Markov Decision Processes

Feng Zhang
fzhang326@gatech.edu

1. Introduction to data

1.1 Small dataset: fishing

The first dataset is about whether to fish salmons. There are four states for the amount of fish in the lake: Empty, Low, Medium and High. For Low, Medium and High states, the fishermen have two actions which are fish or not fish. For Empty state, only non-fish is available. For each state, there is a probability that fish/non-fish will jump to another state or stay in the same state. Also, if the state is in Empty then it will cost $200K, at other states, fishermen will receive reward for fishing and no reward for non-fish. Most cases, non-fishing will increase volume of fish but small chance it will maintain the current state without fishing

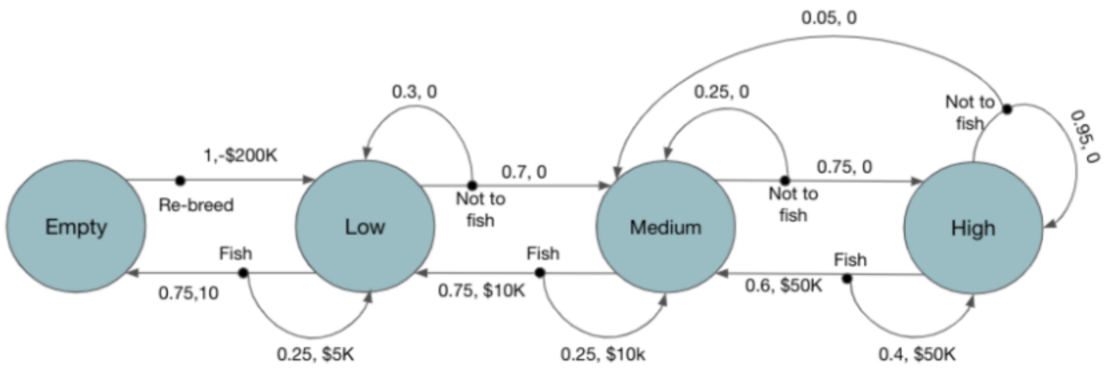


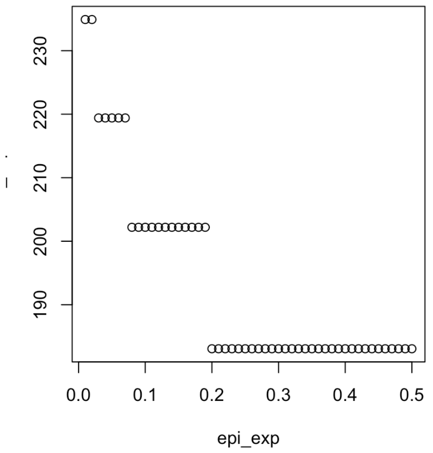
Figure 1: Transition graph of fishing salmons MDP

1.2 Large dataset: Random generation.

The second dataset I applied used a randomized generated case which contains 500 states and 10 actions. This is a large dataset which will include a 500*500 probability matrix for each action. I applied this data mainly due to the interest of parameter tuning. I want to use this dataset to see what the results change when I changed the parameters such as the epsilon and discount rate. The overall purpose is to see whether the results will meet the expectation for each scenario and understand the underlying mechanism.

2. Models: fishing

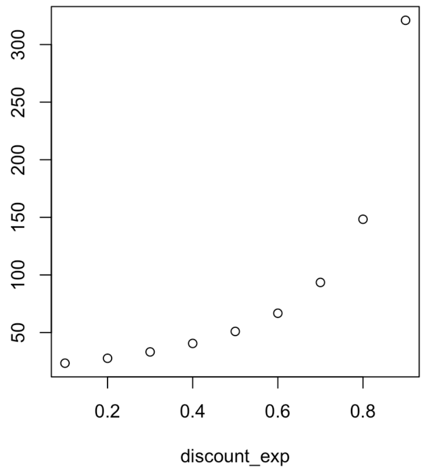
For the fishing dataset, we will first apply the policy iteration, I will run multiple iterations using different parameters: discount. Discount varies from 0.1 to 0.9. I will check the convergency for each iteration.



2.1 policy iteration

For the policy iteration, it converges very quickly and have identical policy across all discount values. The optimal policy would be 1 1 2 1 which is "holding, holding, fishing and fishing".
This policy makes sense to me since only this strategy could make sure there is enough fish in the pond.

The Figure 1 is to measure the optimal value change as the discount increases from 0.1 to 0.9. We can see that as the discount increases, which means more historical policy matters, then the overall value increases which is reasonable.



