CS 7641 — Assignment 4: Reinforcement Learning

Niranjan Thakurdesai

December 4, 2018

In this assignment, three different algorithms, viz. value iteration, policy iteration and Q-learning were evaluated on two different Markov Decision Process (MDP) problems.

1 MDP Problems

Two MDPs were explored, one with a small number of states and the other with a large number of states. Both are 2D grid worlds where an agent has to navigate towards one or more goals. The first grid world has size 5×5 with 25 states while the second one is 25×25 with 625 states. The worlds have walls and bumping into them results in a large penalty of -50. The living reward is -1, i.e. the reward for going to a state other than a wall or a goal is -1. The reward for reaching the goal is 0. The set of permissible actions is up, down, left and right. If the agent bumps into a wall, it stays in the same state. The actions are noisy, i.e. if the agent wants to take a particular action, it will take some other action with a fixed probability. Thus, the agent has to reach the goal without bumping into walls to maximize the reward collected. The two worlds are shown in figure 1. The black lines represent walls and the orange squares represent goals. The worlds have tunnels and traps where the agent can get stuck. The second world has two goals and it would be interesting to observe the policies learnt at different states and their relation to their relative positions with respect to the goals.

These problems are interesting because they represent simplified versions of real-world environments in which a robot would be expected to navigate towards one or more targets. The two different sizes would help us understand the challenges we can expect to encounter as we scale the problem.

2 Overview of algorithms

The three algorithms used to solve the MDPs are described briefly below.

2.1 Value iteration

This is an algorithm that computes the optimal policy by iteratively updating the values (expected utilities) at each state based on the action that maximizes it computed from the transition probabilities, expected utilities and rewards of the neighboring states. The initial values can be initialized arbitrarily. It can be proven that it converges to the optimal values.

