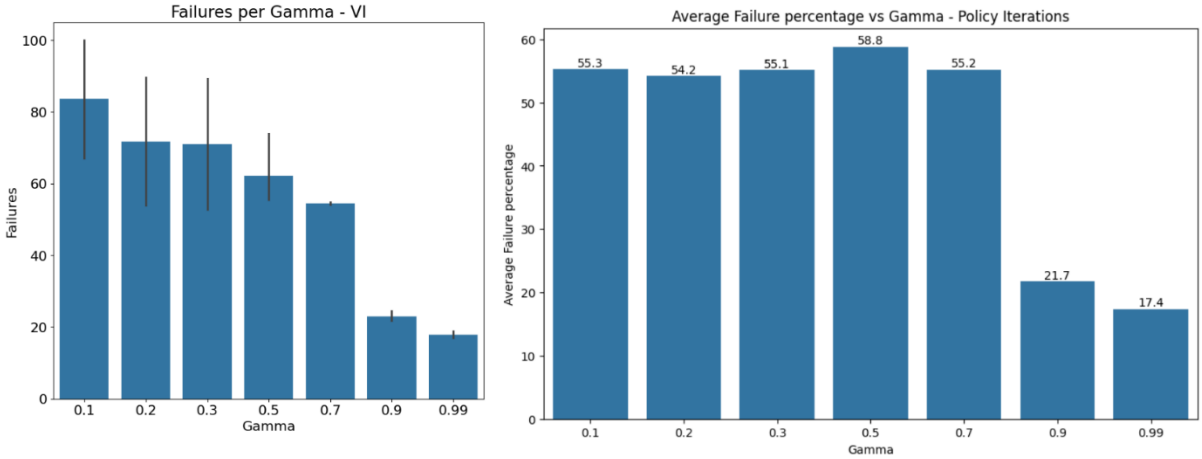


Fig 2: Choice of Gamma for both Value and Policy Iterations (same behaviour observed for both MDP)

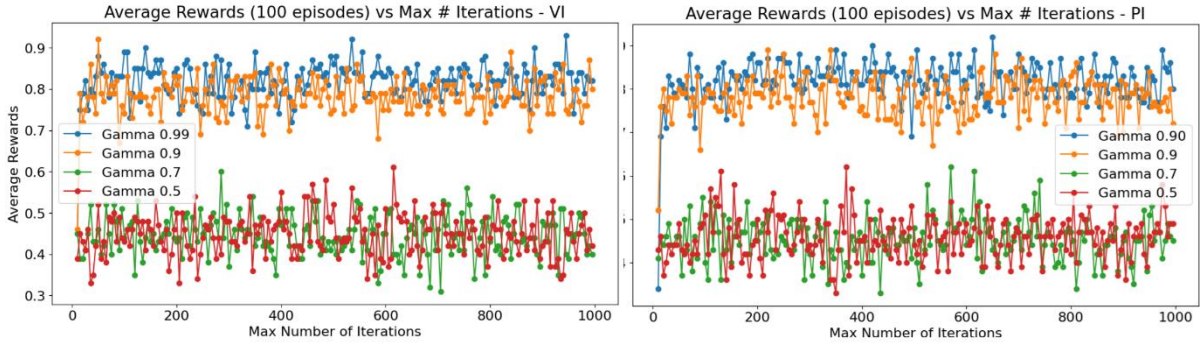


Fig 3: Average Reward as per iterations increases with increasing Gamma – Frozen Lake

