Rental Car Problem

Name: Kunal Lad
UT Eid: KL28697
Email: kunal.lad@utexas.edu

Goal

Goal of this assignment to study policy iteration approach through continuing finite MDP problem (Jack's rental car problem described in Chapter 4 of textbook [1]). We also study how this approach can be extended when state space is large.

Problem Description

Jack manages two locations for a nationwide car rental company. Each day, some number of customers arrive at each location to rent cars. If Jack has a car available, he rents it out and is credited \$10 by the national company. If he is out of cars at that location, then the business is lost. Cars become available for renting the day after they are returned. Cars can be moved between the two locations overnight, at a cost of \$2 per car moved.

Number of cars requested and returned at each location are Poisson random variables with means 2, 3 and 2, 1 respectively as shown in fig 1. Both locations have maximum capacity of 6 (any additional cars are returned to the nationwide company, and thus disappear from the problem) and a maximum of 3 cars can be transferred on any particular night.

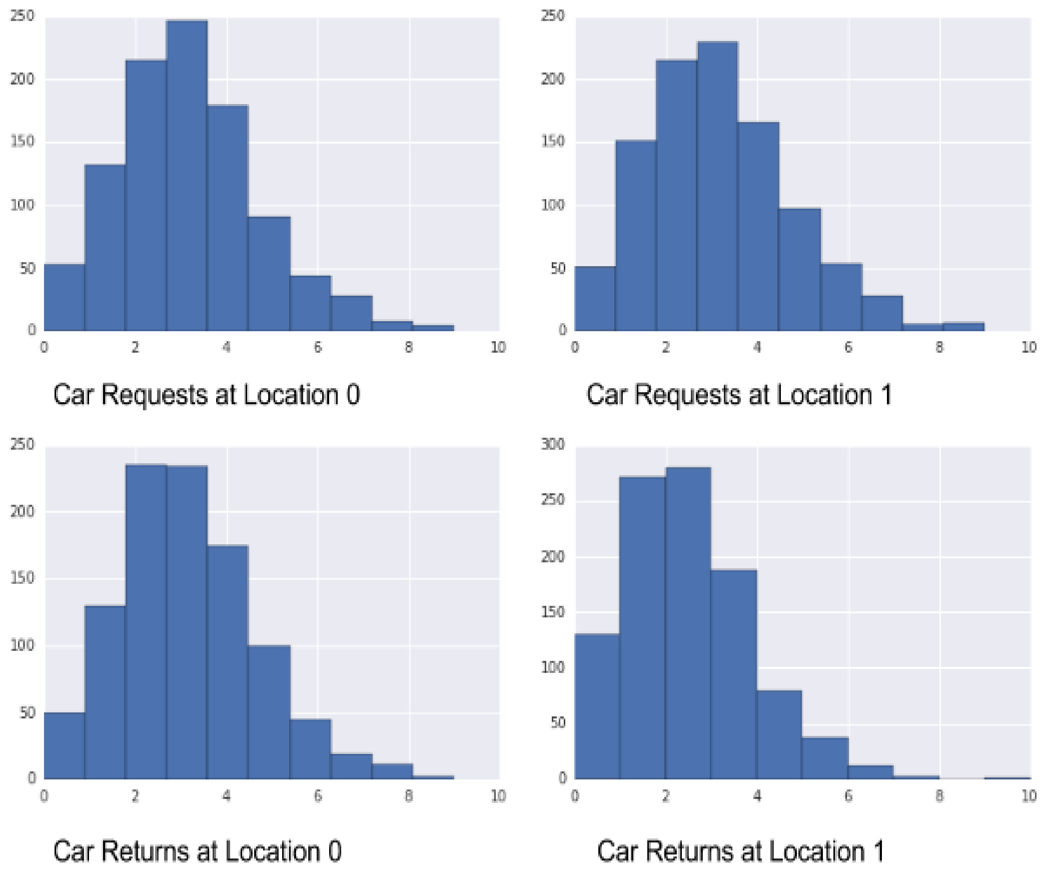


Fig 1 Probability distributions for car requests and returns at both locations

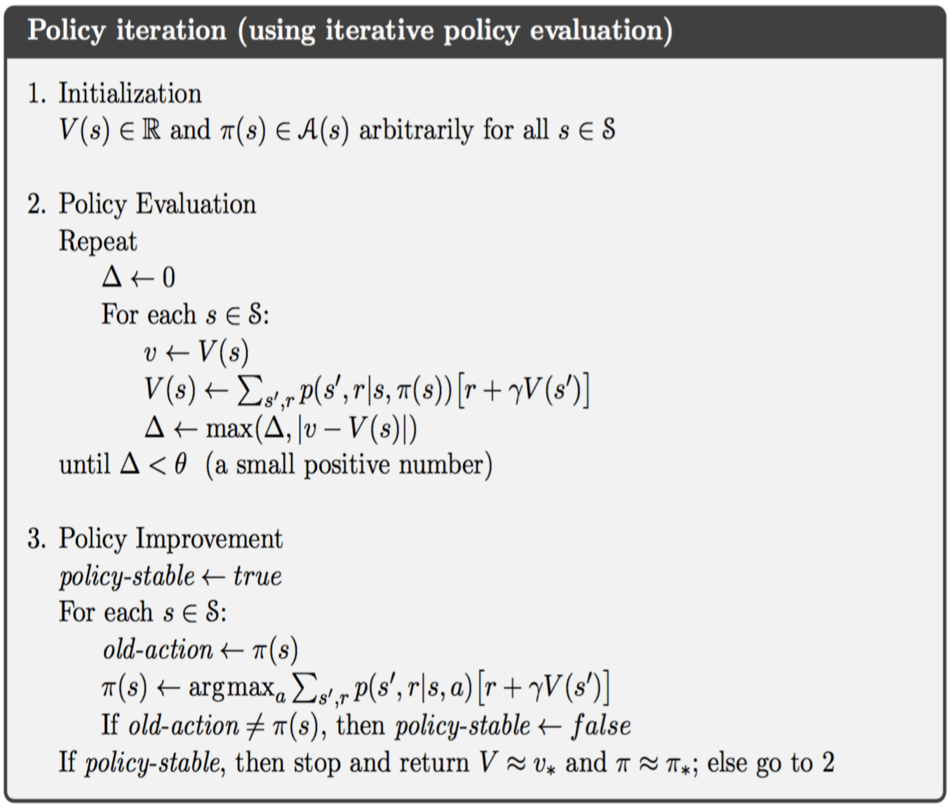
Methodology

Approach

I used the policy iteration algorithm described in the textbook [1] for solving this problem. It consists of alternate cycles of policy evaluation and policy improvement. For practical purposes we stop both policy evaluation and policy improvement after some predefined number of iterations depending on problem scenario. Fig 2 shows pseudo code for the algorithm.

Implementation

I implemented the above algorithm in ipython notebook. All the code and results for various experiments which were performed can be found at my github profile under the repository rental_car_problem



Policy iteration (using iterative policy evaluation)

```
1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
       $\Delta \leftarrow 0$