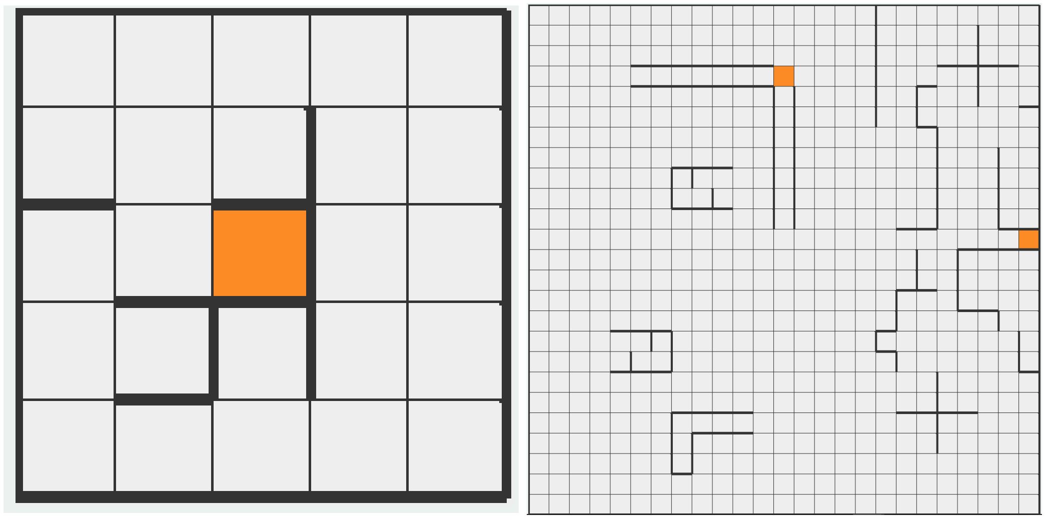

(a) Small (b) Large

Figure 1: Chosen MDPs

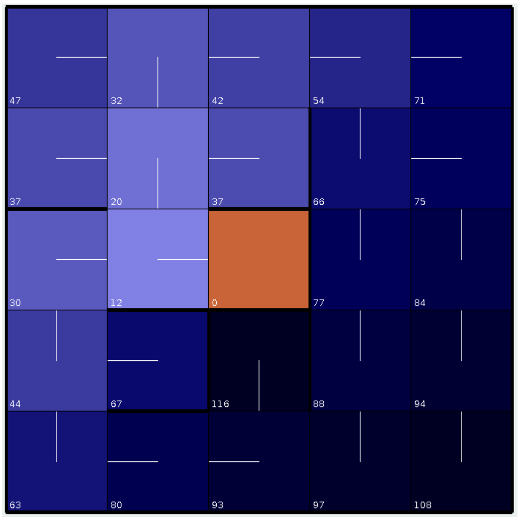
2.2 Policy iteration

Value iteration keeps improving the the value function until it converges. However, the optimal policy converges long below the value function. Since the agent only cares about the optimal policy, this is inefficient. Policy iteration works around this problem by iteratively improving the policy at each step and computing the value function according to the new policy instead of repeatedly improving the value function. Policy iteration is also guaranteed to converge to the optimal policy. It often takes fewer iterations to converge than value iteration.

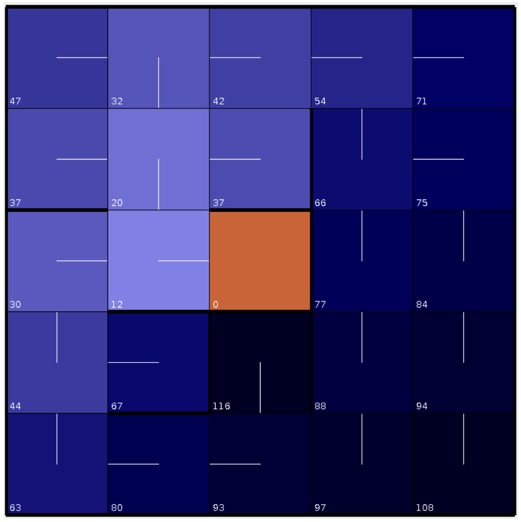
2.3 Q-learning

This is a reinforcement algorithm in which we assume that the transition probabilities and rewards are not known beforehand unlike the previous two algorithms. The agent only knows what are the set of possible states and actions, and can observe the environment current state. In this case, the agent has to actively learn through the experience of interactions with the environment. The Q values are iteratively updated based on the next state that maximizes them.

An important question is how the agent selects actions during learning. It can either take the action which maximizes the Q value or take other actions which may lead to better rewards. This is known as the exploration-vs-exploitation dilemma. In this assignment, we use the ϵ-greedy policy where at each step the agent takes a random action with some small probability ϵ instead of taking every action greedily. This encourages exploration. ϵ can be decreased over time as the agent becomes more confident with its estimate of Q values. This algorithm converges to the optimal policy under certain conditions on learning hyperparameters.



(a) Value iteration



(b) Policy iteration

Figure 2: Policies and penalties for the small MDP

3 Experiment and analysis

The Reinforcement Learning Simulator[1] developed by Rohit Kelkar and Vivek Mehta at Robotics Institute, Carnegie Mellon University was used for experiments. For all three algorithms, the noise probability was set to 0.3 and the threshold for convergence was set to 0.001. For policy iteration, the number of policy evaluations in each iteration was set to 500. For reinforcement learning, different exploration strategies were used by changing the values of ϵ. The learning rate was set to 0.7 initially and decayed according to the formula $1000/(1000 + \text{number of episodes})$. The higher learning rate in the beginning encourages exploration and the lower learning rate towards the end encourages exploitation. The algorithm was run for 10,000 iterations in case of the small MDP and 250,000 iterations for the larger MDP.

For value iteration and policy iteration, the policies and the penalties (negative of the values) are visualized in figures 2 and 3. The shades of blue indicate the penalties. Darker shades mean larger penalties. The number of iterations and the time taken to converge are shown in table 1.