Since this problem is not complicated so the algorithm can only run 1 iteration to get the problem done. I have tried different discounts and all return the 1 iteration. I think this is due to the environment settings which does not too much computing. This is reasonable since it means that past value matters more than small discount value.

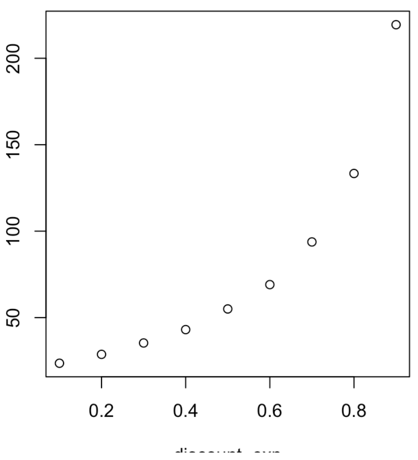
Therefore, I upgraded the environment: changing the reward matrix and the probability transition matrix and rerun the model. I increase uncertainties among the transition matrix which set most of them to 0.5. In this case, the model can still converge pretty fast in the policy iteration. In this case, I would choose the running time as the criterion to define the convergence. With random initial policy, the iteration usually will converge within 10 times which is pretty fast. In this case, I did not discuss the impact of states number in the running time since the states are fixed. However, in the next scenario, I would discuss how the number of states change impact the total running time.

2.2 Value iteration

For the value iteration, I also plays with different parameters: the epsilon and discount. The epsilon varies from 0.1, 0.05, 0.01, 0.005, 0.001 and the discount changes from 0.1 to 0.9.

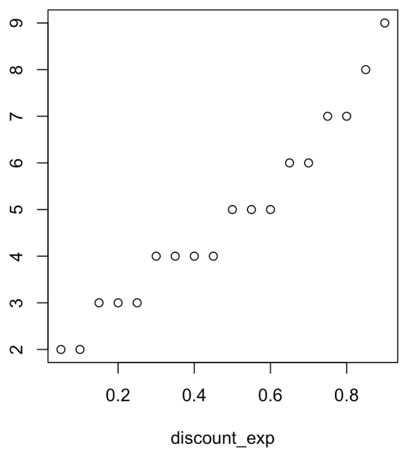
First experiment I did is to set the epsilon equals to 0.05 and the discount rate setting as 0.9. This experiment has a quick convergence rate. One of the interesting result I got from this experiment is that the optimal policy changes from previous "1, 1, 2, 1" to "1, 1, 2, 2". This means the fisher could do "holding, holding, fishing, fishing".

The right figure shows the optimal value changes as the discount rate changes from 0.1 to 0.9. Similar to the policy-iterated one, the overall trend shows an increase trend. However, the optimal value was lower than the policy iteration algorithm when the discount rate is 0.9. This result is also pretty interesting.



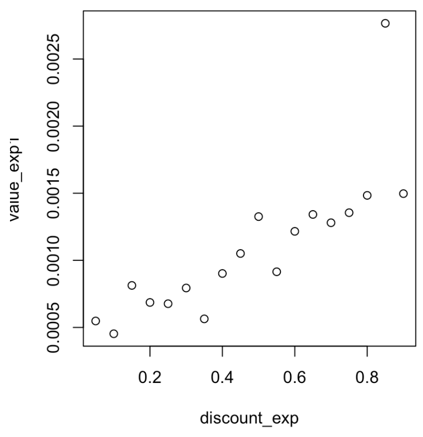
Next I fixed the discount to the 0.9 and check the performance of model when the epsilon varies from 0.01 to 0.5. Epsilon was used to determine the final convergence rate so I will also check the result of iterations vs epsilon.

The right figure is to express the relationship between epsilon and the optimal value. We can see that as the epsilon increases, the optimal value decreases. When the epsilon increases to some value around 0.2, then the optimal value did not change anymore.



Therefore, I further checked the relationship between the iteration vs. discount rate. The discount rate varies from 0.05 to 0.9 and the y axis is the iteration times. The epsilon was set as 0.05 as the global variable. Overall, the running iteration times did not varies significantly but there is an increase trend when the discount rate increases. From the right figure, we could see that the even through sometimes when the discount is 0.9, then the required iteration is the largest.

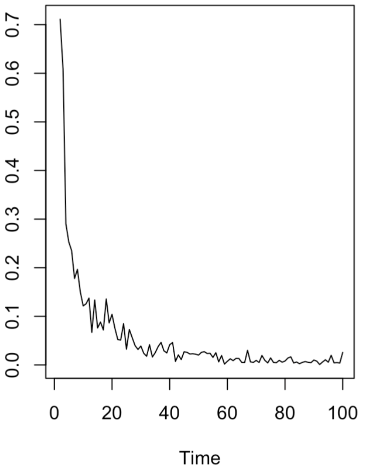
Also, I run an experiment to check the relationship between the running time vs. discount rate. The pattern should be similar to the last figure which means more iterations, more time to run. Similarly, I set the discount rate varies from 0.05 to 0.9 and the y axis is the total running time.



The right figure shows that the overall trend is similar to the expected one. From the right figure, when the discount rate is 0.9 the time spent is 0.0015 second while the minimal running time is around 0.005.

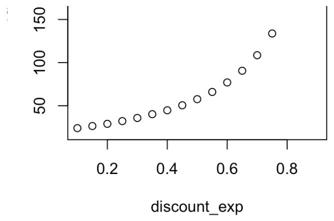
With random initial policy, the iteration usually will converge within 3 times which is faster than the policy-iteration.

2.3 Q - learning



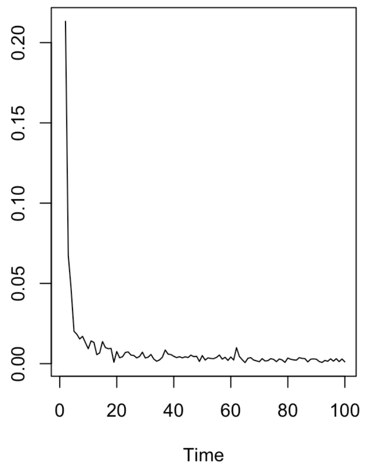
Q-learning was also applied to this problem. The parameter discount was adjusted to check various experiments. The discount varies from 0.05 to 0.9.

Before the experiment, when the discount was set as the 0.9, then the optimal policy was 1 1 2 2 which is "holding, holding, fishing, fishing" which is similar to the value-iteration algorithm.



The first experiment was that the optimal value varies as the discount varies. From the Q-learning, the overall trend was similar to the previous two approaches. However, it seems like when the discount is 0.9, then the optimal value is higher than the previous two approaches.

The next figure shows the discrepancy means over 100 iterations and varies as the iteration went. We can see that as at the beginning, the discrepancy is pretty large and noise is large, then as the iteration continues, the overall mean converges at around 40 to 60. It is a sign of convergency.



Next figure was based on the discount equals to 0.5. In this case, the convergency rate shows faster convergency. This is an interesting finding. The smaller the discount, the faster it converges. I also did experiments on the sequence of discount from 0.1 to 0.9 and similar patterns was shown.

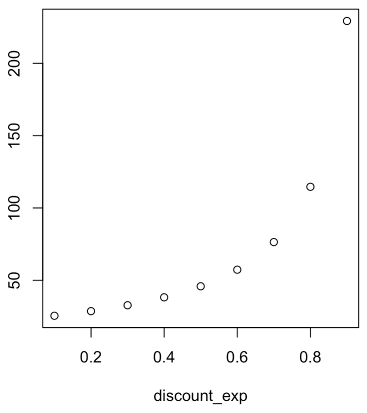
Most of the case, the algorithm would converge around 10 iterations and the convergency could be expressed as the number of running time.

3 Models: large dataset

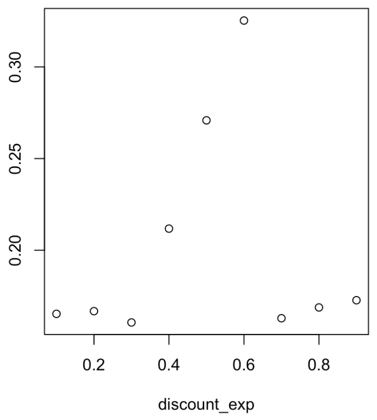
For all experiments below, the data will be generated using the seed = 123 to generate the data in order to make sure the results can be replicated. Then for each scenario, the inference will be all based on this dataset.

3.1 Policy iteration

For the policy iteration with discount varies from 0.1 to 0.9, the optimal value varies and follows the similar patterns as the first dataset. We can see that as the discount increases, the optimal value increases. This is reasonable as the discount increases, the past value will have more value than small discount. Also we can see that the increase seems to follow an exponential distribution and has an exploding trend for the large discount. This pattern also similar to the first dataset.



The next experiment I conducted was to see the overall running time when the discount varies from 0.1 to 0.9. the results are surprising since the pattern is not as what I expected. The longest time it has was not in extreme value of the discount, but in the middle. The right figure shows that when the discount rate is around 0.5-0.6, will have the longest running time, and when the discount is around 0.8, it decreases significantly.

