COL864 Special Topics in AI
Assignment-2

Somanshu Singla 2018EE10314
Lakshya Kumar Tangri 2018EE10222

April 12, 2022

Reinforcement Learning from Value Iteration to Balanced Wandering to Q-Learning

Contents

1 Question 1 : MDP

In this problem our agent is robot moving in discrete 2-D grid and recieves rewards as specified. In this problem we have defined our state $s = (x, y)$ to be defined using position of the agent. A grid describes the state space with a transition and reward function. Action space is represented by actions dictionary consisting of 4 keys with corresponding movements. The grid can be seen below

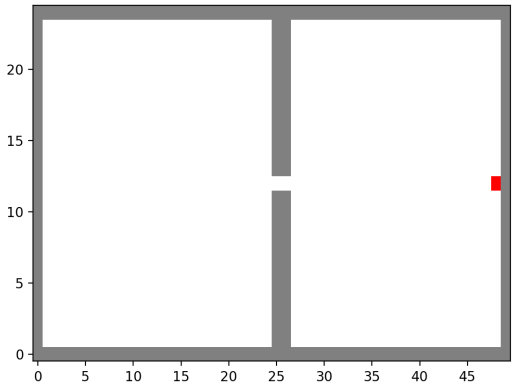


Figure 1: Grid

1.1 (a) Value Iteration

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
| $\Delta \leftarrow 0$
| Loop for each $s \in \mathcal{S}$:
| $v \leftarrow V(s)$
| $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
| $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$

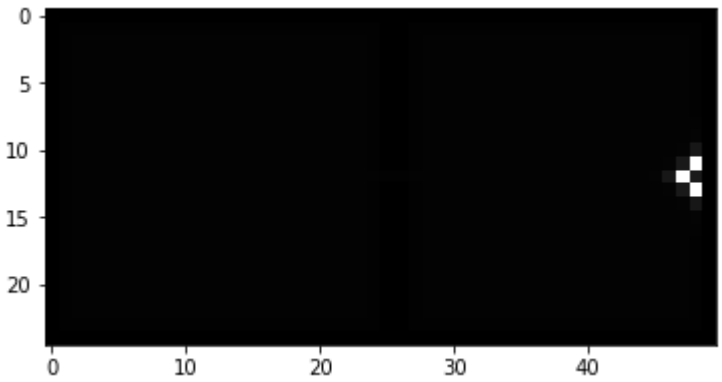


Figure 2: Value Function at $\gamma = 0.1$

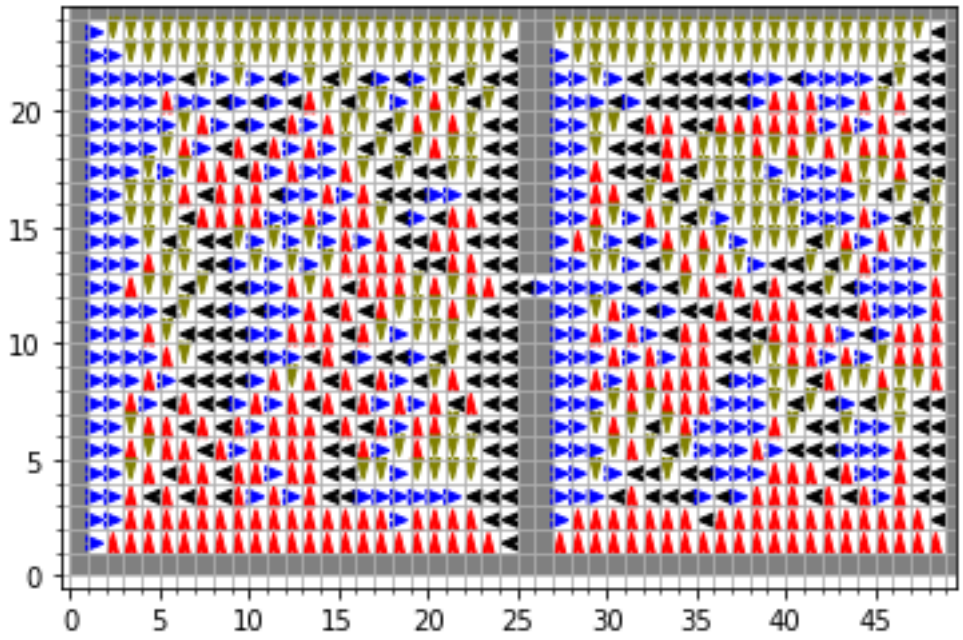


Figure 3: Policy by Value Iteration at $\gamma = 0.1$

The above policy and value function are generated by using $\gamma = 0.1$, as we can see in the value function the information about the goal has not propagated much because γ has a very important role to play in that due to γ being small the updates became small very quickly and the algorithm terminated after just 4 iterations and the goal information was not propagated much and hence the policy is also not very good.

1.2 (b) Effect of Discount Factor

In this part we will increase the γ to 0.99 and see the effects.