Table 1: Performance of value iteration and policy iteration

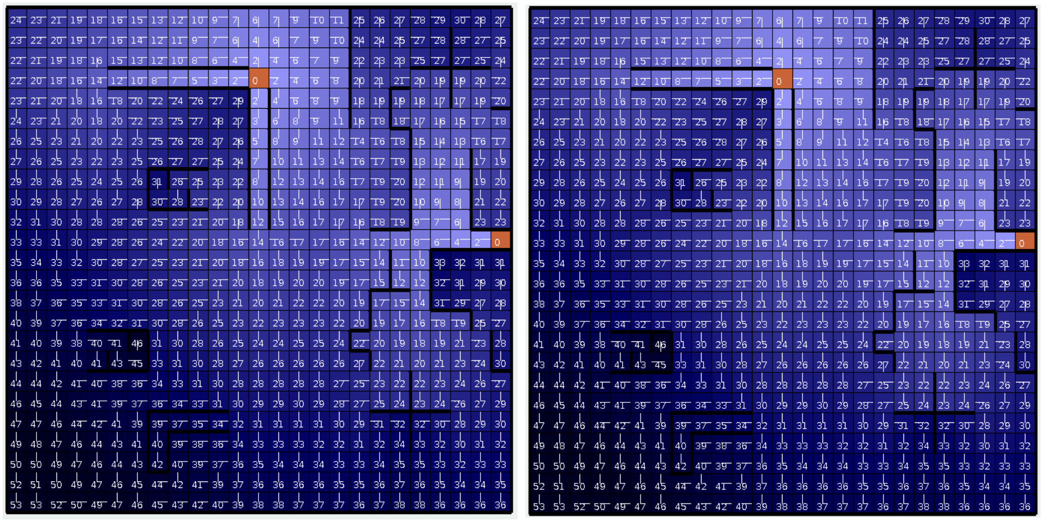
(a) Small MDP

	VI	**PI**
Iterations	50	6
Time (ms)	13	6

(b) Large MDP

	VI	**PI**
Iterations	91	8
Time (ms)	340	866

[1] `http://www.cs.cmu.edu/~awm/rlsim/`



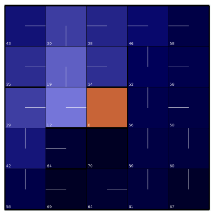
(a) Value iteration (b) Policy iteration

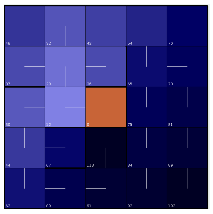
Figure 3: Policies and penalties for the large MDP

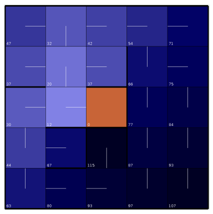
We can observe that both value iteration and policy iteration found the same policy in both problems. This is expected as both value iteration and policy iteration provably find the optimal policy. For the small MDP, policy iteration is faster than value iteration in terms of both time and the number of iterations needed to converge. However, for the large MDP, policy iteration needs fewer iterations than value iteration but runs for a longer time. To understand how problem size affects running time of the two algorithms, we look at their computational complexity. The efficiencies of value iteration and policy iteration are $O(S^2A)$ and $O(S^2N)$ respectively. Here, S is the number of states in the problem, A is the number of possible actions at each state and N is the number of policy evaluations in each policy iteration. Even though policy iteration takes fewer iterations than value iteration, each iteration is more expensive. This especially becomes a problem for large state spaces since even a small increase in the number of iterations taken by policy iteration will result in a large increase in time required to converge. As both of them are optimal, we can conclude that value iteration should be used for small MDPs and policy iteration for large MDPs.

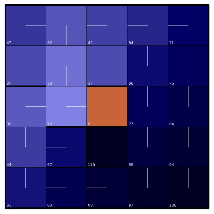
The policies and values at different iterations for the two algorithms are shown in figures 4, 5, 6 and 7. We can observe how they slowly improve and converge to their optima. In the case of value iteration, the policy converges before the values. For the small MDP, the policy converges as early as the 20th iteration. This shows that value iteration is inefficient for small MDPs.

