Simulation of MDP and Q-Learning

Sarthak Behera (2018CS10384)
Pratik Pranav (2018CS10368)

April 2021

Introduction

Autonomous navigation has been a major field of research for AI researchers for many decades. In order to formalise the notion of environment, we use concepts of cost and reward to provide incentives to the mobile robot to act towards the goal. Sometimes we know the transition-model for the robot. This case reduces to solving the grid-world MDP to find the optimal policy through various state value functions (or through state-action value functions). When we don't know the transition model, we resort to reinforcement learning to learn the optimal policy. Here, in this assignment, the first part involves calculating the value function from the MDP while the second part involves using Q-Learning to compute the optimal policy for the grid-world.

1 Optimal Policy through Value Iteration

Value Iteration is a generic algorithm which consists of Value Evaluation and then policy improvement. We keep track of the state-action values (Q-values) for all the states and in each step and in each iteration, we improve our evaluation of the value of a state (expected-reward on the long run considering the discount factor if we follow the optimal policy - denoted by V^*) and then make improvements in the policy through these new evaluations. This is in some sense similar to the *Expectation-Maximisation* (*K-Means* is a special case of it) algorithm. It also has the nice property that it converges which makes it an extremely viable algorithm.

Mathematically,

$$Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)\{R(s,a,s') + \gamma * V_k(s')\}$$

$$V_k(s) = \max_{a \in \mathcal{A}(s)} Q_k(s,a)$$

Here k denotes the iteration number. As $k \to \infty$, $V_k \to V^*$ so by running the algorithm till it satisfies some suitable convergence criteria, we may obtain the optimal policy (or a good-enough approximation) for it. As we cannot do arbitrarily large number of iterations, we look for signs of convergence while doing the iterations. One particular convergence criteria is the *max-norm* criteria, where we take the max difference between values of states in consecutive iterations. Now we set a particular Θ as the parameter and if the *max-norm* is less than it then we say that Value iteration has converged.

A Value Iteration with heavy discount

Heavy discount means that the value of discount factor is small (here it is 0.01). Small discount factor means that the agent is short-sighted about the rewards. Value iteration is basically information flow from the high magnitude reward space to low magnitude reward space. But with low discount factor, this information is heavily attenuated when it flows to the neighbouring states. Thus only the immediate neighbours manage to receive information about the possible high magnitude reward. The grayscale image also shows that only the high reward state (48, 12) and its immediate neighbours have high values and rest of the states have very little clue about this high reward state. Also note that

we converge very quickly (barely within 4 iterations). Figure 1 shows the image obtained

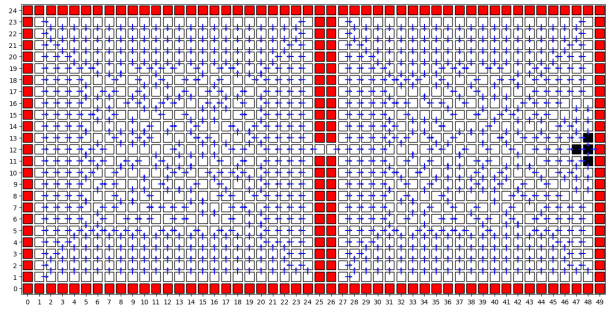

Figure 1: Optimal Policy ($\epsilon = 0.01$)

B High discount factor

Now when we increase the discount factor, the algorithm adopts a far-sighted approach to reward accumulation thus information flow is much faster. In the grayscale image as well we see that the gradient is much smoother. We also notice that we take some time to converge in this case as we have more information that needs to travel to distant states. Figure 2, 3, 4 shows the grayscale image for different iterations as mentioned in question