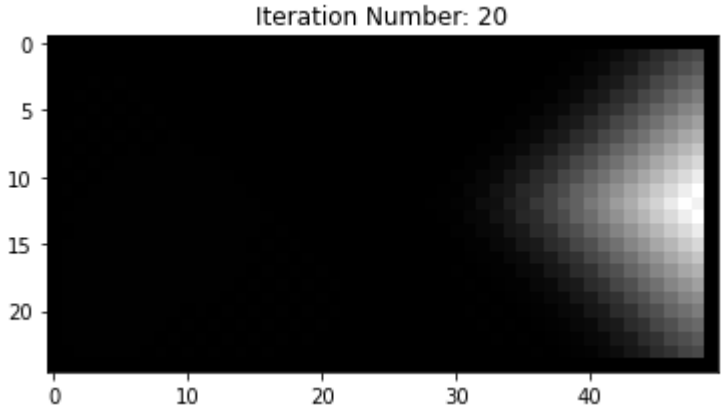


Figure 4: Value Function at Iteration 20 for $\gamma = 0.99$

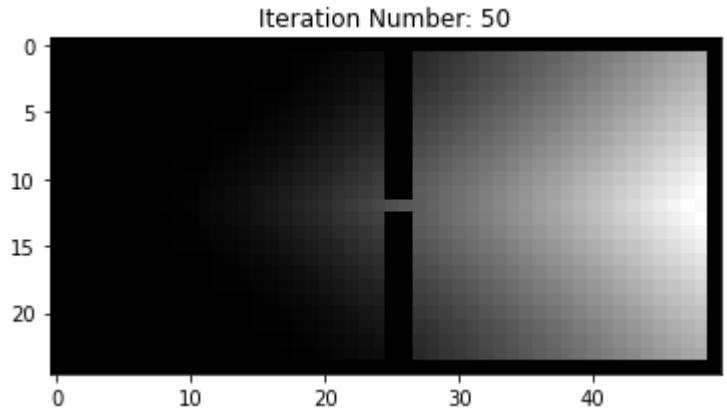


Figure 5: Value Function at Iteration 50 for $\gamma = 0.99$

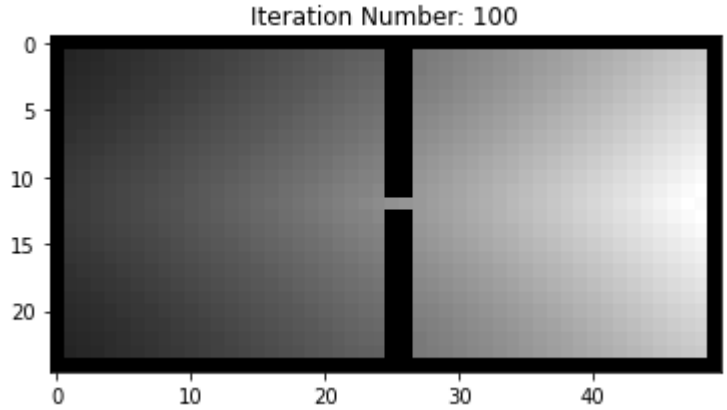


Figure 6: Value Function at Iteration 100 for $\gamma = 0.99$

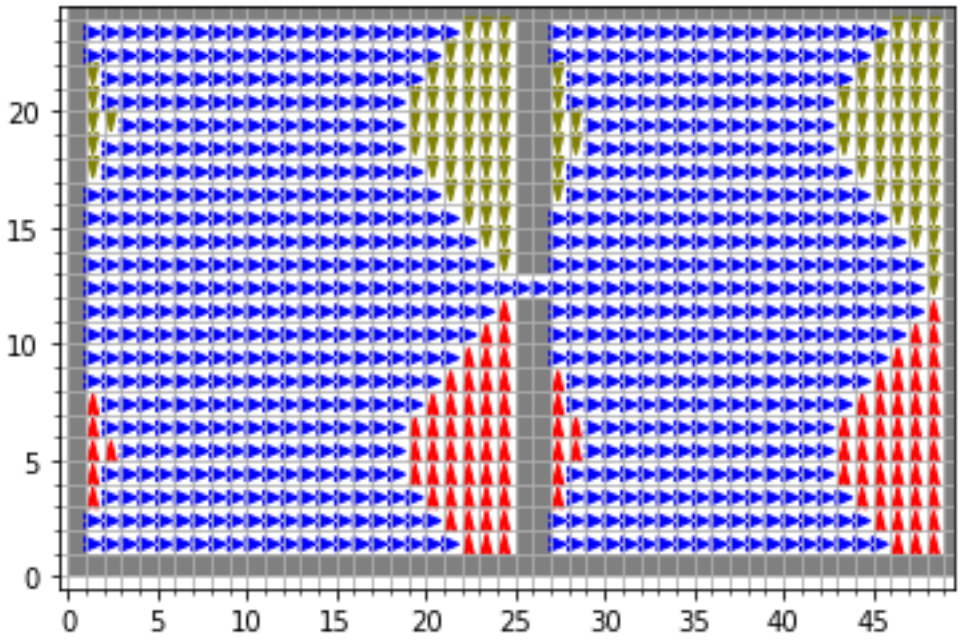


Figure 7: Policy by Value Iteration at $\gamma = 0.99$

As we can see this case by the time algorithm terminates the information about goal has propagated to far-off states in the system and hence the policy also looks much better as compared to part-a.

1.3 (c) Max Norm Study

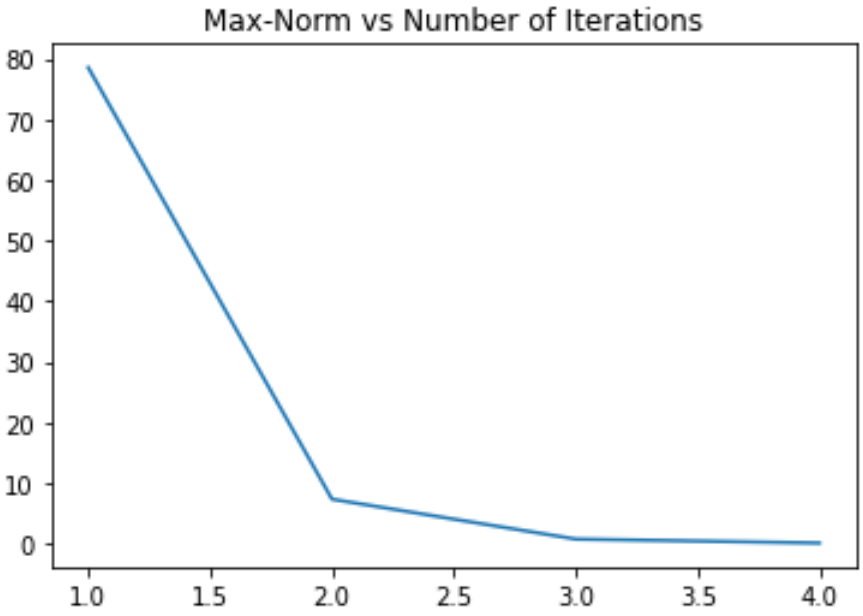
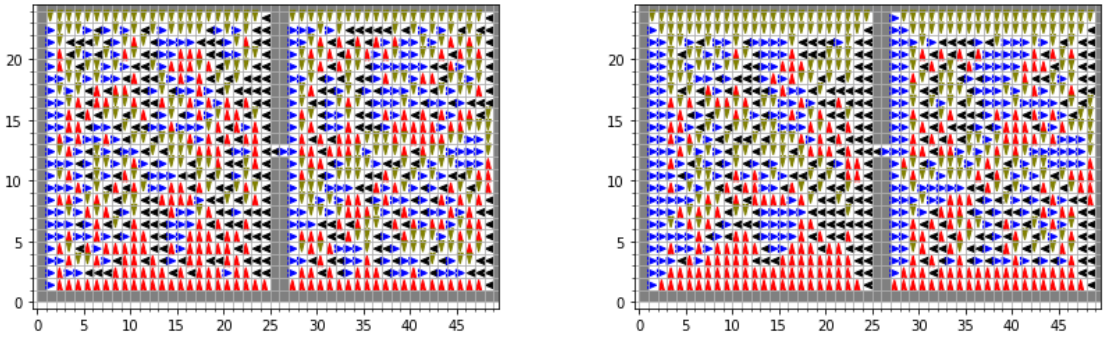


Figure 8: Max-Norm vs Iterations at $\gamma = 0.1$



(a) Policy after 3rd iteration

(b) Policy after 4th iteration

Figure 9: Comparison of Policies at $\gamma = 0.1$

As we can see that for $\gamma = 0.1$ the algorithm terminates at 4th iteration but still there is a significant difference between the policy at the end of 3rd and 4th iteration. So for $\gamma = 0.1$ the policy hasn't converged properly.

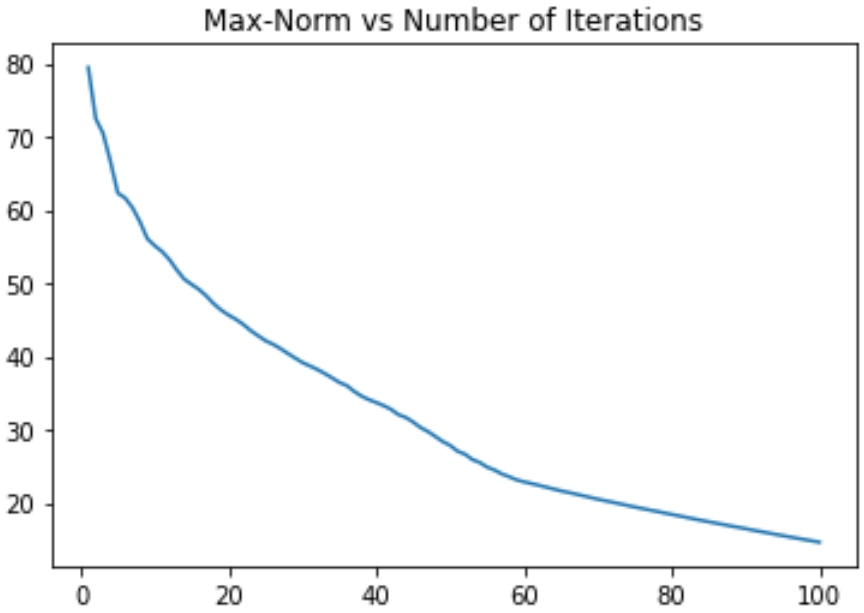
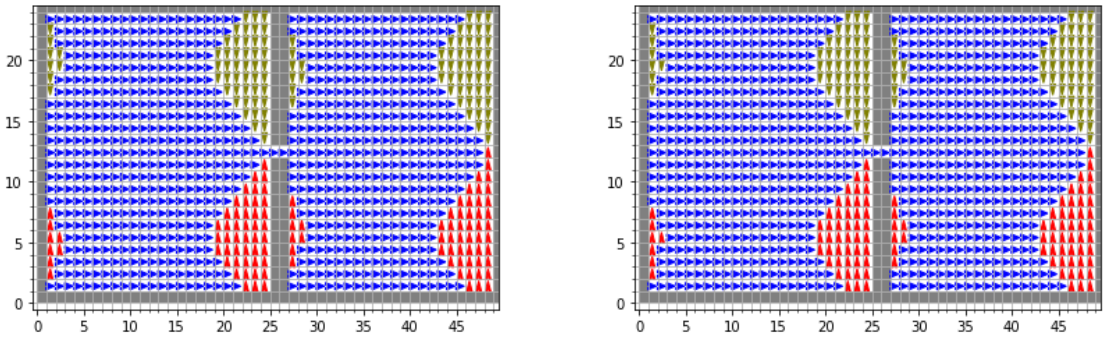


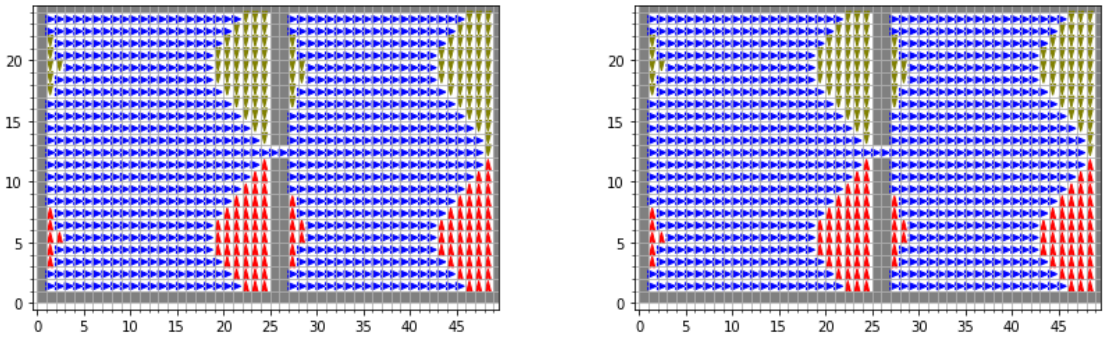
Figure 10: Max-Norm vs Iterations at $\gamma = 0.99$



(a) Policy after 70th iteration

(b) Policy after 80th iteration

Figure 11: Comparison of Policies at $\gamma = 0.99$



(a) Policy after 90th iteration

(b) Policy after 100th iteration

Figure 12: Comparison of Policies at $\gamma = 0.99$

For $\gamma = 0.99$ as we can see policy didn't change from 70th to 100th iteration which we can also see in the Max norm plot as after 70 iterations max norm had decreased to a low value and policy had started converging.

2 Question 2 : Model Free RL

2.1 (a) Q Learning

Initialize $Q(s, a)$ arbitrarily
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

Figure 13: Q-Learning Algorithm

2.2 (b) Visualizing Value function

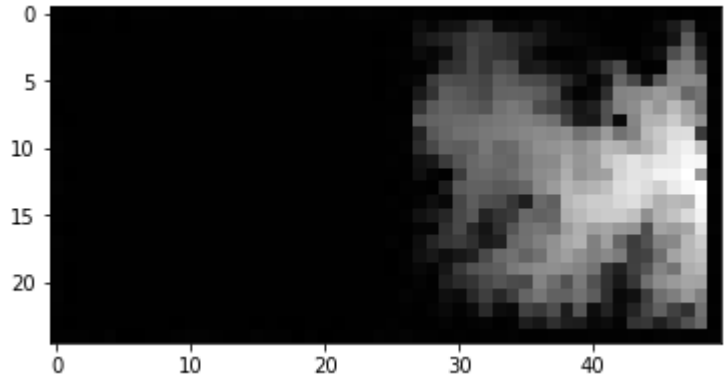


Figure 14: Value Function at $\epsilon = 0.05$

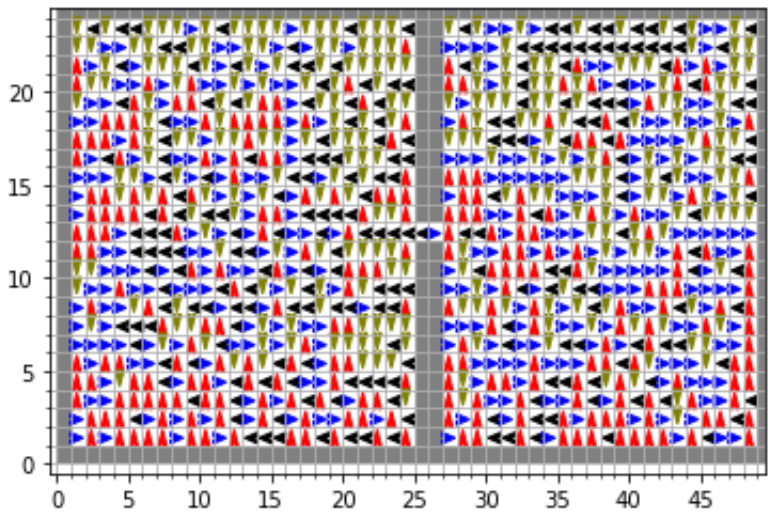


Figure 15: Policy at $\epsilon = 0.05$

2.3 (c) Changing Exploration parameter

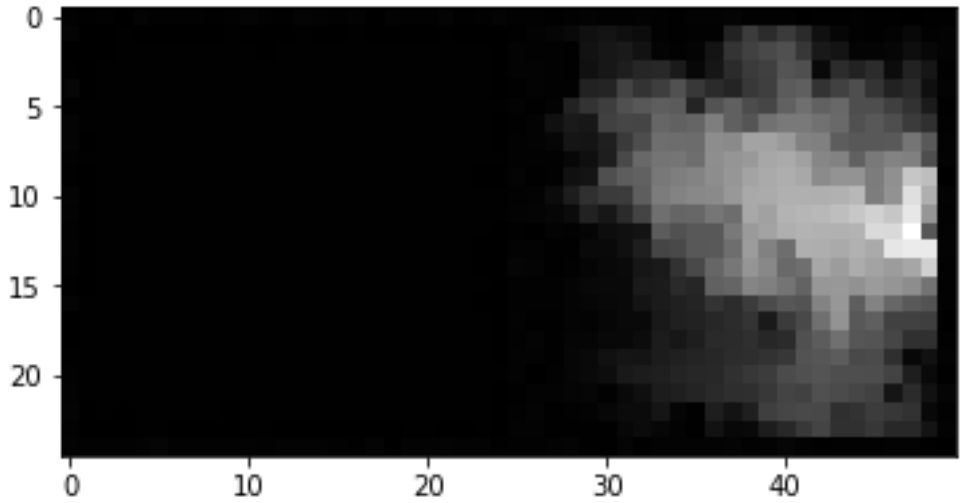


Figure 16: Value Function at $\epsilon = 0.005$

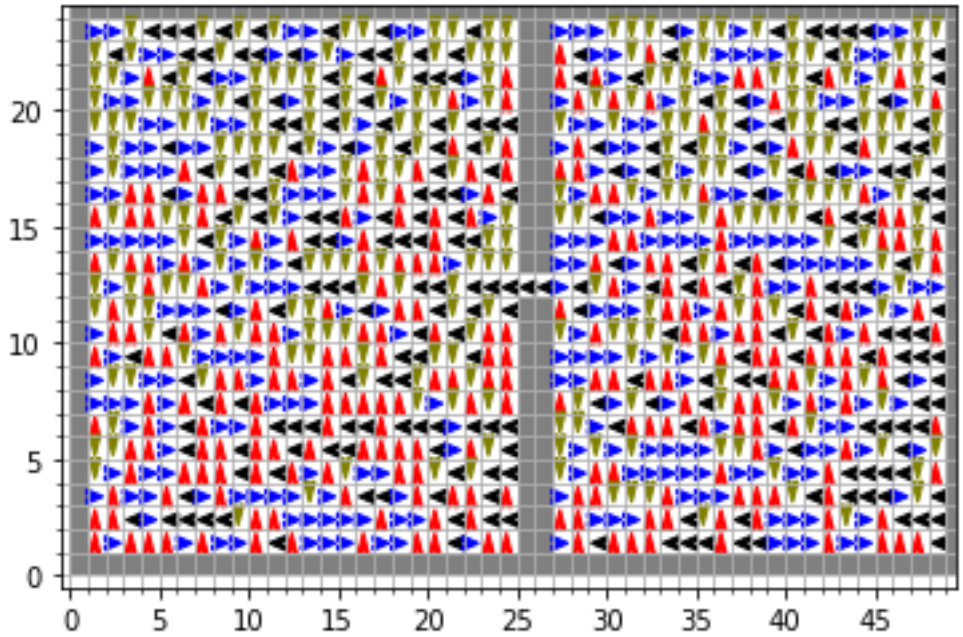


Figure 17: Policy at $\epsilon = 0.005$

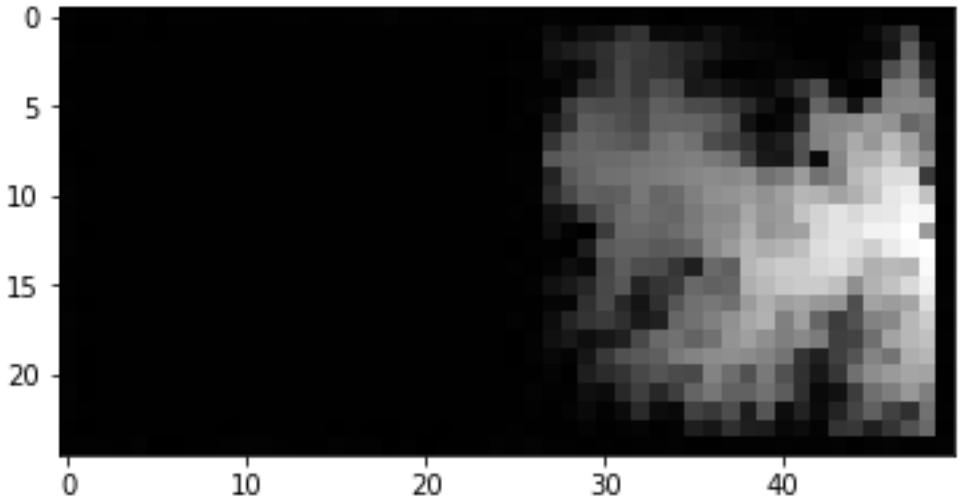


Figure 18: Value Function at $\epsilon = 0.05$

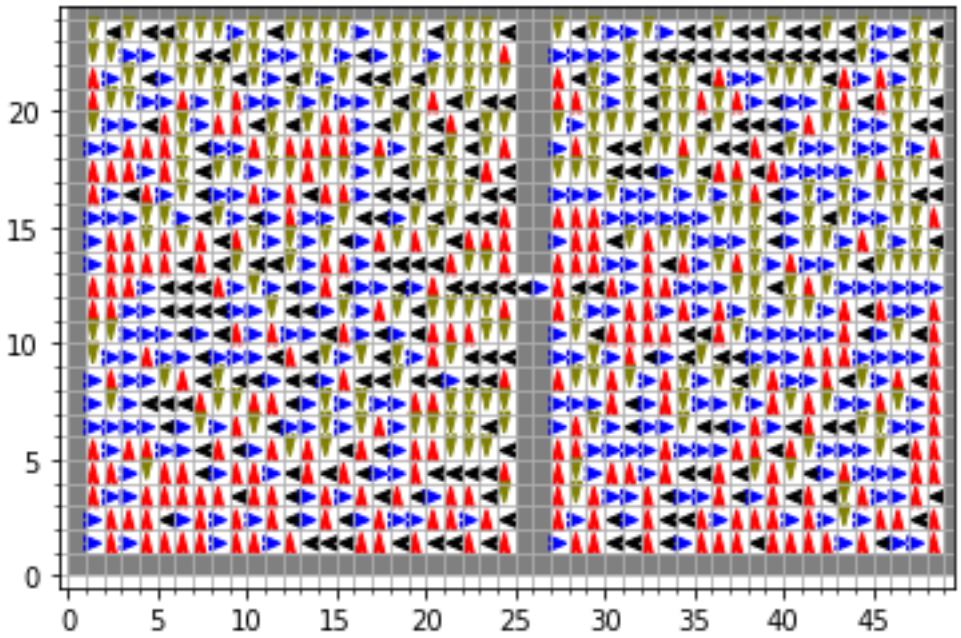


Figure 19: Policy at $\epsilon = 0.05$

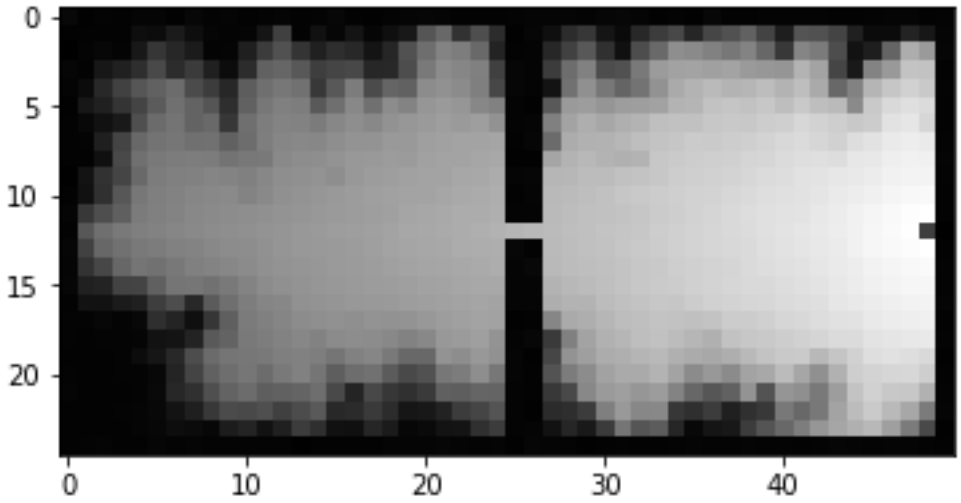


Figure 20: Value Function at $\epsilon = 0.5$

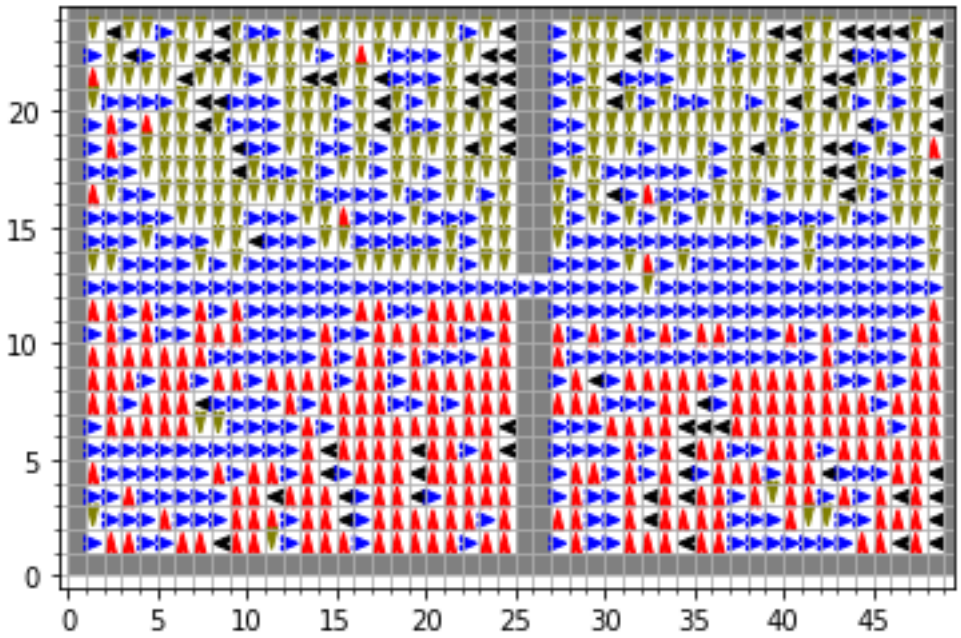


Figure 21: Policy at $\epsilon = 0.5$

As we can see in the plots that as the ϵ increased the information about the goal was able to reach farther states which was not reached at a smaller value and due to reaching of information about goal to farther states the policy also became better as ϵ increased.

2.4 (d) Reward Plot

3 Question 3 : Model-based RL

3.1 (a) Balanced Wandering

- Assuming we are in state s
  - If s is a state we have never seen before, take any action with uniform probability.
  - If have been at s before then take the action that has been tried the fewest times from s (breaking ties randomly).

Figure 22: Balanced Wandering

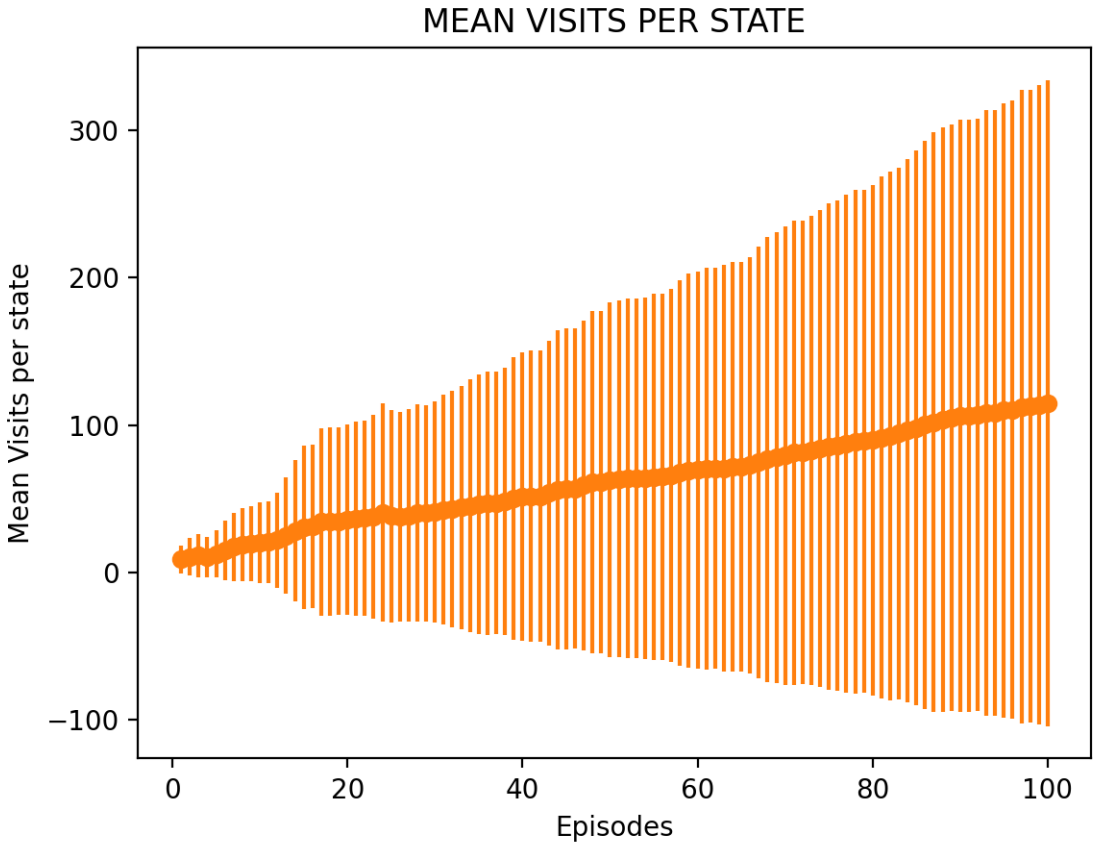


Figure 23: Balanced Wandering

As we can see mean visits per state increase as number of episodes increase along with standard deviation.

3.2 (b) Model Estimation

We estimated transition and reward functions from episodes using the below equations

- Assume some prior policy π.
- Follow π and record state transitions (s_t to s_{t+1}, etc.), and reward $r(s_t, a_t, s_{t+1})$ at each time step.
- Estimate model parameters, e.g.,

$$p(s^k|s^i, a^j) = \frac{n(s^i, a^j, s^k))}{\sum_s n(s^i, a^j, s)} \quad (1)$$

$$R(s^i, a^j, s^k) = \frac{\sum_t r(s^i = s_t, a^j = a_t, s^k = s_{t+1})}{n(s^i, a^j, s^k)} \quad (2)$$

where $n(s^i, s^j, a^k)$ is the number of times in state from s^i to s^k based after taking action a^j.

Figure 24: Certainty Equivalence Principle

3.3 (c) Policy Extraction using Value Iteration

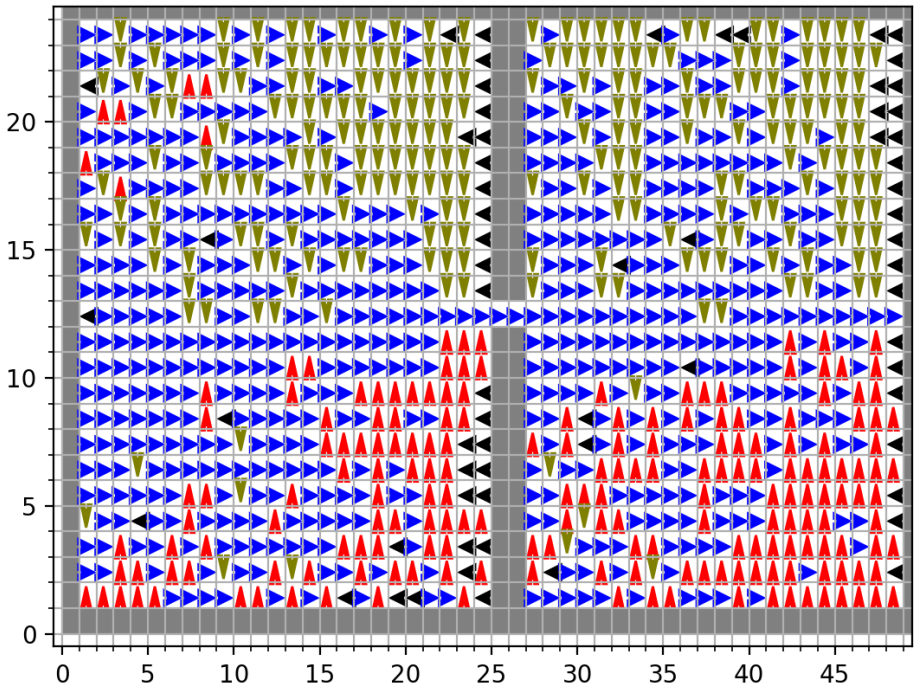


Figure 25: Policy by Model Based RL

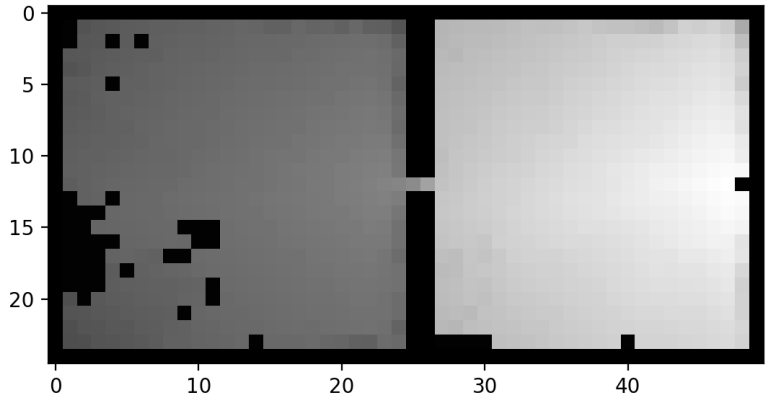
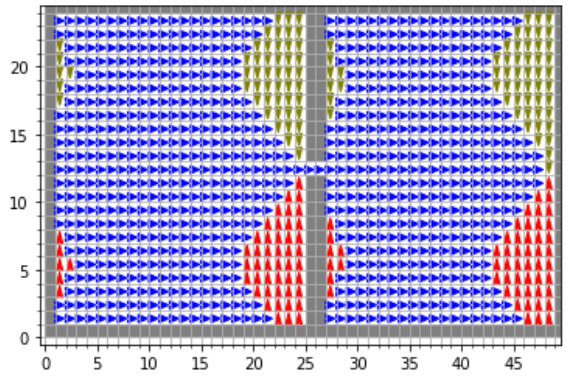
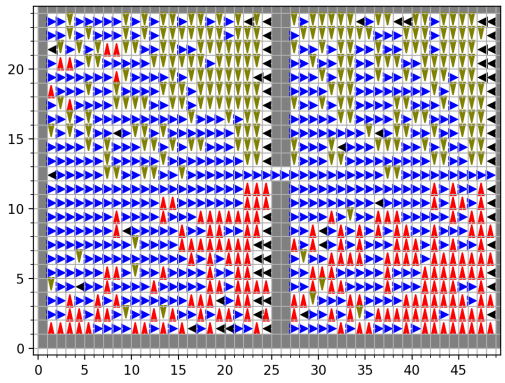


Figure 26: Value Function by Model Based RL

For comparison with policy and value function obtained by solving the exact i.e known MDP



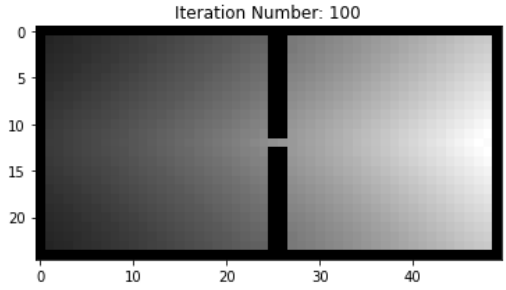
(a) Policy from MDP



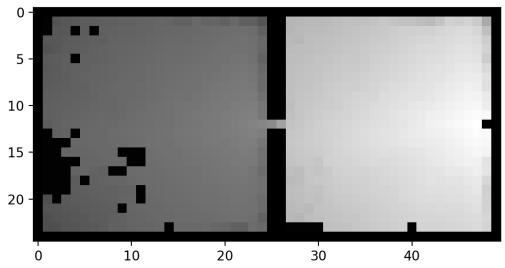
(b) Policy obtained from Model Based RL

Figure 27: Comparison of Policies

As we can see there is randomized behaviour in policy from model based RL mimicking the optimal policy except at walls where it tries to move away due to negative reward being received there due to balanced wandering.



(a) Value function from MDP



(b) Value function obtained from Model Based RL

Figure 28: Comparison of Value Functions

Similarly apart from noise due to randomness ,value function of wall neighbouring states is negative/ darker than normal states due to higher chances of getting negative reward during balanced wandering.Also states to the left have lower value