For Q-learning, the policies and Q values for different values of ϵ are shown in figures 8 and 9. For the small MDP, we can observe that it hasn't converged to the optimal policy in any case. For the first four cases, it gets 3–6 states wrong. The best performance is obtained with $\epsilon = 0.5$, the highest ϵ tried, where it gets only one state wrong. This shows the importance of exploration in Q-learning. If the exploration is too low, Q-learning may not converge to the optimal policy. Even for the large MDP, Q-learning could not find the optimal policy for any ϵ. All exploration strategies resulted in some portions of the grid being mapped out correctly. However, states near walls,


(a) Iteration 10


(b) Iteration 20


(c) Iteration 30


(d) Iteration 40

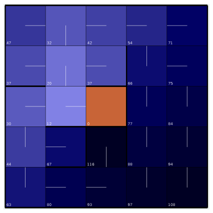
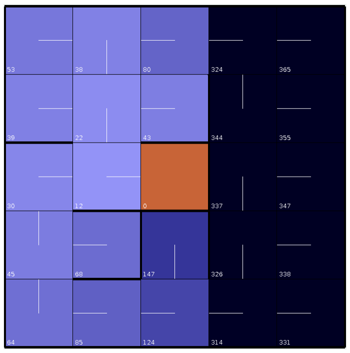
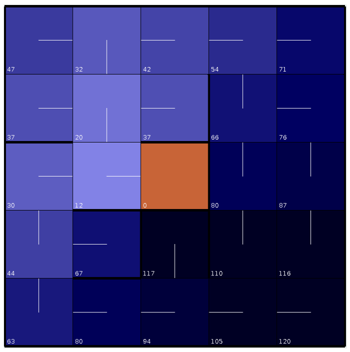

(e) Iteration 50 (convergence)

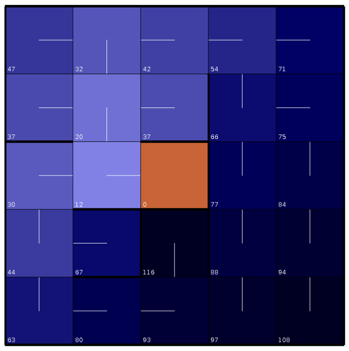
Figure 4: Value iteration: policies and values after different iterations for small MDP



(a) Iteration 2

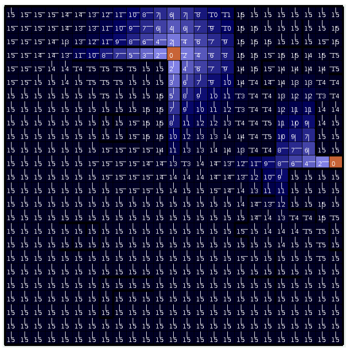


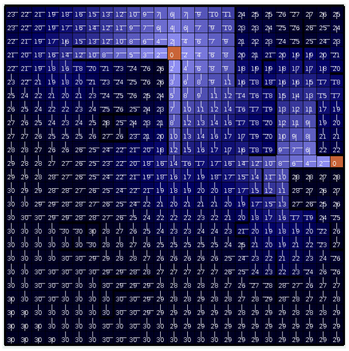
(b) Iteration 4

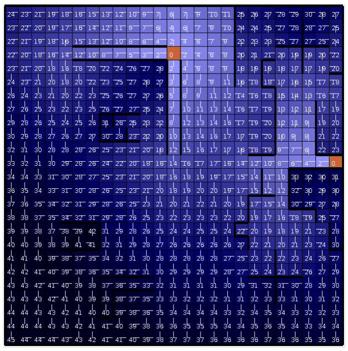


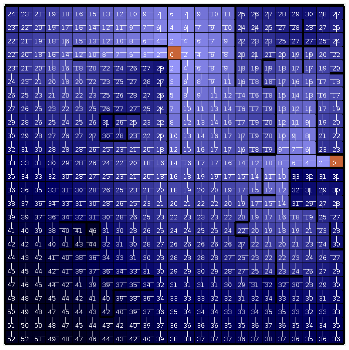
(c) Iteration 6 (convergence)

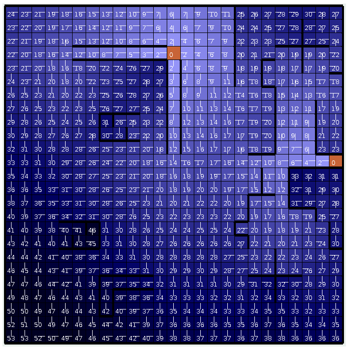
Figure 5: Policy iteration: policies and values after different iterations for small MDP


(a) Iteration 15


(b) Iteration 30


(c) Iteration 45


(d) Iteration 60


(e) Iteration 75

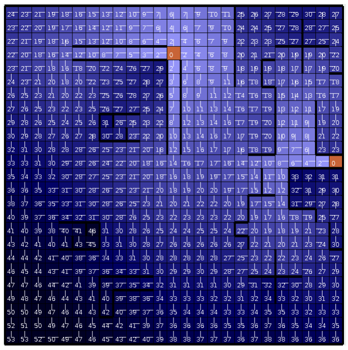
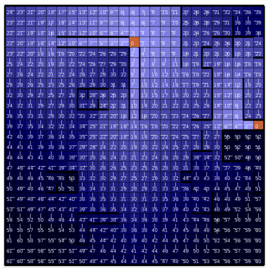
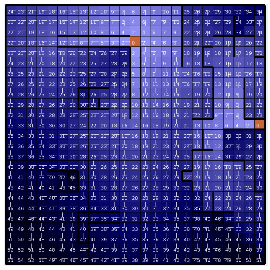
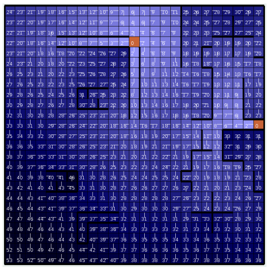

(f) Iteration 91 (convergence)

Figure 6: Value iteration: policies and values after different iterations for large MDP


(a) Iteration 2


(b) Iteration 4


(c) Iteration 6

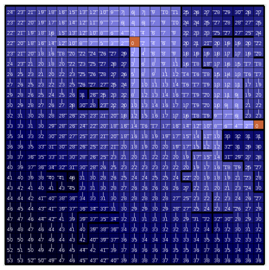
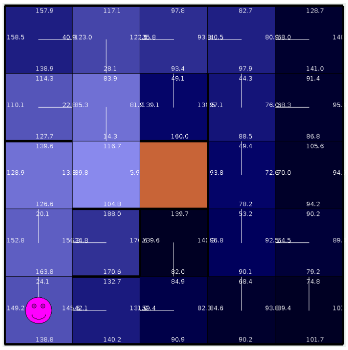
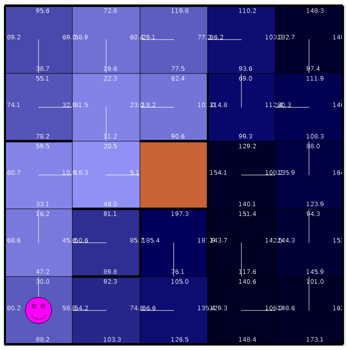
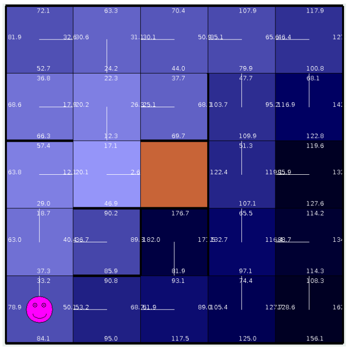
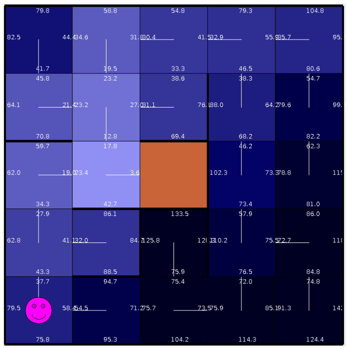

(d) Iteration 8 (convergence)

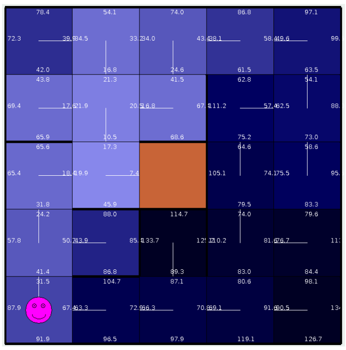
Figure 7: Policy iteration: policies and values after different iterations for large MDP


(a) $\epsilon = 0$


(b) $\epsilon = 0.1$


(c) $\epsilon = 0.2$


(d) $\epsilon = 0.3$


(e) $\epsilon = 0.4$

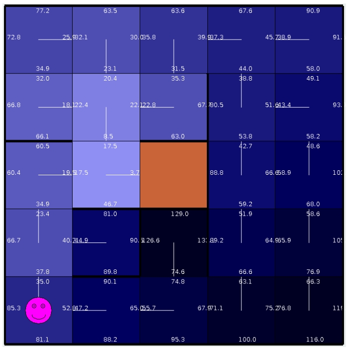
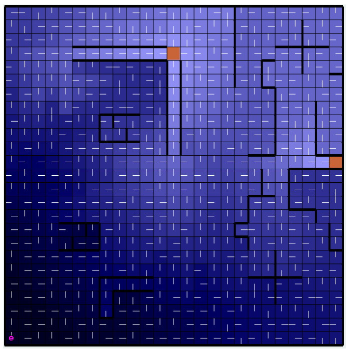
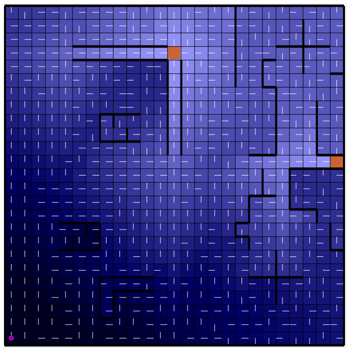

(f) $\epsilon = 0.5$

Figure 8: Q-learning: policies and Q values for small MDP

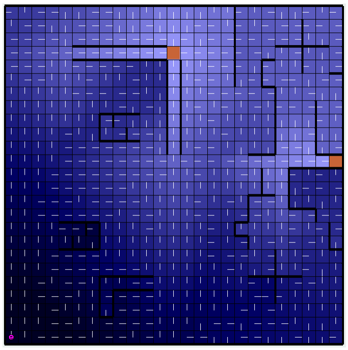
tunnels and traps noticeably have incorrect policies. For large values of ϵ, the policies look better overall, again emphasizing the importance of exploration. Even though Q-learning did not result in optimal policies, it is nevertheless remarkable to observe how it gets most of the decisions right inspite of not having knowledge of the environment and without modeling transition probabilities and rewards.



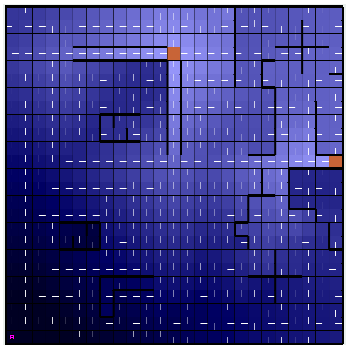
(a) $\epsilon = 0$



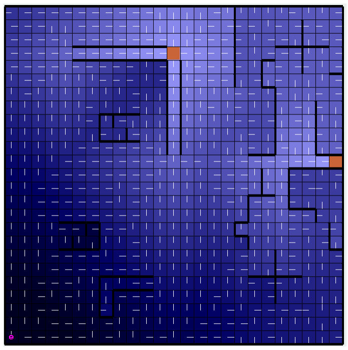
(b) $\epsilon = 0.1$



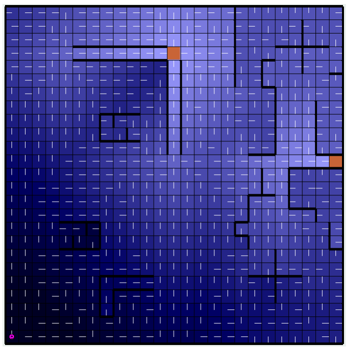
(c) $\epsilon = 0.2$



(d) $\epsilon = 0.3$



(e) $\epsilon = 0.4$



(f) $\epsilon = 0.5$

Figure 9: Q-learning: policies for large MDP