Assignment 4: Markov Decision Processes

Ran Chen
email: ranchen@gatech.edu

Abstract

This report tries to solve two different Markov Decision Processes (MDP) problems using Value Iteration (VI), Policy Iteration (PI), and Q-Learning (QL), and analyzes the differences among these methods. Code used to run experiments described in this report is hosted here: `https://github.gatech.edu/rchen350/cs7642summer2018p1`, which is also linked in the header as required

1 Problem: Random Walk

Due to the fact that most Markov Decision Processes (MDP) problems can be represented using a "Grid World", and also because they are easy to visualize, two grid worlds with different sizes were chosen in the report, namely "easy world" and "hard world", as shown in **Fig ??**. Whats interesting about this type of simple MDP problems is that visualizable changes can be made directly and comparisons can be extremely intuitive.

Transition: The transition function $T(s)$ in these worlds is set up as follows: at any position in the grid, the agent has movements in four directions, once the agent chooses one direction, it will have 80% probability to move toward the chosen direction, while the rest 20% probability is divided evenly among the other three directions. The outer edge of the grid world and positions marked with number 1 (red in **Fig ??**) indicate the wall, and if the agent makes a move toward the wall, it will stay at its original position before the move, without any penalty.

Reward: The reward function $R(s)$ in these worlds is set up as follows: the positions marked with negative numbers $-x$ carry a penalty, or negative reward of $-10x$ (blue in **Fig ??**). The terminal position is at the top right corner, which carries a reward of 100. All other numbers (including 0 and positive numbers) indicates a reward of -1. The agent always starts from the bottom left corner and the discount factor is 0.99.

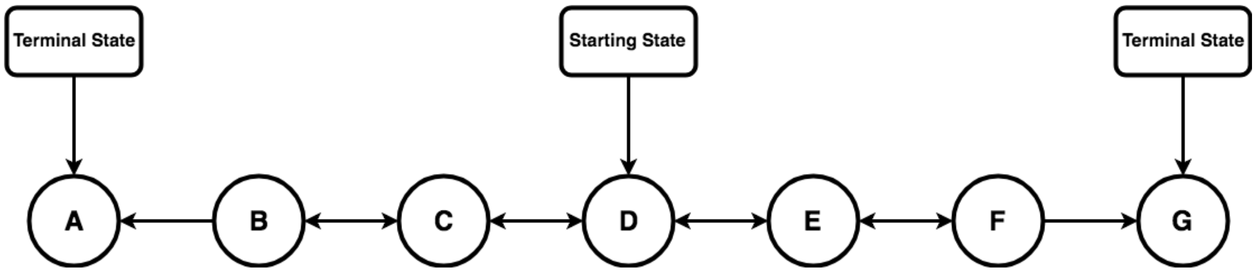

Figure 1: QL performances on the easy world. Variation of parameters: (a) initial Q, (b) initial Q (close-up), (c) learning rate, (d) epsilon, (e) epsilon decay rate, (f) optimal combination of parameters.

2 Experiments

Value Iteration (VI) starts with randomly initiated utility function $U_0(s), \forall s \in S$, where S is the set of all possible states, and updates the utility function $U_{i+1}(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U_i(s')$ at each iteration, then infer the policy $\pi(s)$ after U converges.

Policy Iteration (PI) starts with randomly initiated policy function $\pi_0(s), \forall s \in S$, at each iteration, first evaluate the utility corresponding to current policy $U_{i+1}(s) = R(s) + \gamma \sum_{s'} T(s,a,s')U_i(s')$, the use the calculated utility to update the policy $\pi_{i+1}(s) = \text{argmax}_a \sum_{s'} T(s,a,s')U_i(s')$.

Since the goal of a MDP is to find the optimal policy, and the policy may stay the same even for changing utility function, PI may converge sooner than VI does in terms of number iterations, however more calculation is need at each iteration for PI, the actual computation time needed for PI to converge may still be longer. Also since both are guaranteed to converge to the only solution, should the solution exists, the optimal policy obtained using both methods should theoretically be identical, given that the transition and reward function are both well defined an known to the agent.

2.1 Experiment 1

The easy world is a 10×10 grid with a wall within the second column and a enclosed space (shown in **Fig ??**). The optimal policies obtained from VI and PI are identical, shown in **Fig ??** and **??**, except the top right position, which does not matter since it is the terminal state and policy will not be evaluated further from it. Policies on this smaller world are quite easy to make sense of: (1) there is no policy at "walled" positions, i.e. positions marked as 1; (2) policies in the first column all go up because of the penalty of -30 at position (2,1) should be avoided; (3) policies at position (4,10) and (5,10) are at opposite directions, taking into account both the penalty of -50 at (9,10) and discount factor, which means if the agent is very close to the terminal state, it could be willing to go through that large penalty, otherwise it would choose the long way to go around it.

Now let's compare the performance of VI and PI. **Fig ??** shows the changes in reward over the iterations for both VI and PI: PI converges with fewer iterations as expected. **Fig ??** and **??** clearly shows the algorithm converges at iteration 15 for PI and 36 for PI. However, when looking at the actual time **Fig ??**, VI actually takes shorter time to converge compared to PI.

2.2 Experiment 2

The hard world is a 20×20 grid with much more complicated wall structures and more penalty positions (shown in **Fig ??**). The optimal policies obtained from VI and PI are also identical in the hard world case, shown in **Fig ??** and **??**, except the top right position, which does not matter since it is the terminal state and policy will not be evaluated further from it. Policies on this larger world are more difficult to make sense of by looking at it, however the general tendencies seem to be correct: (1) choosing the shorter path as often as possible; (2) avoiding penalty positions.

Again the performance comparison of VI and PI in hard world is shown in **Fig ??**. **Fig ??** shows the changes in reward over the iterations for both VI and PI: PI converges with fewer iterations as expected. **Fig ??** and **??** clearly shows the algorithm converges at iteration 15 for PI and 35 for PI. Again, when looking at the actual time **Fig ??**, VI actually takes shorter time to converge compared to PI.

3 Results

Q-Learning (QL) is a model-free learning algorithm, in the sense that both transition and reward function are unknown to the agent and it only needs to know what states exist and what actions are possible in each state. As a result, the agent will have to observe the next state and reward once it takes a specific action from a particular state. Also note that the the same state and action pair may need to be repeated multiple times since both reward and transition function can be stochastic. In addition, unlike VI and PI, which knows the rewards for all states and thus can calculate the utility for all states, QL only observes reward at the states it has visited, so if it always follows the best policy it knows, it will not visit those previously unvisited states, thus will not know the value of those positions.

A QL starts with randomized initialized $Q(s, a)$, $\forall s \in S$ and $a \in A$, where S is the set of all states and A is the set of all actions. Then the agent takes an action a according to the policy $\pi(s)$ inferred from current $Q(s, a)$, lands in a new state s', observes reward r, then update $Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a'))$, the agent keeps making actions according to policy inferred from current Q, and update Q until it reaches the terminal state, then the agent will repeat the whole process from the starting state until converge. Here α $(0 \leqslant \alpha \leqslant 1)$ is the learning rate, which determines how much the agent believes newly observed information over the prior belief. If α is large, the agent makes dramatic updates to Q, which may causing the rewards of different iterations to fluctuate significantly. As mentioned above, the problem is that, the agent's ability to explore positions that are unvisited during previous iteration is limited by the transition function: if the transition function is deterministic, then the agent is unlikely to explore unvisited states. In order to balance between exploitation and exploration, we set the agent to ignore the policy inferred from current Q, and take a uniformly distributed random action with the probability of ϵ. Again, for very large ϵ the rewards in different iterations are also expected to change drastically due to stochastic movement of the agent.

3.1 Experiment 1

Similar to VI and PI, QL is tested on the easy world with variations of parameters described above.

Initial Q has a huge impact on the behavior of a QL agent. **Fig ??** shows the reward curve of QL agents with $\alpha = 0.1$, $\epsilon = 0.1$ with no dacay, and Q values initiated at 100, 0, -100 respectively. Agent with 0 initial Q converges after 500 iterations. Higher initial Q encourages "optimistic search", meaning the agent will be willing to explore since all possible action would lead to scores lower than current Q value, as indicated by the green curve, due to such "optimistic search" the algorithm converges within much fewer iterations. Agent with a low initial Q value could get stuck on some very low reward value, because the agent is discouraged from exploring since all unvisited states have very low Q value. Reward curve of the agent with initial Q value of -100 is shown in purple, which got stuck at total reward of -200, and stoped within 30 iterations (closeup shown in **Fig ??**).

Learning Rate is another important parameter that heavily influences the agent's behabior. We tested different learning rate at 0.1, 0.2 and 0.9, with initial Q set to 0 and $\epsilon = 0.1$ with no decay, the results are shown in **Fig ??**. Agents with learning rate of 0.1 (green curve) and 0.2 (red curve) converges at approximately 550 and 400 iterations respectively. This is expected, since at learning rate of 0.2, the agent will update the prior belief on Q a little more aggressively, so the ground truth of the Q value at each state will propagate faster. However with higher learning rate at 0.2, the irregular fluctuations happened more often. Such fluctuation was especially significant in the case of learning rate 0.9, in which case the agent put little faith on the Q value previously learned, thus causing the reward value to jumpy drastically (purple curve).

epsilon (ϵ) dictates how likely the agent chooses exploring over exploiting. With probability ϵ at each step, the agent will take a uniform random action. The comparison of reward curves from agents with different ϵ values (0.1, 0.3, and 0.5) is shown in **Fig ??**. With relatively low learning rate at 0.1 and moderate initial Q value at 0, larger ϵ causes the agent to search for unvisted states more aggressively, thus agents with $\epsilon = 0.5$ reaches convergence much earlier than the other, at 230 iterations, however due to its more stochastic nature, the reward curve shows more significant fluctuations (purple curve). Agents with ϵ of 0.1 and 0.3 converges with more iterations, but the curves are much smoother.

epsilon Decay (ϵ Decay) controls how ϵ decays during the training process. Typically, the agent need to be more aggressively search in unvisited states during the initial period of learning, because the agent does not have much information about the world yet, while during the later, or ending period, the agents need to emphasize on its optimal policy rather than choosing random actions. Thus typically ϵ can be decayed by a constant rate over the iterations. As show in **Fig ??**, the reward curves of agents with no decay and with a decay rate of 0.99 both reach the maximum reward level within similar number of iterations, however the non-decaying agent causes much more fluctuation in the reward curve (green curve) than the one with decaying ϵ (red curve).

Optimal Strategy is typically a balance among all the parameters described in this section, e.g., as shown in **Fig ??**, in order for the agent to reach higher reward, within fewer iterations, and not creating too much fluctuations, especially during the later stage in the learning process, the combination of the parameters need to be tuned carefully.

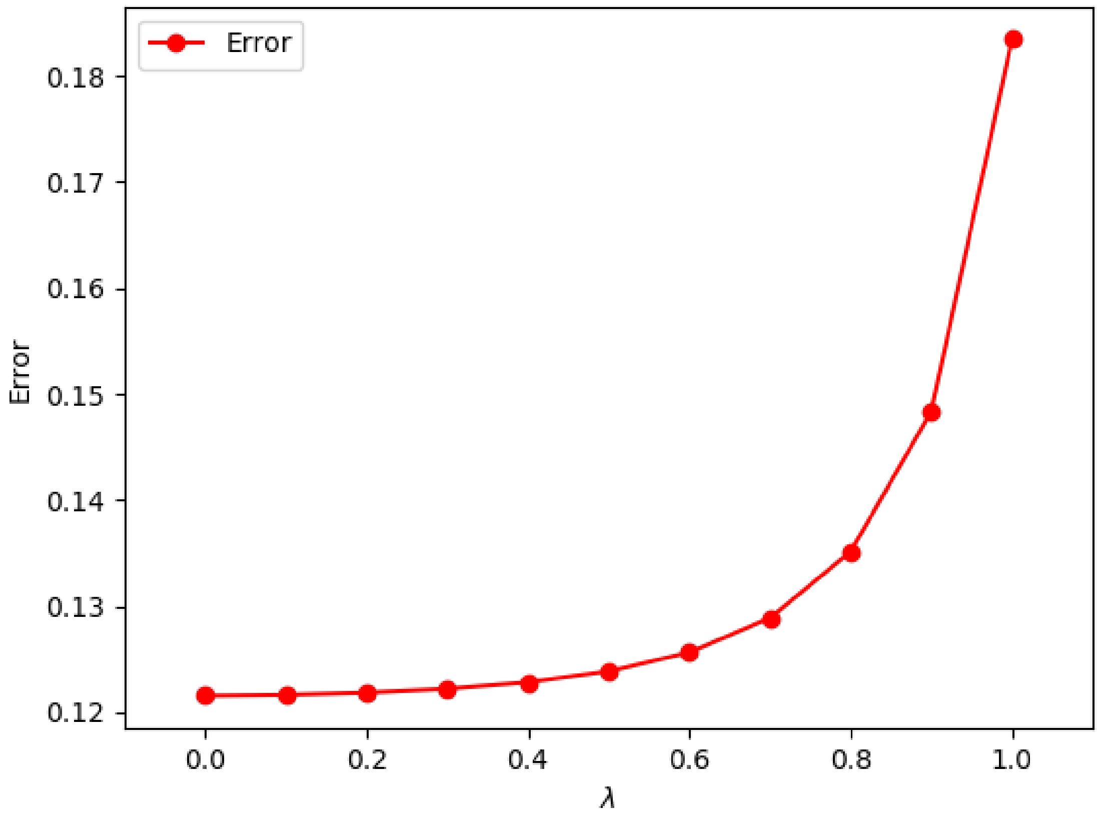

Figure 2: QL performances on the easy world. Variation of parameters: (a) initial Q, (b) initial Q (close-up), (c) learning rate, (d) epsilon, (e) epsilon decay rate, (f) optimal combination of parameters.

3.2 Experiment 2

Similar arguments could still hold in the case of hard world here, as shown in **Fig ??**, only the fluctuation is much more significant, due to the fact that there exist many more penalty states in this world now, and the agent requires much more iterations to reach the highest possible reward because the world is now larger and more complicated. Also note that in this more complicated world, aggressive exploration (higher initial Q, epsilon, and no decay) or higher learning rate may not be able to help the algorithm converge sooner as it may in the case of easy world, e.g. in **Fig ??**, the agent with initial Q value of 100 actually required more iterations to reach the same level of reward as the one with initial Q value 0 did.

4 Difficulties

More experiments were done on how the size of the world affects the performance of the three algorithms. In these experiments, the worlds were constructed with only 0s, which indicates penalties of -1. These worlds did not have any walls (values of 1). Reward at the terminal state is still 100. **Fig ??** shows the performance differences of VI, PI, and QL with respect to world sizes. **Fig ?? ??** indicates that all algorithm were able to converge to the same reward for world sizes up to 35. **Fig ??** and **??** shows that for VI and PI, although the number of iterations remain largely unchanged, the time required is in increasing, indicating more computation per iteration is required for larger worlds. Also the higher increase rate in time and number if iterations required for QL with $\epsilon = 0.1$ is because lower epsilon would require the agent to take much longer

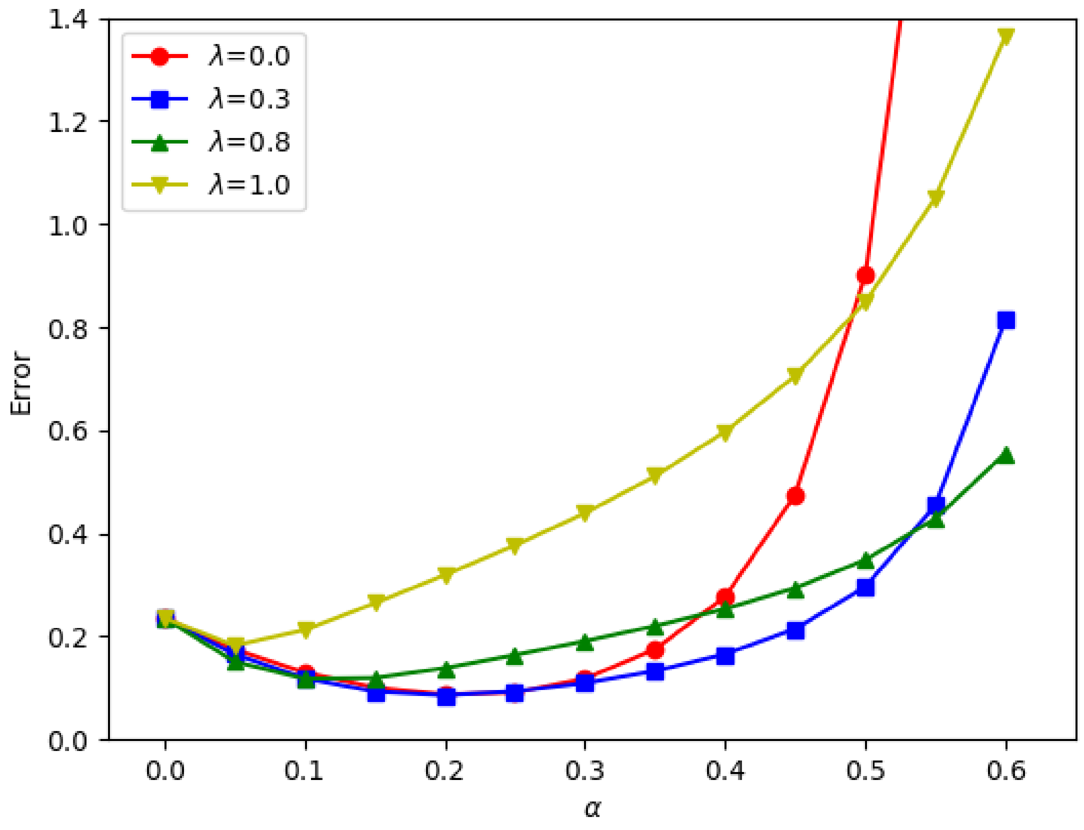

Figure 3: QL performances on the hard world. Variation of parameters: (a) initial Q, (b) initial Q (close-up), (c) learning rate, (d) epsilon, (e) epsilon decay rate, (f) optimal combination of parameters.

to explore the unvisited states.
%

5 Appendix

Code used to run experiments described in this report is hosted here: `https://github.gatech.edu/rchen350/cs7642summer2018p1`.

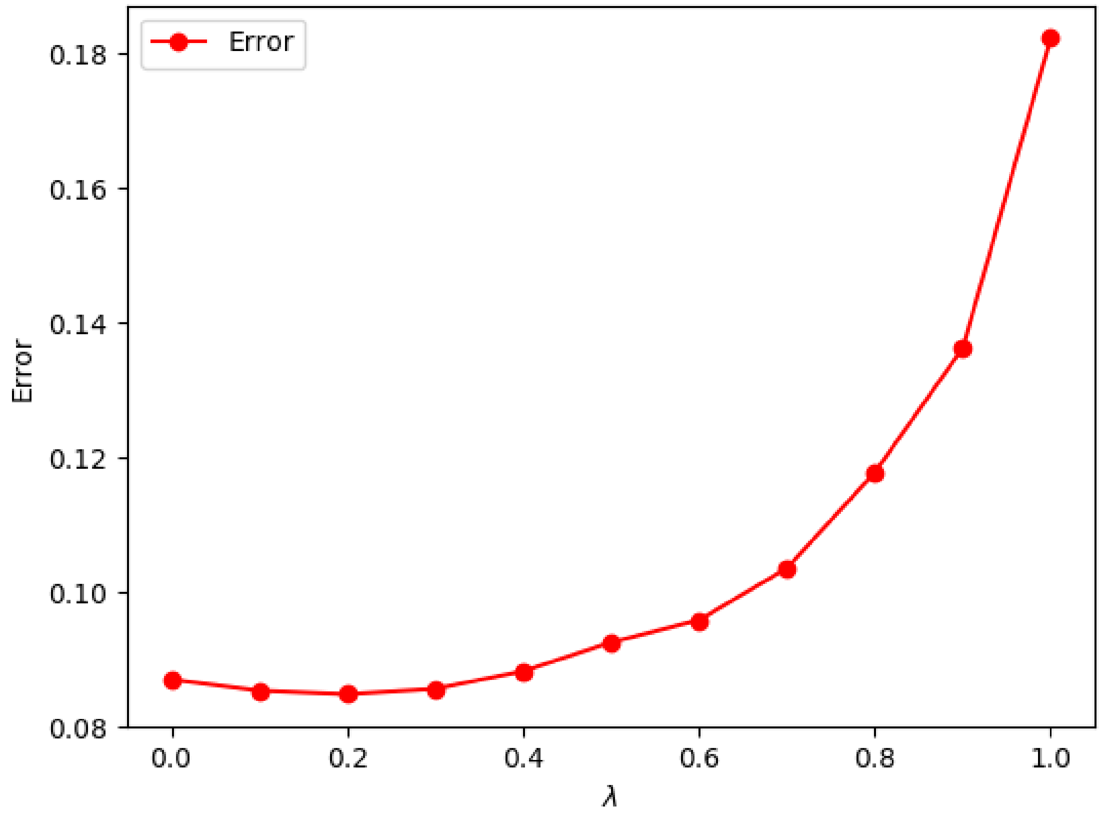

Figure 4: QL performances on the hard world. Variation of parameters: (a) initial Q, (b) initial Q (close-up), (c) learning rate, (d) epsilon, (e) epsilon decay rate, (f) optimal combination of parameters.