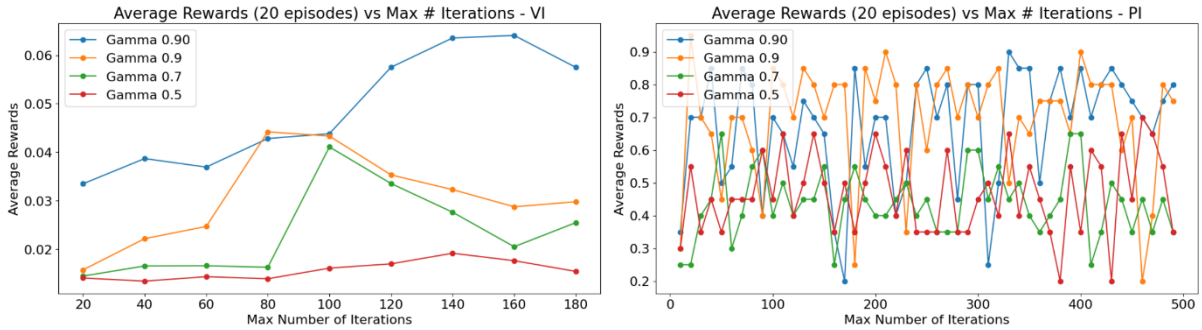


Fig 4: Average Reward as per iterations increases with increasing Gamma – Taxi

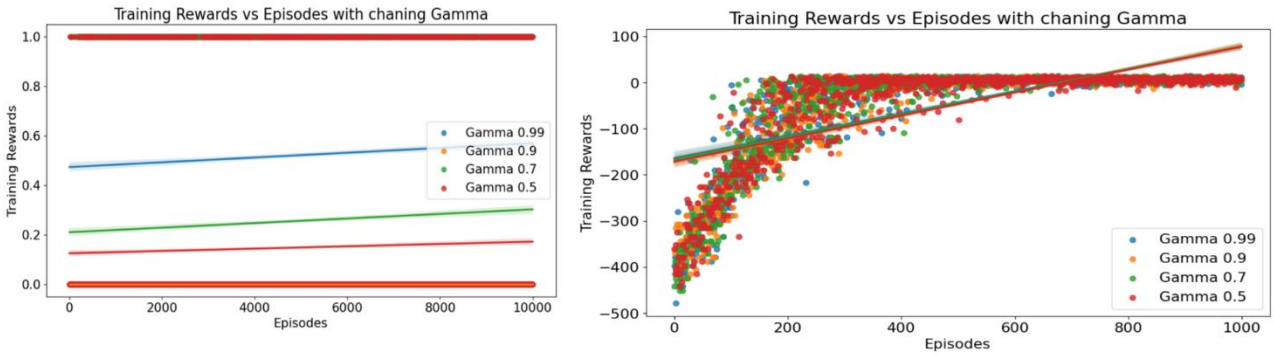


Fig 5: Mean Training Reward as per increasing Gamma in Q-learning – left for Frozen Lake and Right for Taxi

Choosing Epsilon: Lesser values of Epsilon help achieve lesser failures and better convergence.