       For each $s \in \mathcal{S}$:
           $v \leftarrow V(s)$
           $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
           $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   $policy\text{-}stable \leftarrow true$
   For each $s \in \mathcal{S}$:
       $old\text{-}action \leftarrow \pi(s)$
       $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
       If $old\text{-}action \neq \pi(s)$, then $policy\text{-}stable \leftarrow false$
   If $policy\text{-}stable$, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2
```

Fig 2 Policy Iteration Algorithm (Fig Credits [1])

Experiments & Results

In this section we describe present various experiments for studying Policy Iteration approach for continuing finite MDP problems. We also discuss the problem with policy iteration for large state spaces and how we can tackle it.

E1: Simple version v/s Modified version

In this experiment we investigated how the optimal policy changes if we add additional constraints to the original problem. In the modified version we added the following modifications similar to what is suggested in the textbook [1]:

- One car can be moved from location 0 to location 1 for free
- Parking cost of \$3 if number of cars exceeds 3 at any location.

Results are shown in fig 2 below. As expected free car moving from location 0 to 1 causes that action to be taken more often for modified version as is highlighted in fig 2. However additional parking cost does not cause any change in the policy. We can see that optimal policy for simple version already moves cars for states (4,0), (4,1), (0,4) and (1,4) for which you would have expected the modified version to move cars. So we don't observe any change in policy. Another reason is that additional parking cost of \$3 is not very large as compared to moving cost of \$2 per car. My hypothesis is that this constraint's effect will be more observable for larger state space if parking cost is higher than moving cost since it will affect many more states in optimal policy then.

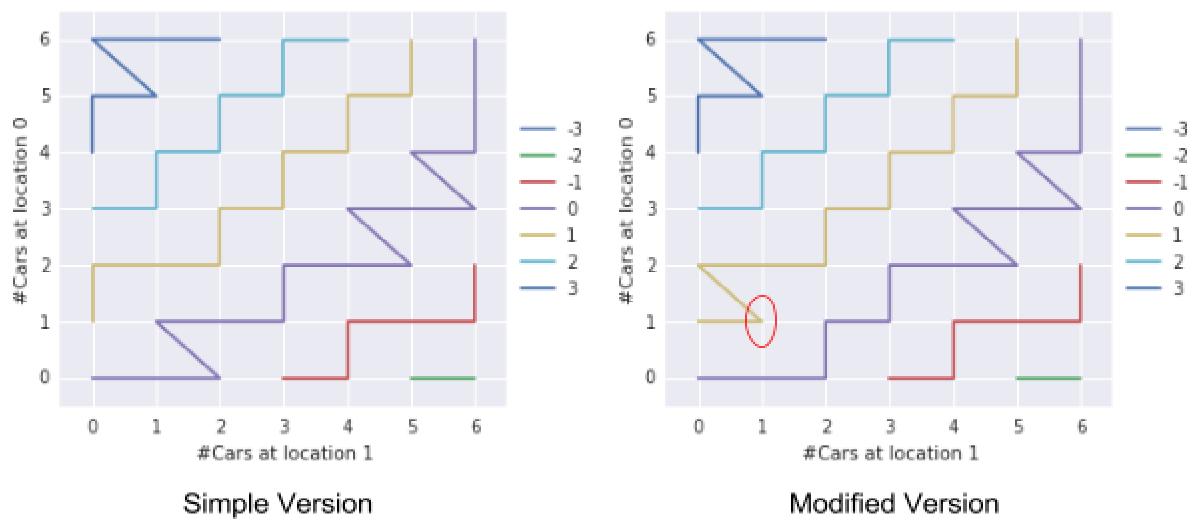


Fig 2 Simple Version v/s Modified Version

E2: Poisson Distribution v/s Uniform Distribution

In this experiment I investigated how the policy changes if we use uniform distribution for requests and returns (with same means as before) instead of poisson distribution. Expectation computation for poisson distribution is extremely computationally expensive O(N^4) for large state space. Motivation for this experiment was to see if we could use a simpler probability distribution for approximating the poisson expectation. Fig 3 shows the results of this experiment. We see that the optimal policy for small state space is almost similar for both poisson and uniform distribution of requests and returns. This suggests that we can use uniform distribution for approximating poisson distribution expectation for large state space.

