Markov Decision Processes

Yi Zhao

[Yzhao644@gatech.edu](mailto:Yzhao644@gatech.edu)

1. Introduction

This paper discusses methods to solve two Markov Decision Processes problems. Three methods, Value Iteration, Policy Iteration and Q-learning method were introduced to solve two MDP problems, a non-grid Forest Management Problem with larger states and a grid problem, Frozen Lakes with relative smaller number of states.

Further discussed are convergence analysis, hyper parameter’s individual effect on how the Q-learning can converge to optimal policy.

2. Problems

Since we are required to use Value Iteration and Policy Iteration to solve the problems, the problem needs to be model based and a transition matrix is required for the two algorithms. It easier to form the transition matrix for distinct state problems. Model based and distinct states are two reasons for both of the below problems.

2.1 Frozen Lake (small state problem)

For the grid problem, I used Open AI Gym’s frozen lake with some modification on the rewards. The states are locations on the frozen lake. Agents could take 4 actions, i.e. up, left, down, right. The grid is a square frozen lake that has several holes that agent can fall into. There is one grid which is the ultimate reward location. If the agent can reach that location, a final large reward is granted. When agent takes an action, there’s only 1/3 chance that the action will be taken, since the frozen lake is slippery, the agent might end up in the other directions. For up move, agent could move to left or right with 1/3 chance.

Since I will use mdptoolbox to solve the problem with Value iteration and policy iteration, I’ll need to make the reward and transition matrix transfer from gym environment to mdptoolbox. Detail could be found in section 3.1.

There are three reasons I liked about this problem. Firstly, it is a small problem with 16 states. Secondly, it is a real-life problem that is easy to interpret and relatable. Also, it is easy to visualize the result. Lastly, the agent could have multiple actions and the action is dynamic, with a probability to move to unintended directions, making this small state problem complex to solve.

2.2 Forest Management (large state problem)

Forest management is a non-grid problem working on deciding whether to cut a forest to make money or reserve it for wild animal. There is a possibility of p that the forest could be burnt down to back to initial state. States are defined by the life of the tree. The actions are cutting or reserve the forest. Immediate rewards are 1 for cutting before final state, and 200 for final state. And 0 for reserving the forest before the final state and 400 for final state.

The reward is gain chosen by how the q learning will work with mdptoolbox.

The reasons I liked about the problem are it is a non-grid problem. The actions space is small to handle large state spaces.

Larger problem needs larger gamma for method to work. Otherwise, it’s all cutting and done.

3. Value Iteration and Policy Iteration

In MDP problems, the goal is to find the optimal policy to maximize the total rewards. It focus on the total rewards instead of immediate rewards. The bellman equation is used for reinforcement learning problems.



Dynamic programming is a method to solve the problem. For both value iteration and policy iteration, the overall idea is similar. First step is to get an update of the value table and then exact policy according to the table.

Text, letter

Description automatically generated

Value iteration and policy iteration are two algorithms to solve MDP problems. For value iteration, we begin with random value function. Then we step through each state and get the Q table and update the value function with the max value of Q(s,a). this continues until the improvement is less than the threshold or maximum iteration is reached.

For Policy iteration, we start with a random policy. then we use the new policy to update value table and see if the value table converges as above. If not, we update the value table and recreate a policy with the largest state action combination as above the repeat, until converges.

According to the above definition, one of the difference is that policy iteration might need more time to converge, as in each iteration, PI has to finish all the policy before checking the convergence. But in terms of number of iteration needed for convergence, it’s usually hard for policy to use many times to converge. Details will be shown in part below.

For both QI and PI, one hyper parameter is the discount rate Gamma. Plots are also created to show as the gamma increase, the iteration to converge increases.

3.1 Frozen lake problem

For frozen lake problem, I transformed open AI Gym’s environment to the P and R table for the mdptoolbox. The changes include immediate reward and P and R tables.

The immediate reward is the expectation of the actions that the agent could take. For example, if the move makes the agent reach to the ultimate reward, which could award agent 10 points, the reward for that action is given by 1/3(10+0+0).

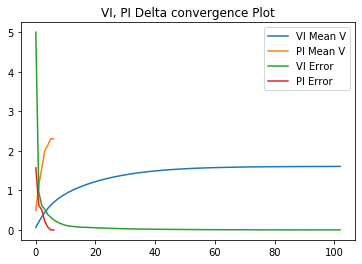
My modification from the gym environment is change the ultimate reward from 1 to 10, this is to encourage agent to go further to reach the ultimate state. Also, there’s a -5 penalty for agent to fall in the hole and -0.5 for staying on the path. This is set up for agent to reach the ultimate location as soon as possible. Also for mdptool box’s Q-learning algorithm implementation, there’s no terminal states but a state reset for every 100 steps. It is important to have the agent reach to goal as soon as possible instead of staying on the path and not moving forward.

For the transformation part, I create the P table with dimension as action\* state \* state(4\*16\*16), with probably as 33% for each action. For R table, the shape is state\*action, with each reward as the expectation of the state with the action.

Froze lake optimal policy.

(0, 3, 3, 3, 0, 0, 0, 0, 3, 1, 0, 0, 0, 2, 1, 0)

|  |  |  |  |
| --- | --- | --- | --- |
| S  left | F  up | F  up | F  up |
| F  left | H  left | F  left | H  left |
| F  left | F  down | F  left | H  left |
| H  left | F  right | F  down | G  left |

 A picture containing chart

Description automatically generated

Chart, line chart

Description automatically generated

Forest

Graphical user interface, application

Description automatically generated

Chart, line chart

Description automatically generated

A picture containing chart

Description automatically generated

3.2 Forrest Management problem

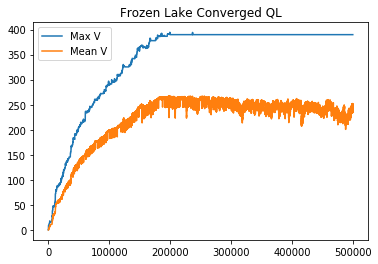
For forest management problem, I use state as 400 to make it a large problem. The fire probably is 0.1, which means every time we reserve the forest, there is 10% chance the forest will be burnt down. The final reward for reserving to the final state is 400 and 200. This is chosen with the mdptoolbox setup. As there’s not terminate state and gamma is set up to be 0.999. after 400 discount, the reward is still worth agent to go for.

I’ve also updated q learning source code to generate random state for every 400 steps.

4. Q-learning Algorithm

Q learning can converge to local optimal with small epsilon.

lake

Chart, histogram

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

forest

A picture containing line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Chart, line chart

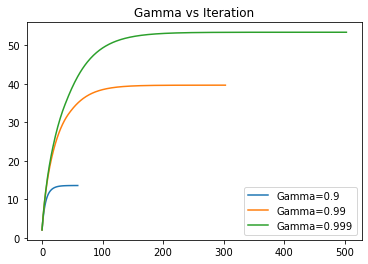
Description automatically generated

Chart, line chart

Description automatically generated

5. Solution for two problems

5.1 convergence

  
Diagram

Description automatically generated

delta convergence plot

5.2 Q-learning Hyper parameters: greedy method, epsilon greedy, all explore.

Forest management delta converge

A picture containing diagram

Description automatically generated

5. Summary

VI converge faster than PI in terms of time, but needs more iterations to converge.

VI and PI converge to the same optimal

Reference

P3: Exploration Strategies

Plot error to see converge?

VI: Epsilon, Gamma

PI: Epsilon, Gamma

Q: gamma, alpha, alpha\_decay, alpha\_min, epsilon, epsilon\_min, epsilon\_decay, n\_iter

How does discount affect convergence?

I can see here higher discount takes more iterations to convergence ?

How is that so?

we should do this (vary the size of each MDP) because a grid world and non-grid world problem might scale differently in terms of how policy and value iteration handle it.

probs\_f, rewards\_f = hiive.mdptoolbox.example.forest(S=3000, p=0.001)

epsilon greedy

greedy

if ultimate reward is too small or gamma is too small and steps are long to get the final reward, they will choose to cut no matter what.