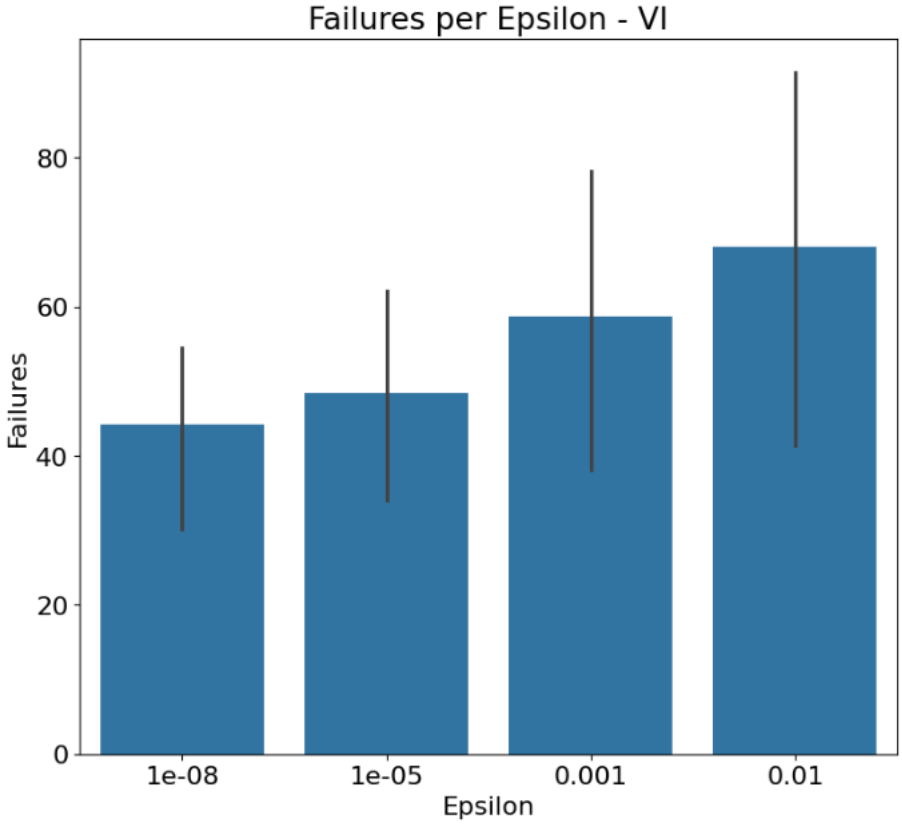


Fig 6. Failures reduction as we reduce epsilon.

Choosing Alpha for Q-learning: Increasing Alpha increases the Training Rewards across episodes.

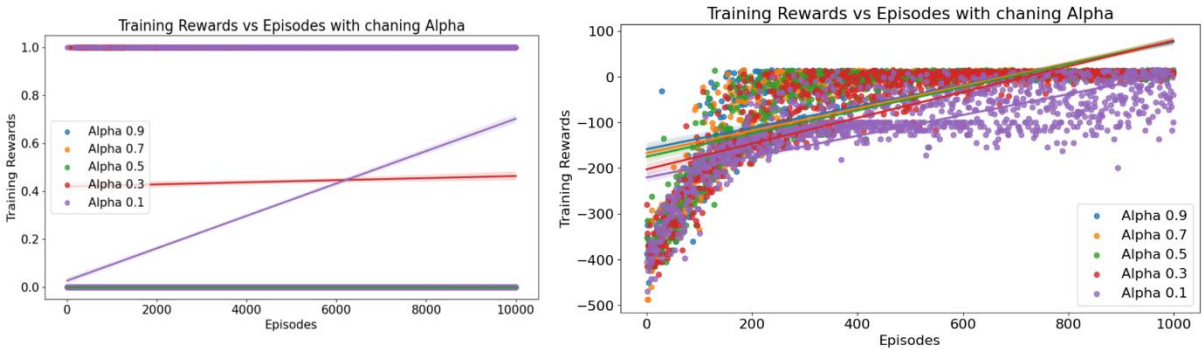


Fig 7: Mean Training Rewards vs changing values of Alpha – left for Frozen Lake and Right for Taxi

Choosing Decay in Q-Learning: Choosing higher values of Decay helps increasing Training Rewards.

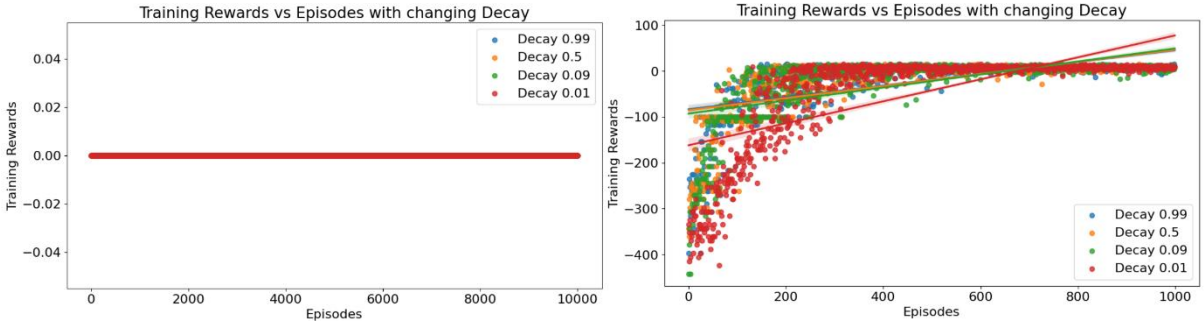


Fig 8: Mean Training Rewards vs changing values of Decay – left for Frozen Lake and Right for Taxi

Chosen Values:

Gamma	0.99
Epsilon	1e-8
Alpha	0.9
Decay	0.99

Why increasing the above values provides optimal results?