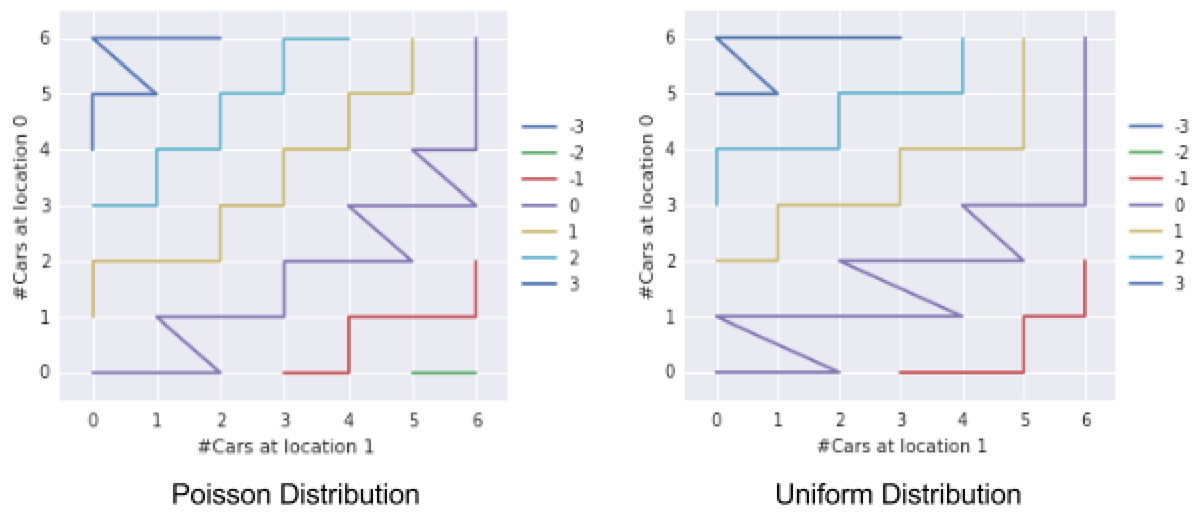
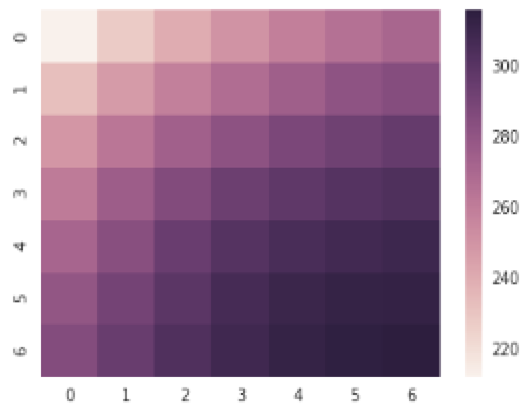


Fig 3 Poisson v/s Uniform Distribution

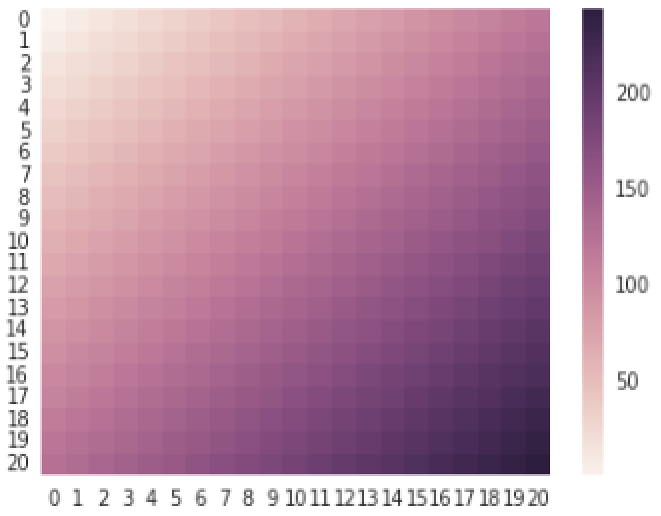
E3: Approximate solution for Larger State Space

In this experiment we investigate several approaches to generating an approximate solution for large state space using solution for smaller state space.

We observed that optimal state values for smaller state space had a pattern which seems to be learnable by simple regression techniques. I applied Support Vector Regression(SVR) for this. Fig 4 shows the learnt values for larger state space with maximum capacity 20. Comparing it to state value plot for smaller state space, we observe that it has similar pattern and seems reasonable since it has higher values for states with larger number of cars. This suggests that our estimated state values are good.



Optimal State Values for Small State Space



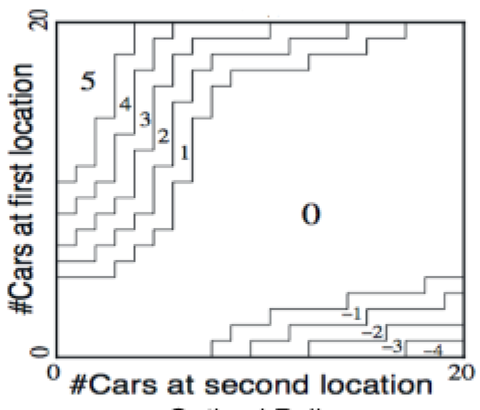
Learnt State Values for Large State Space

Fig 4 State Values

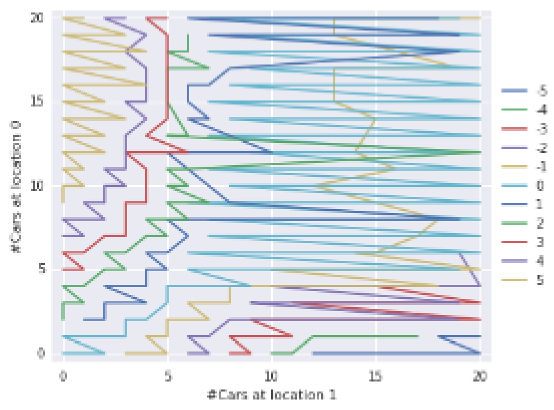
After learning good initial estimate for state values we apply policy iteration to learn the policy from initial state value estimates. As stated in previous experiment poisson expectation computation is extremely computationally expensive. So we try the following 2 ways to approximate it:

- Poisson sampling
- Uniform probability distribution

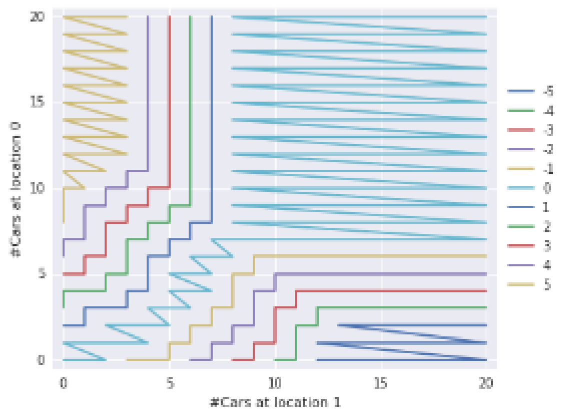
Fig 5 below shows the policies obtained by these 2 approaches along with the optimal policy plot which was taken from textbook [1]. Surprisingly we observe that uniform distribution policy plot looks more like optimal policy than poisson sampling plot. A possible reason for this could be that the number of samples used for poisson estimation is not sufficient to estimate the expectation to a good extent.



Optimal Policy



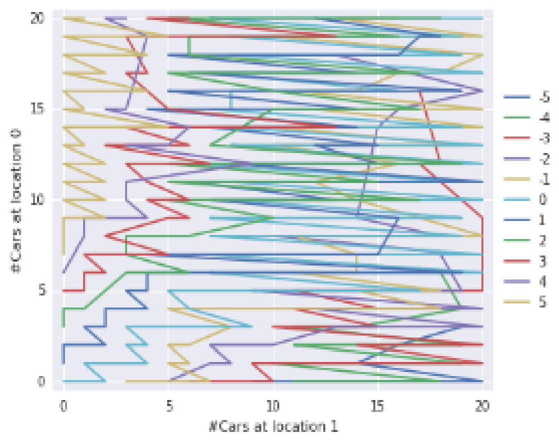
Poisson Sampling



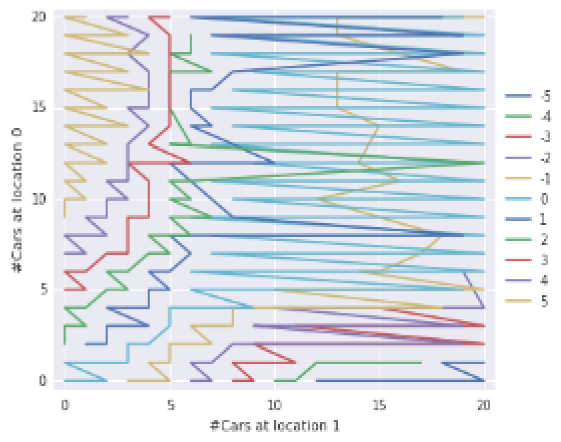
Uniform Distribution

Fig 5 Comparison of approximate solutions to optimal solution

To investigate this hypothesis mentioned above, we perform another experiment in which we vary the number of samples used for poisson estimation and see how the the policy evolves with it. Fig 6 shows the results. We observe that as the number of samples used for approximation increases, policy moves more and more towards the optimal policy, thus confirming our hypothesis for poor performance of poisson sampling.



100 Samples



1000 Samples

Fig 6 Poisson Estimation with different number of samples

Conclusion

If environment model is know we can exploit structure in state - action space and apply supervised learning techniques to learn state values/policies to:

- Get good initial guesses for State Values and Policy
- Approximate solution when larger space

References

[1] Reinforcement Learning: An Introduction