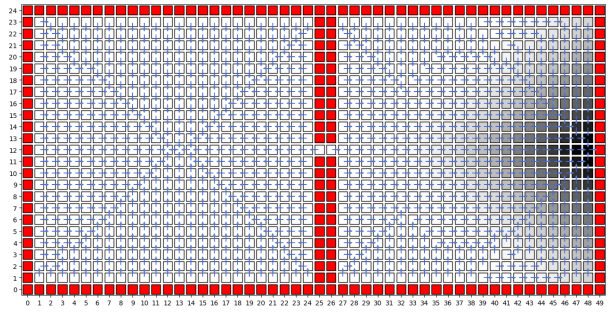

Figure 2: Iterations = 20

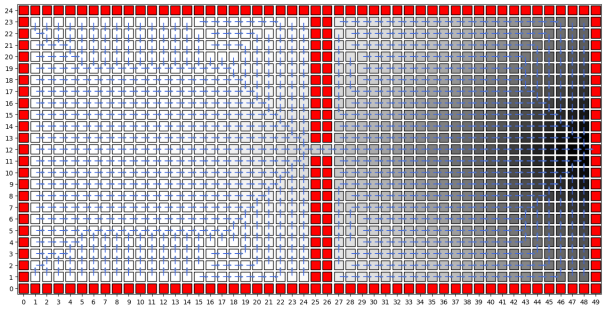

Figure 3: Iterations = 50

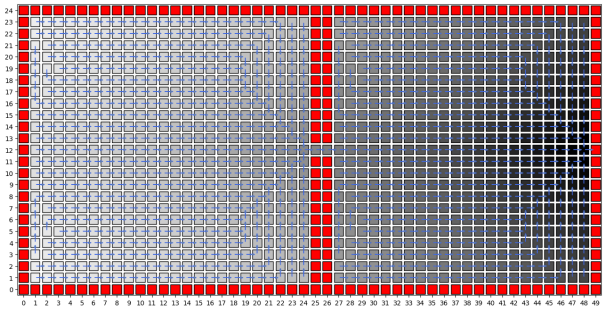

Figure 4: Iterations = 100

C Policy extraction from Value iteration

First, we visualise a sample run on the prescribed policy. We see that due to the probabilistic nature of the

transition model, sometimes we deviate from prescribed policy but ultimately we reach the goal state. Now we

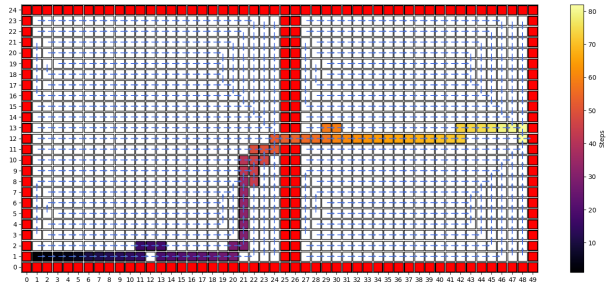

Figure 5: Sample Run

count the number of times state is visited by bounding the episode size. But if we take a large episode size (say 1000) most of the time is spent in the goal state and thus we get a skewed colormap. Given below is the colormap for episode size=1000 and after taking log of the number of times state is visited. To get a view of the

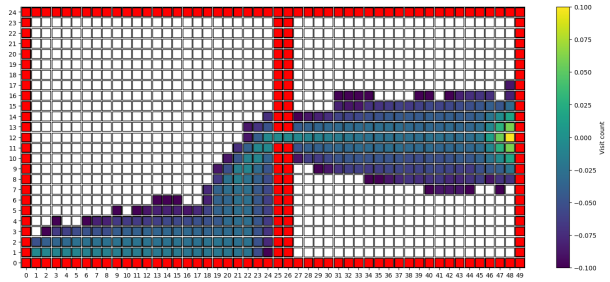

Figure 6: Log visitation for $ep\_sz = 1000$

approximate path followed by the agent after following the prescribed policy for a large number of episodes, we can set the episode size to be 75 and note the state visitations. Given below is the colormap.

D Max-norm vs time (for different discounts)

We see that *max-norm* decreases rapidly when discount factor is low while the *max-norm* has much smoother descent in case discount factor is high.

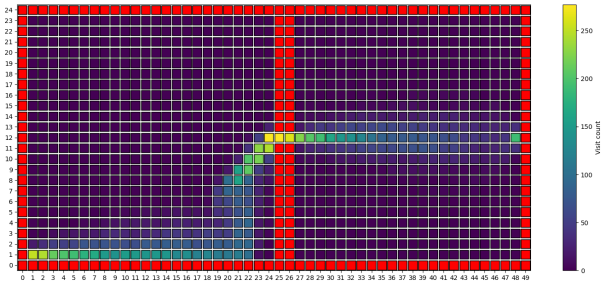

Figure 7: Visitation Count for $eps\_sz = 75$

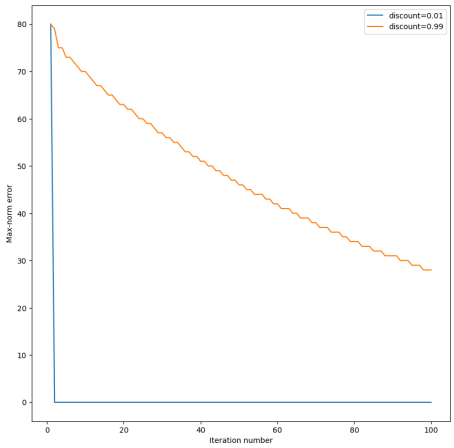

Figure 8: Max Norm vs Iteration Number

2 Optimal Policy thorugh Q-Learning

Q-Learning is model-free so has no information about the various state transition probabilities. It is also off-policy learning as it follows a greedy policy which might not be the same as the current policy. In Q-Learning, we maintain Q-values for all the state-action pairs. The principal equation in Q-Learning is given below:

$$Q(s,a) \leftarrow Q(s,a)+\alpha*\{r+\gamma*\underset{a'}{max}\ Q(s',a')-Q(s,a)\}$$

Here $r = R(s,a,s')$

A Running Q-Learning

Q-Learning takes some time to run. As it is model-free, much of the learning depends on the exploration-exploitation trade-off. We see that increasing the number of episodes has a positive effect on the learning but increases the computational cost.

B Visualising the state value

To obtain the state-values, we take the max among the Q-values for that state, i.e., $V(s) = \underset{a}{max}\ Q(s,a)$. We notice the overall trend similar to that of Value Iteration but the goal state is white as now it is a terminal state as well as we see that shades are not always smooth but has somewhat discontinuous gradients. This is to be expected as it is dependent on how much exploration the agent has done the RL setting.

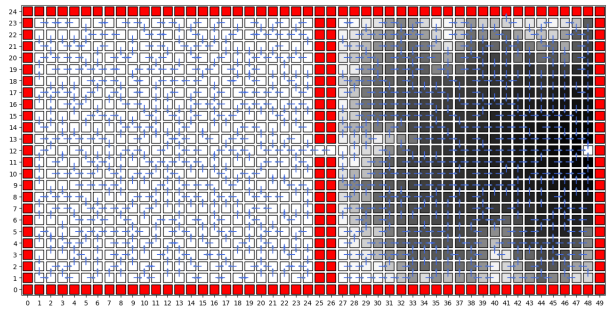

Figure 9: Q-Learning

C Varying the exploration parameter

Increasing the exploration parameter leads to smoother gradients and information spreads to distant states. We can consider the phenomenon of diffusion as quite similar to the running of the algorithm. The reward from the goal state is sort of diffusing to the other room via the hole. Higher exploration parameter makes it more probable to sort of diffuse more into the adjoining room.

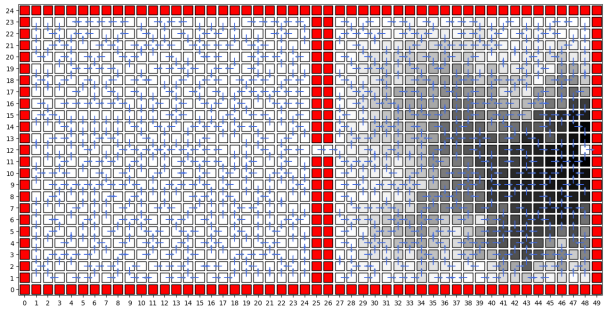

Figure 10: $\epsilon = 0.005$

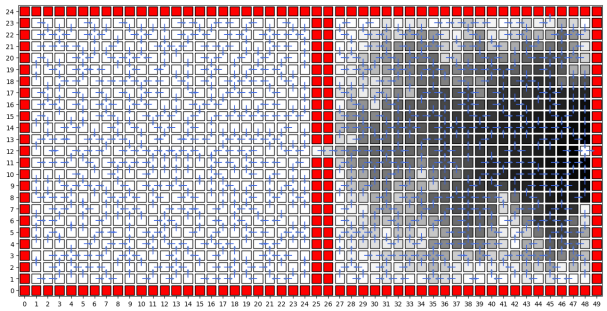

Figure 11: $\epsilon = 0.05$

D Average reward accumulated (varying ϵ)

We note that for higher exploration values, we learn more initially and the reward values are higher compared to lower exploration values. As the number of episodes increases, average rewards increase as well but lower exploration values performs asymptotically better

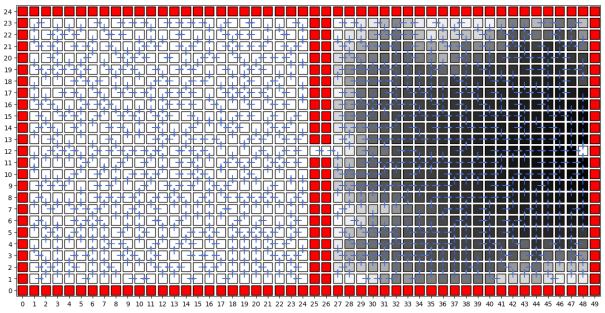

Figure 12: $\epsilon = 0.5$

than the higher exploration methods. This is somewhat expected. Thus ideally we prefer initially having a higher exploration rate which we gradually decrease as the number of episodes increases and this dynamic strategy yields higher rewards. This is also true if we consider our life as well. Initially we explore different kinds of subjects ranging from Science, Language, Social Sciences, etc. but as we grow we focus our attention to a particular field (for us Computer Science) and further narrow our field to do various specializations. But here we see that $\epsilon = 0.05$ sort of performs better initially as well. This is due to the fact that there is only one maxima in the grid thus once the maxima is known, the lesser ϵ exploits this given maxima. The perks of the higher ϵ are only seen when we have multiple optima and lower ϵ are more prone to be stuck to a local optimum. Another thing to observe is that we are doing random starts which is also another randomising element that helps the lower ϵ.

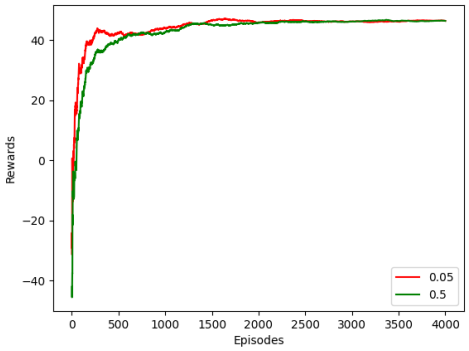

Figure 13: Rewards Accumulated per episode