Increasing Discount Factor γ in Value Iteration and Policy Iteration: The discount factor γ controls the importance of future rewards in the computation of the value function. A higher discount factor places more emphasis on future rewards, encouraging the agent to prioritize long-term gains over short-term rewards. When γ is increased, the agent becomes more farsighted, leading to better planning and decision-making. It may choose actions that lead to higher cumulative rewards over time, even if they involve longer paths or delayed rewards. As a result, increasing γ in Value Iteration and Policy Iteration can lead to higher average rewards across iterations because the agent makes better-informed decisions that optimize long-term performance.

Increasing Learning Rate α or Decay in Q-Learning: In Q-Learning, the learning rate α determines the extent to which newly acquired information overrides existing Q-values. A higher learning rate gives more weight to new experiences, leading to faster updates of the Q-values. Additionally, decay mechanisms in Q-Learning, such as exponential decay or scheduled decay of the learning rate, gradually decrease the learning rate over time. This allows the agent to explore the environment more thoroughly in the early stages of learning and refine its actions based on accumulated experience as training progresses. Increasing the learning rate or decay in Q-Learning accelerates the learning process, enabling the agent to quickly adapt to the environment and learn optimal policies. This, in turn, can lead to higher average rewards across iterations as the agent makes better decisions based on updated Q-values.

b. Algorithm Result Comparison

Both the algorithms – Value Iteration and Policy Iteration, converge to same policy at the end in Frozen Lake and Taxi; as can be seen follows:

Convergence Defined as:

Convergence in Value Iteration, Policy Iteration, and Q-Learning refers to the point at which the iterative algorithms have reached an approximation of the optimal value function (or action-value function) or policy. Here's how convergence is defined for each algorithm:

1. Value Iteration:

Convergence in Value Iteration occurs when the change in the value function between iterations falls below a specified threshold ϵ, indicating that the values have sufficiently converged to the optimal values. Here it is defined as:

$$\max_{s \in S} |V_{k+1}(s) - V_k(s)| < \epsilon$$

$V_k(s)$ represents the value of state s at iteration k, and ϵ is a small positive threshold value, and the ϵ Is 1e-8

2. Policy Iteration:

Convergence in Policy Iteration typically involves two steps: policy evaluation and policy improvement. The algorithm iterates between these steps until the policy no longer changes between iterations, indicating that the policy has converged to the optimal policy. Here it is defined as:

$$\pi_{k+1}(S) = \pi_k(S)$$

Here, $\pi_k(s)$ represents the policy at iteration

3. Q-Learning:

Convergence in Q-Learning is typically based on the convergence of the action-value function (Q-function). It occurs when the Q-values have converged to the optimal Q-values. One common convergence criterion is to monitor the change in Q-values between iterations and halt the algorithm when the change falls below a certain threshold. Here it is defined as:

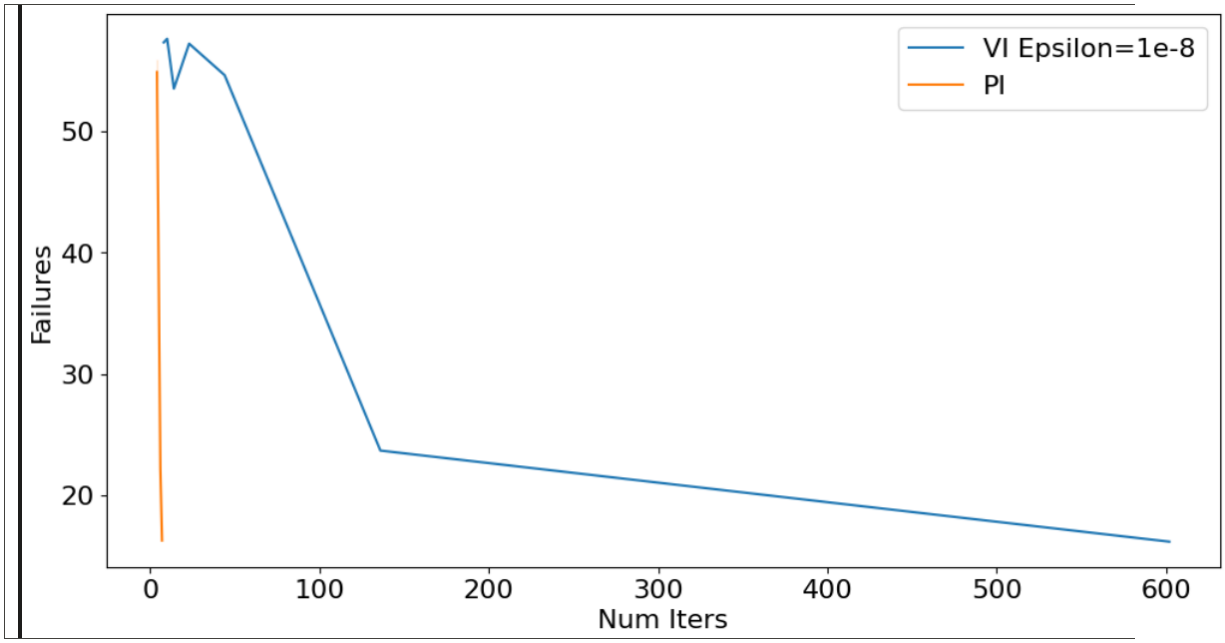
$$\max_{s,a}|Q_{k+1}(s,a)-Q_k(s,a)|<\epsilon$$

Here Qk(s,a) represents Q-value of action a in state s at iteration k and ϵ is positive threshold value (here 1e-8)

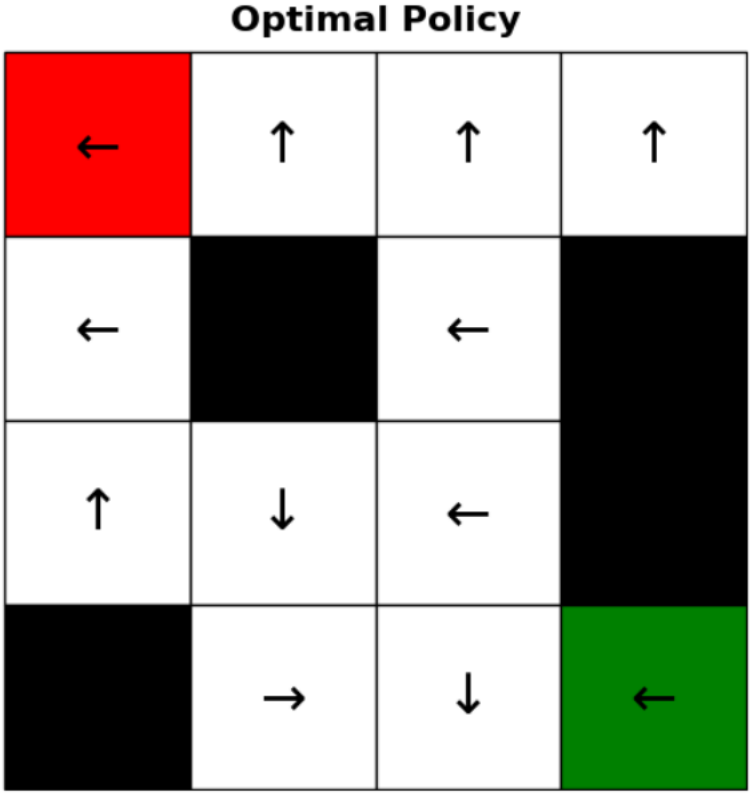
Here

Convergence Sequence in terms of Speed:

Policy Iteration converges faster than Value Iteration, and Q-Learning takes even more time.



All of the policies converge the same policy:



To measure the convergence, we use Mean squared Error w.r.t V as follows: