DEPARTMENT OF COMPUTER SCIENCE, GEORGIA INSTITUTE OF TECHNOLOGY

Markov Decision Processes

Joseph Su

January 6, 2019

1 INTRODUCTION

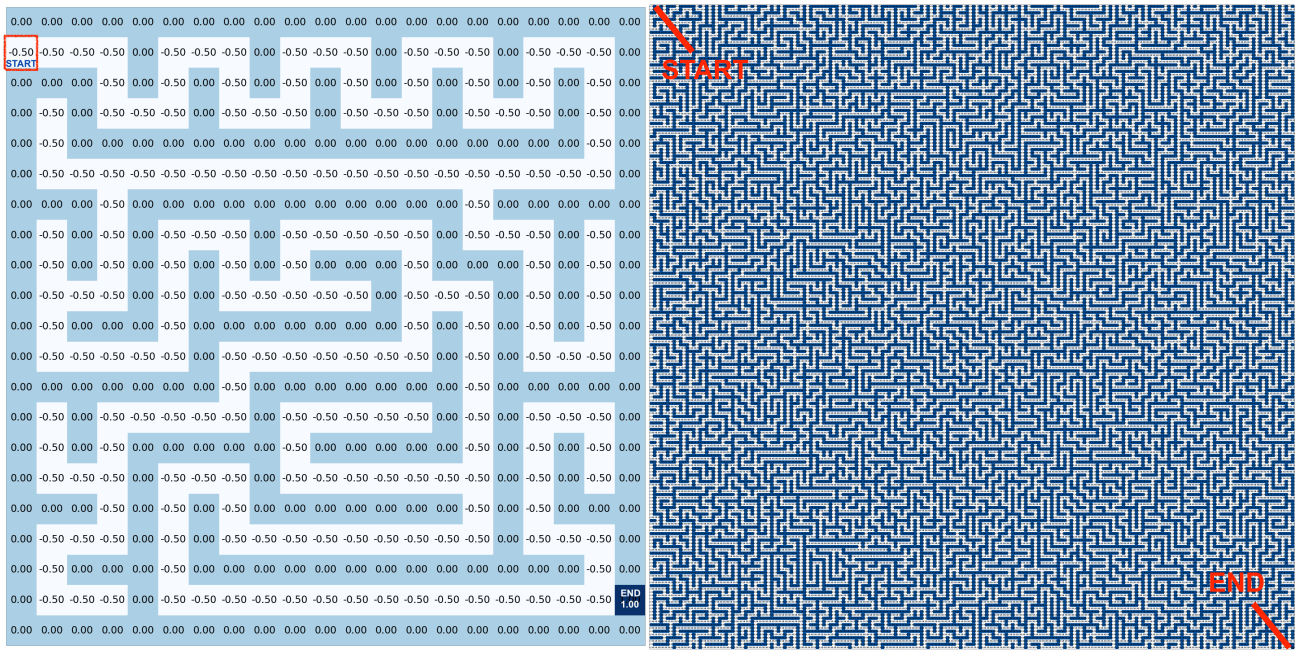
This paper considers two Markov decision process (MDP) problems and employs several variations of Bellman update algorithms to solve them. Specifically we compare and contrast, from time, space, and utility estimation perspectives, the efficacy of value iteration (VI), policy iteration (PI), and a model-free, non-greedy Q-learning that uses VI as Q update to solution convergence on these problems.

COMPUTING FACILITIES This work makes use of various libraries, toolsets, and modules in Python 3.4 to arrive at its studies and results: `numpy`, `scipy`, `IPython`, and `pandas`. Our code is based on the open source Python libraries in [2]. All of the experiments are conducted on a 2015 Mac OSX 2.5 GHz Intel Core i7 with 16GB DDR3.

2 MDP PROBLEMS

We study two MDP problems and an agent living in these worlds with the ability to make non-deterministic decisions over a finite horizon. The agent is restricted to move either up, down, left, or right with an 80% chance heading in the intended direction, and a 20% chance at right angles to the intended direction: 10%

Figure 1.1: Initial Conditions of a 5x5 Grid World



(a) 20x20 Maze (b) 200x200 Maze

Figure 2.1: Initial Conditions of Mazes

of which to the right and 10% to the left. The initial reward for non-terminal states is -0.05. All obstacles are marked with a reward of 0.00.

FIRST MDP This MDP is a canonical example consisting of an agent living in a grid world. We modify this widely studied MDP to give a 5x5 grid, with a starting location of (0, 0) and two terminals, one +1.00 shown in red and -1.00 shown in blue in Fig. 1.1.

SECOND MDP This MDP consists of a stochastic maze of 20x20 in size. We briefly expand the analysis to a 200x200 maze for validation purpose. Fig. 2.1 displays both mazes with a starting location on the top left, terminating at the lower right with a reward of +1.00.

2.1 MARKOV DECISION PROCESS

An MDP is a discrete-time stochastic process involving taking an action $a \in A$ at any given state, $s \in S$, with a transition probability $T(s,a,s')$ of moving from s to the next state, s'. An immediate reward, $r(s,a,s')$ is observed. We assume a Markovian property that the effect of an action taken in s depends only on that state, not on its prior history. Namely $T(s,a,s') = T(s,a)$. Also, the ordering preference of states are independent of when they occur, making rewards cumulative. We measure the performance, or utility U, of an MDP agent by summing up its rewards over the visited states. Lastly, rewards become less important in the distant future; we model such characteristic by discounting them with a discount factor that is $0 < \gamma < 1$:

$$U(s_0, s_1, s_2, \cdots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \cdots \tag{2.1}$$

A discount is equivalent to an interest rate of $(1/\gamma) - 1$.

POLICY A policy, π, is a choice of determining which action to take at each state. The objective policy, π^*, of an MDP is to maximize expected discounted reward over a finite discrete horizon, H, starting with $t = 0$:

$$\pi^* : S \times \{0, 1, \cdots, H\} \rightarrow A = \arg\max_{\pi} E[\sum_{t=0}^{H} r_t(s_t, a_t, s_{t+1}) | \pi] \tag{2.2}$$

2.2 VALUE ITERATION

Value iteration (VI) is based on the Bellman equation for computing ϵ-optimal policy, for a discounted problem spanning infinite horizon with the following form, $\gamma \in (0,1]$:

$$U_{i+1}(s) \leftarrow \max_{a \in A} \{r(s,a) + \gamma \sum_{s' \in S} T(s,a,s') U_i(s')\} \tag{2.3}$$

Optimal policy is thus computed

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a) U(s') \tag{2.4}$$

Each VI iteration cuts its current value to optimality by a factor of γ. This reduction is along a fixed point, getting closer to it in the limit. As γ approaches 1 the rate of convergence decreases until a bound is reached. This bound is defined by, δ, the maximum change in the utility of any state in an iteration and its inequality $\delta < \epsilon(1-\gamma)/\gamma$. The algorithm is as follows:

Algorithm 1 Value Iteration

1: **procedure**
2: Initialize U
3: **REPEAT**:
4: $U \leftarrow U'$
5: **FOR** each state s:
6: $U'(s) = r(s) + \gamma \max_a \sum_{s'} T(s'|s, \pi(s)) U(s')$
7: **END**
8: **UNTIL** convergence at $\delta < \epsilon(1-\gamma)/\gamma$

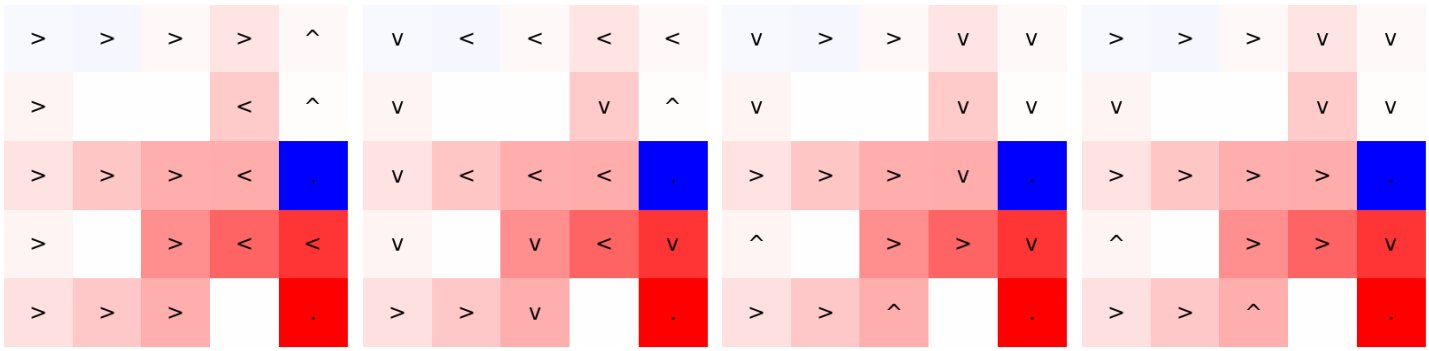
The Bellman update is applied to all the states at each iteration. Information propagates through the state space until all the states have the optimal utility estimates giving us the corresponding optimal policy.

2.3 POLICY ITERATION

PI alternates between policy evaluation and improvement. The former chooses a policy, π, and calculates its next policy for each state by way of $U_i \leftarrow U^\pi$, whereas the latter derives a new maximum expected utility π_{i+1} with an one-step lookahead based on U_i. PI terminates when policy improvement yields no change in the utility. These steps are summarized as follows:

Algorithm 2 Policy Iteration

1: **procedure**
2: Initialize π
3: **REPEAT**:
4: $\pi \leftarrow \pi'$
5: $U \leftarrow U'$
6: $U'(s) = r(s) + \gamma \sum_{s'} T(s'|s, \pi(s)) U(s')$
7: **FOR** each state s:
8: $\pi'(s) \leftarrow \arg\max_a \sum_{s'} T(s'|s,a) U(s')$
9: **END**
10: **UNTIL** $\pi = \pi'$



(a) $r > 0.522$ (b) $0.11 < r < 0.131$ (c) $-0.59 < r < -0.35$ (d) $r < -0.6996$

Figure 3.1: Optimal Policy with Varying Rewards, r

-0.03	-0.04	0.03	0.10	0.03
0.04	0.00	0.00	0.20	0.01
0.12	0.22	0.32	0.33	-1.00
0.05	0.00	0.44	0.61	0.80
0.12	0.22	0.32	0.00	1.00

(a) VI: 18 Iterations

-0.03	-0.04	0.03	0.10	0.03
0.04	0.00	0.00	0.20	0.01
0.12	0.22	0.32	0.33	-1.00
0.05	0.00	0.44	0.61	0.80
0.12	0.22	0.32	0.00	1.00

(b) PI: 4 Iterations

-0.19	-0.19	-0.04	0.04	-0.03
0.03	0.00	0.00	0.11	-0.08
0.10	0.20	0.30	0.23	-1.00
0.03	0.00	0.44	0.60	0.80
0.11	0.21	0.31	0.00	1.00

(c) Q Learning: 100,000 Iterations

Figure 3.2: Utility Estimates in a 5x5 Grid World

2.4 Q-Learning

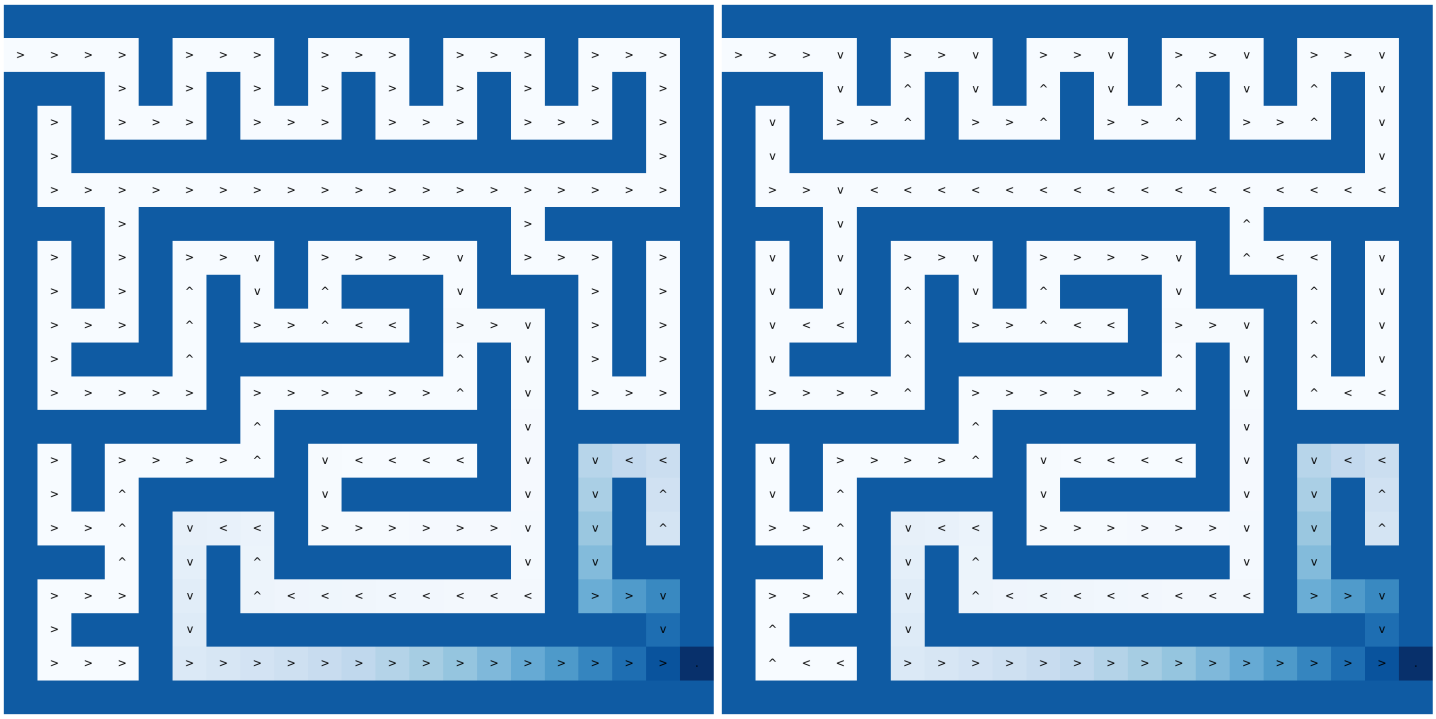
Q-learning is a model-free technique in reinforcement learning that focuses on finding optimal policy, $\pi^* : S \rightarrow A$, in any MDP. Specifically, Q-learning is an alternative Temporal Difference (TD) method [5] in that a TD agent does not require $T(s'|s,a)$ in learning or selecting actions. The technique relates to utility by way of $U(s) = \max_a Q(s,a)$, with the following TD constraint:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{2.5}$$

where α is the learning rate and γ is as defined for PI and VI. The above suggests that a TD agent that uses Q is an active learner that learns $Q(s,a)$ of each action-state pair, requiring no policy at all.

3 Results

5x5 Grid World One observes that there are more than a dozen reward profiles for this problem. For brevity we review 4 profiles representative of a swath of possible agent preferences. Fig. 3.1 illustrates 4 varying reward values, r, with respect to optimality; their heat map is consistent with Fig. 3.2. When reward is high ($r > 0.522$) life is absolutely beautiful as seen in Fig. 3.1a. The agent enjoys staying in the grid so much it is in no hurry to exit the +1.00 terminal. The agent avoids the -1.00 pit and even loiters around the adjacent regions in circles. Why go far when big rewards are right on the spot? When life is decent with small rewards, $0.11 < r < 0.131$, the agent avoids the -1.00 pit by moving away from it for as far and long as possible; Fig. 3.1b exhibits this behavior. When the agent is stuck right above +1.00 just beneath -1.00, it chooses to exit with +1.00; the reward is there but not lucrative enough for the agent to consider taking a stroll through town. The agent continues



(a) VI: Optimal Policy

(b) PI: Optimal Policy

Figure 3.3: Optimal Policy via VI and PI for a 20x20 Maze

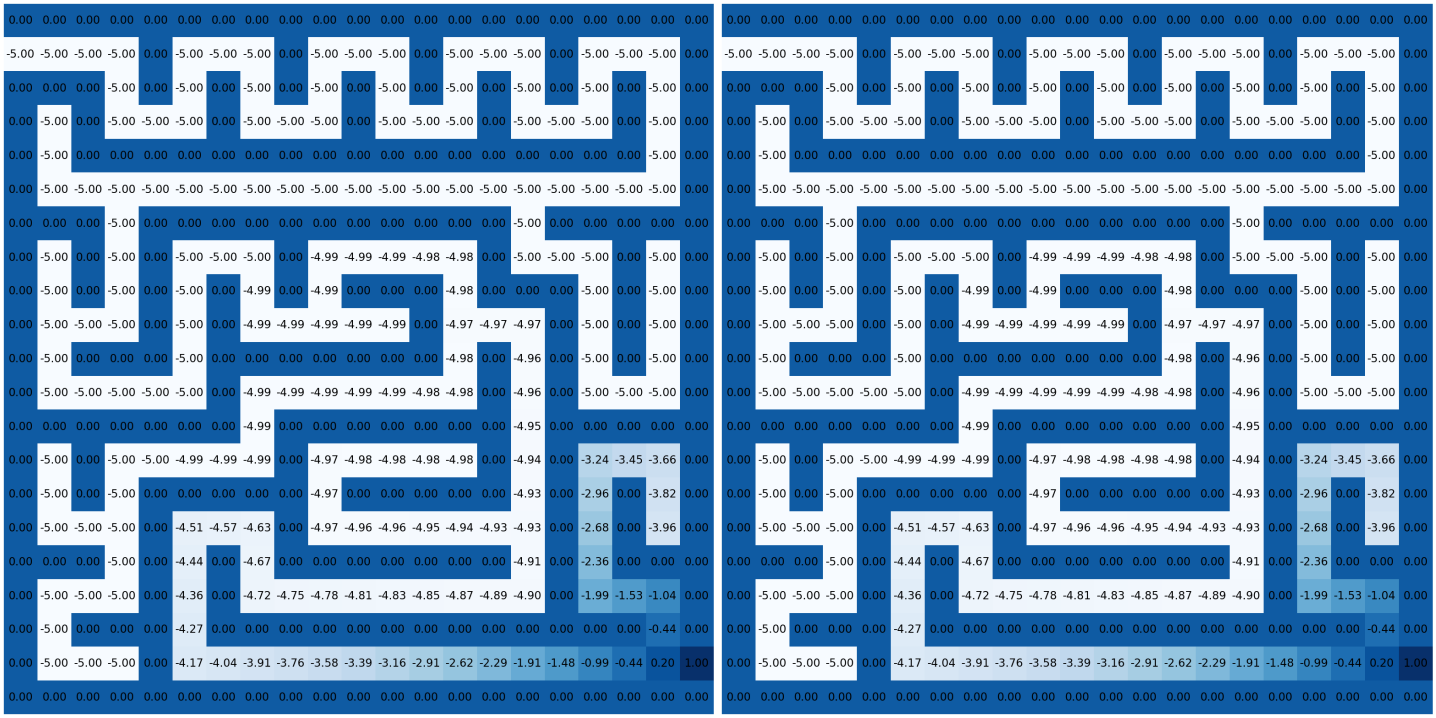
to exploit rewards in all other non-terminal states. Fig. 3.1c showcases a world that is sprinkled with small risks all over town; it is so painfully risky that when the agent is just above the -1.00 terminal it makes for the best exit. All other paths lead to the best possible choice: +1.00. Lastly Fig. 3.1d shows an inhabitable world with $r < -0.6996$ in all non-terminal regions; the agent heads straight to the nearest exit regardless it is -1.00 or +1.00.

Fig. 3.2 exhibits the utility estimates, or action-state values, after VI, PI, and Q-learning with Bellman updates. Both VI and PI yield identical results albeit the latter requires less than 25% of the total iterations in VI. Q-learning converges very slowly and is only "close enough" after 100,000 iterations. The spots farther away from the exits are far from being optimal.

20X20 AND 200X200 MAZES Both VI and PI algorithms produce slightly different optimal policies as shown in Fig. 3.3. PI yields the correct and optimal policy without any backtracking and wall collisions. VI disagrees with PI on spots farther away from the exit, showing the agent's tendency to walk into walls, collecting -0.05 in-place rather than furthering along. This phenomenon can be described as being in an adversarial environment where conditions are so harsh that the only sensible move for the agent is to hit the wall repeatedly, especially when exit is nowhere in sight. Fig. 3.4 displays the optimal utility estimates from VI and PI; both exhibit exactly the same value propagation. PI converges in less than half the VI iterations.

For convergence criteria we obtain both max error and policy loss on a 20x20 maze. Policy loss is defined as an infinity norm (L_∞-norm) of the two utility vectors, $||U^\pi - U||$; the loss is the most an agent can lose by following any suboptimal utility U^π in an iteration. We measure loss as the means to assess how well such approach falls within $||U^\pi - U|| < 2\epsilon\gamma/(1-\gamma)$. Maximum error is the infinity norm of any pre-update and post-update utilities across all states. Fig. 3.5 displays the max error vs policy loss on both 20x20 and 200x200 mazes.

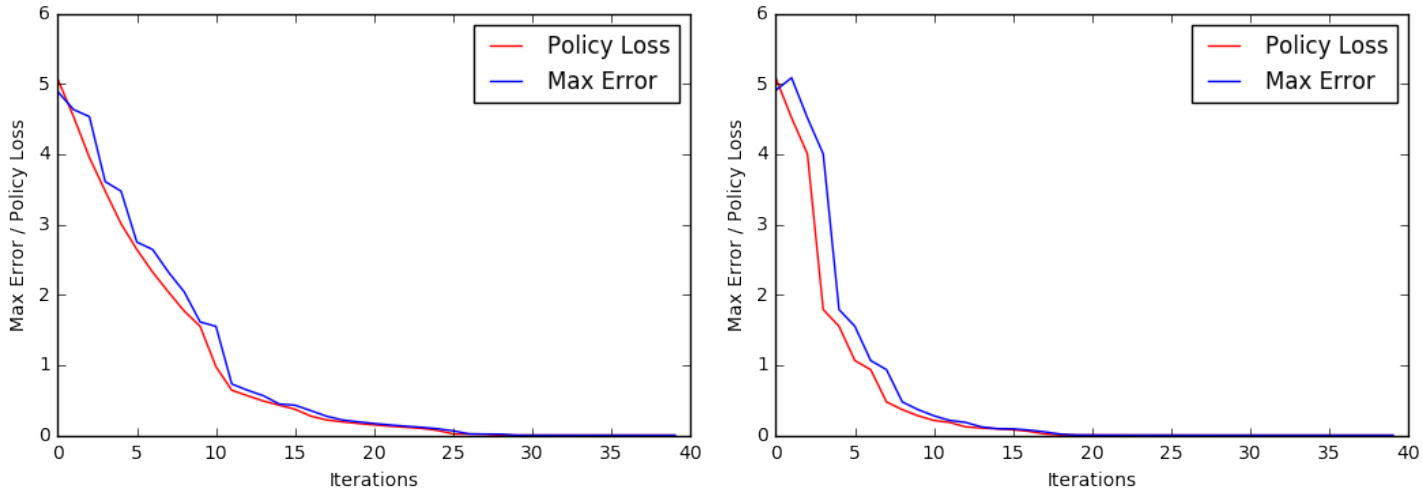
From a runtime and iteration to optimality perspectives, the following are assessed on both or either of 20x20 and 200x200 mazes. Fig. 3.6 displays the number of iterations required to guarantee an error of at most ϵ as a function of γ, the discount factor. Fig. 3.7 displays an optimal policy in state-action values on a 200x200 maze; either VI and PI requires 78 iterations to reach optimality. Fig. 3.8 exhibits the wall clock runtime (ms) for each value and policy iteration. Fig. 3.9 shows the near-optimal policy with Q-learning after



(a) VI: 81 iterations

(b) PI: 38 iterations

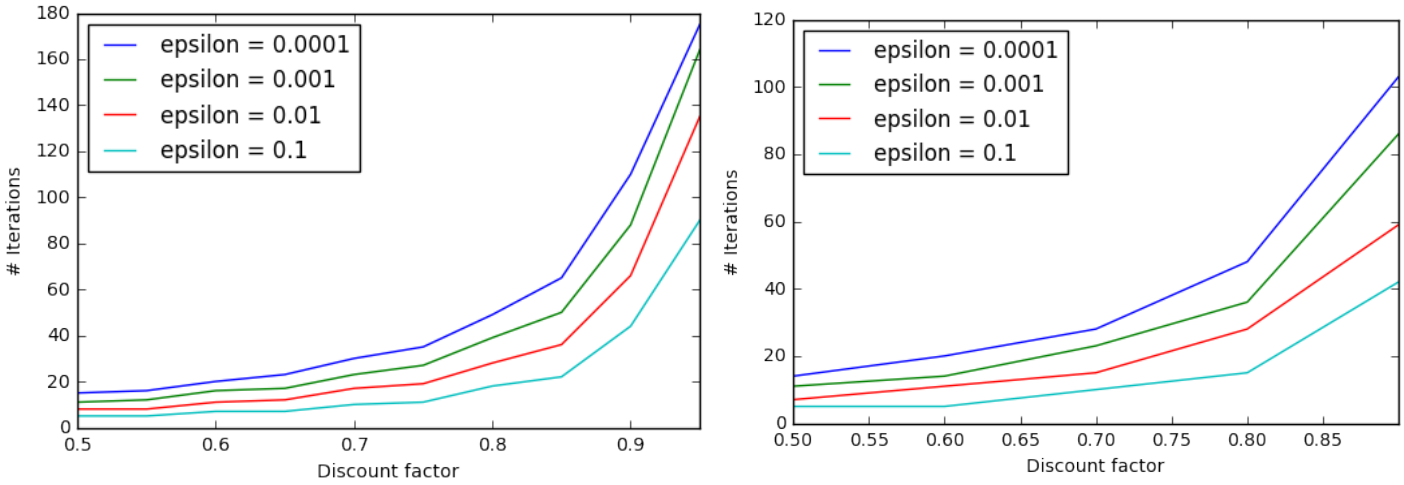
Figure 3.4: Utility Estimates for a 20x20 Maze



(a) 20x20 Maze

(b) 200x200 Maze

Figure 3.5: Runtime (ms) vs Iterations



(a) 20x20 Maze

(b) 200x200 Maze

Figure 3.6: Required Value Iterations vs. γ and ϵ

100,000 iterations.

4 Discussions

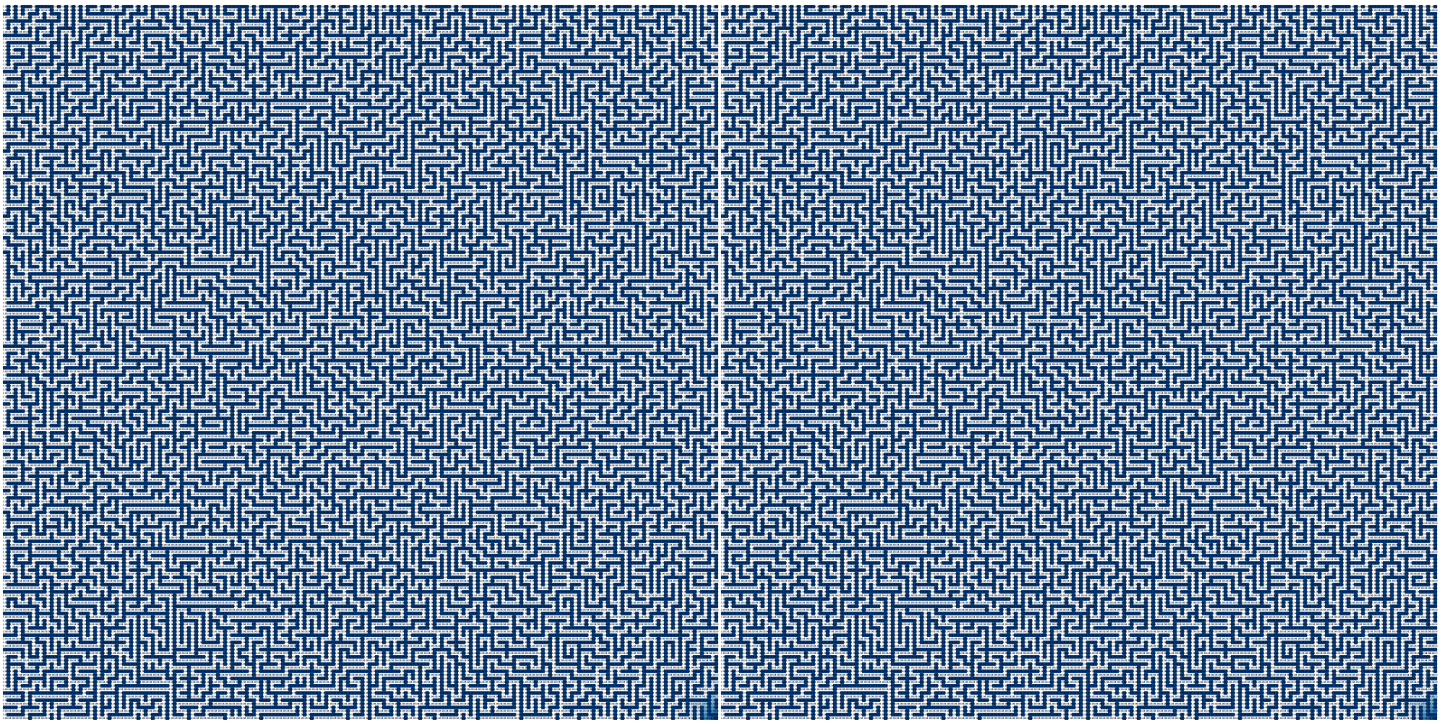
Runtime and Space Given $|A|$ actions and $|S|$ states, VI requires $O(|A||S|^2)$ in time, whereas PI requires $O(|S|^3 + |A||S|^2)$ for each iteration. Fig. 3.8 shows that PI is a much lengthier process than VI per iteration, agnostic of the maze size.

Fig. 3.6 shows that the number of VI iterations grows exponentially in the direction of discount factor, γ. The number of PI iterations is unknown but is almost always lower than VI. We observe that even though both PI and VI result in 78 iterations in Fig. 3.7, the amount of time PI takes is 6-fold more than VI, as seen in Fig. 3.8b. It is well known that for both PI and VI convergence is guaranteed [1, 3]. It is noteworthy that utility U is a fixed point of the Bellman update therefore a solution to the Bellman equations. Hence $\pi \leftarrow U$ from an one-step lookahead must be an optimal policy for that state. In a finite horizon there is a finite number of policies. This means as iteration ensues we are left with fewer "better" policies to choose from. Eventually there is only one policy left at equilibrium, connoting convergence.

Exploration vs Exploitation We observe tradeoffs between exploration and exploitation solving the MDPs. Fig. 3.1 illustrates the balance between risks and rewards in a grid world. The same characteristic is observed in our maze example. Depending on the value of r, an agent chooses to either maximize its reward by exploiting its immediate surrounding or to explore in the hope of finding a better life. There are times when a mixture of strategies must be employed depending on the representation of states in an environment.

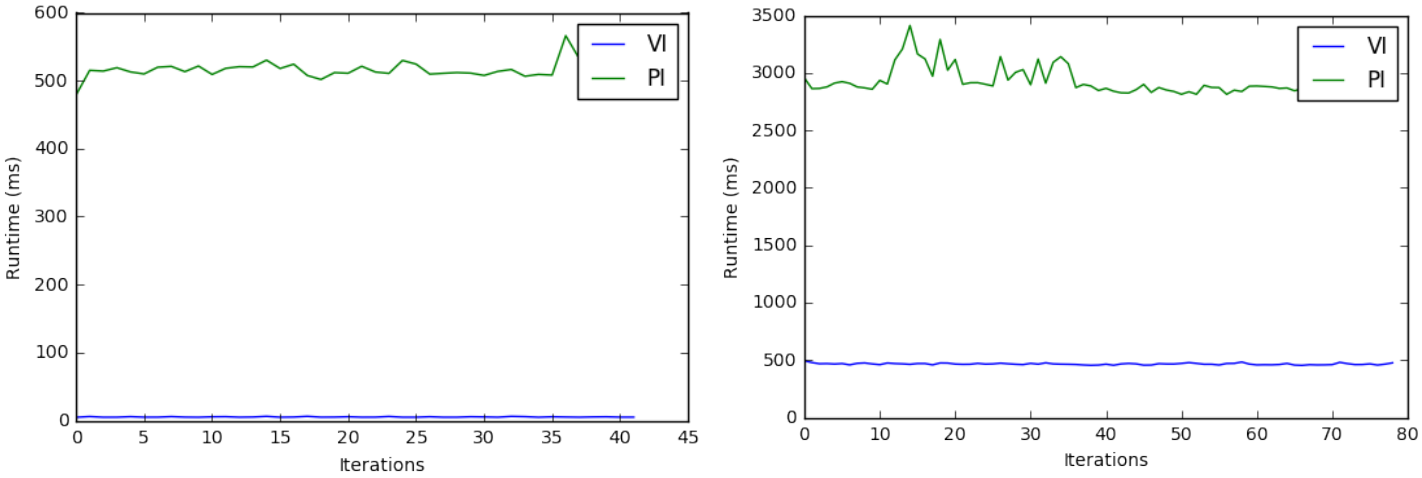
Optimal Policy Optimality in VI results when the maximum change in the utility of a state falls beneath a threshold. Utility estimates from both PI and VI are identical (Fig. 3.4), yet policies may be different, with Fig. 3.3b yielding an optimal policy as if the environment were deterministic, whereas FIg. 3.3a exhibits stochasticity that shows a balance between risks and rewards.

For VI, we default $\epsilon = 0.001$ across all of our experiments. This has a tantalizing effect determining optimality of value iteration. For PI, we avoid poor convergence performance by instituting an exploration function that assigns a higher utility estimate to unexplored regions of state-action pairs, allowing the agent to think that rewards are abound in places that haven't been visited. Thus, the agent chooses optimistic exploration over greedy exploitation and converges to near optimal policy quickly. The above mentioned criteria explain the difference in optimality in our maze problem.



(a) VI: 78 iterations (b) PI: 78 iterations

Figure 3.7: Optimality at 78 Iterations on a 200x200 Maze



(a) 20x20 Maze (b) 200x200 Maze

Figure 3.8: Runtime (ms) vs Iterations

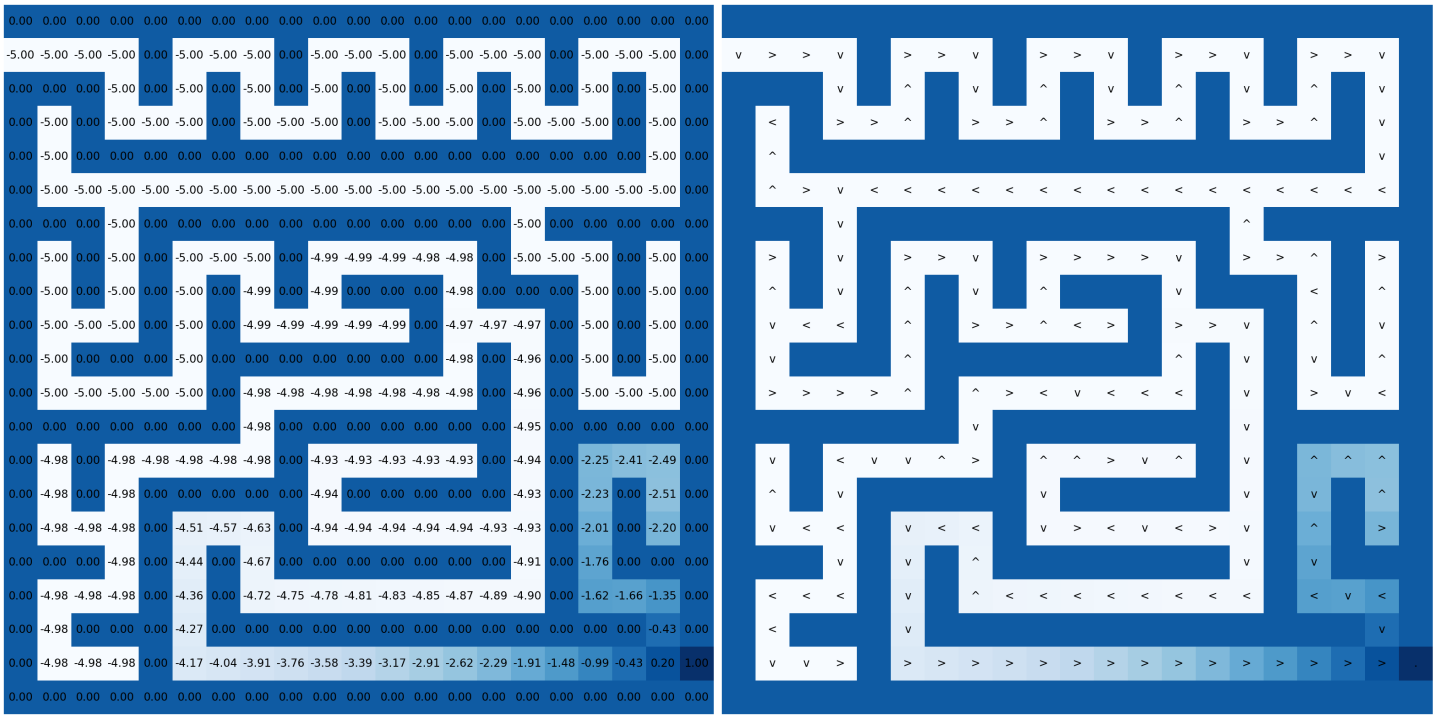


Figure 3.9: Q-Learning: Near Optimal Policy after 100,000 Iterations

ITERATIONS By tuning both the discount factor and the maximum error (ϵ) allowed in the utility of any state in the Maze example, we observe that the number of required iterations per discount factor is inversely proportional to ϵ as seen in Fig. 3.6. As values or information propagate by means of local updates, the maximum change, δ, in the utility of any state decreases per iteration until equilibrium. Since iteration continues until $\delta < \epsilon(1-\gamma)/\gamma$, the smaller the ϵ the more iterations are required for the change in utility to fall into the threshold.

CONVERGENCE CRITERIA Somehow related to the topic of iterations is convergence criteria. We measure both policy loss and maximum error, $||U' - U||$, as a function of iterations in our maze problem. Fig. 3.5 shows the maximum difference in utility between the corresponding policy and the optimal policy, and the max error from utility estimates per iteration. Note that policy loss dips slightly faster than utility errors; this confirms that policy iteration converges before values do in our examples.

Q-LEARNING Fig. 3.9 shows that Q-learning converges slowly to the tune of 100,000 iterations. Although there are ϵ-greedy variants of Q-learning [4], this paper considers a non-greedy variant to explore our MDPs optimistically. An agent following this algorithm has absolutely no incentive to visit undesirable states with little to zero rewards rendering these states seldom visited, if any, by chance. Such propagation leads to a slow convergence of utilities and a near-optimal policy after many iterations. Fig. 3.9 shows 100,000 iterations, and it is still ways from being optimal comparing to Fig. 3.4 applying either PI or VI.

5 CONCLUSION

We compare and contrast 3 algorithms, VI, PI, and Q-learning, using them to solve two MDP problems. These algorithms are variations of Bellman updates with varying degrees of efficacies for different situations.

In many instances an agent must learn by itself to complete a task. Both PI and VI take advantage of the necessary reward feedback to guide the system. This is useful especially when we have a complete knowledge of the system, where every possible state in it may be computed for planning, say, an agent's movements in a state space that isn't so large. In practice, PV converges fairly quickly and is therefore a more feasible method

than VI. VI, on the other hand, does not requires the knowledge of the system but all of the actions in it. This helps when we do not know the system a priori.

In contrast, Q-learning thrives on learning in the absence of a model. This algorithm pays no attention to any actual policy in its iteration; such off-policy algorithm is slow and yields no long-term benefit to MDPs where the state space is big and there is no apparent controlled policy to guide its agent through. This is to say that as our environment becomes more complex we are better off learning a model and a utility function with either PI or VI, than relying blindly on Q-learning to perform local updates with no consistency. This intuition applies to complex situations such as autonomous driving, weather prediction, and our maze problem in high dimension.

REFERENCES

[1] Norvig, P and Russell, S. *Artificial Intelligence: A Modern Approach*. Third Edition. 1994.

[2] Github Repository. *https://github.com/aimacode/aima-python*. Web. 25 Oct. 2016.

[3] The Markov Decision Problem: Value Iteration and Policy Iteration. *http://ais.informatik.uni-freiburg.de/teaching/ss03/ams/DecisionProblems.pdf*. Web. 21 Oct. 2016.

[4] Greenwald, A. and Hall, K. (2003). Correlated-Q learning. *AAAI Spring Symposium*, 242–249.

[5] Sutton, R. Learning to predict by the methods of temporal differences. *Machine Learning 3.1*, 9-44.