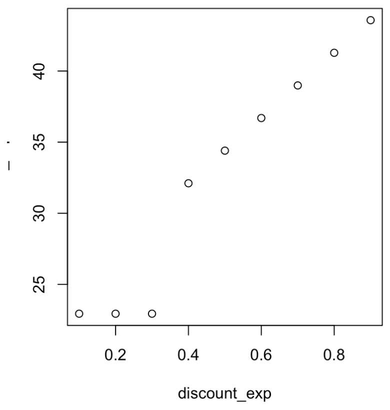


This may also due to the iterations as the larger iteration times, the longer running time. From the results, I saw the similar patterns shows that the iteration at discount value is 0.5 has the largest value. This is consistent with the right figure.

In general, the overall iteration times was around 1500 and much higher than the first dataset due to it complexity. The convergence criterion was similar to the first example.

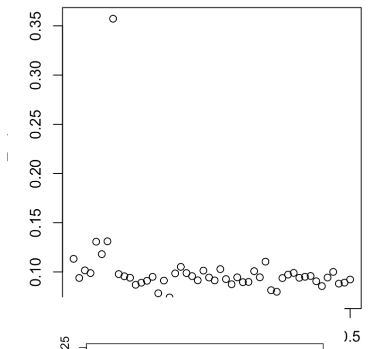
3.2 Value iteration

For the value iteration, the similar simulations were performed to see the results under different scenarios. The first experiment was to check the performance of optimal value conditional on different values of discount varies from 0.1 to 0.9.



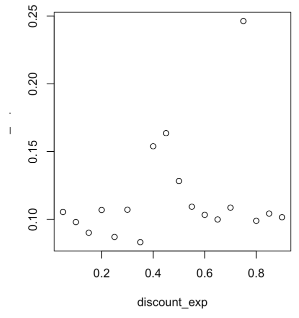
The right figure was a little surprising since it looks like that when the discount is less than 0.3, there is no change for the optimal value, however, when the discount increases starting from 0.4 to 0.9, then we can see that the trend is obvious and not an exponential exploding was observed just like the first dataset which is interesting.

The next scenario I did was conditional on the discount is 0.9, I used different values of epsilon to check the total running time difference. The results as the right figure show that no significant trend can be found through the simulation. Overall the running time was fluctuated around 0.1 and only one outlier was detected at the epsilon around 0.1. This should not be case and the overall trend did not show increase or decrease.



Also the running time of the algorithm did not express a trend. The timing varies from 0.1 s to 0.25 s and did not show the relationship between discount and the running time.

From the right figure we could see that even though there is an extreme value at discount 0.7 but the overall trend is not significant. There is an interesting founding that at the discount value equals 0.4 to 0.5, the running time was the longest. This is similar to the pattern for the first example.

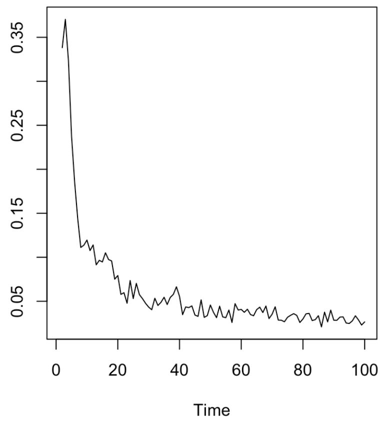


Compared to the previous policy iteration, the value iteration has a faster convergency rate and larger optimal value. Same conclusion as the first dataset.

3.3 Q - learning

Q-learning was also deployed here to explore the optimal algorithms and parameter tuning. However, the running time was too large in the Q-learning algorithm so I spent a long time generating the results.

From the right figure, we could see that the mean discrepancy varies largely before 60 and became stable after 60. Therefore, we could see that the convergence time is around 60 minutes and the iteration times was around 13528. This is too much compared to the previous two approaches given the same dataset.



Further experiments were working on the different epsilon and the discount values. However, due to large computational cost, I did not include the figures/tables in the report.

For Q-learning, it probably will only better on the small dataset since if the states are too large, then it would be pretty hard to find the convergency.

In conclusion, from the above two algorithms, I see that the value iteration works better than the rest two algorithms considering the optimal value, the running time, the iteration times.

However, Q-learning did work well at small dataset while did worst at large dataset. It would take hours to converge. In this case, I would first deploy the value-iteration at the beginning and decide either policy iteration or the Q learning depending on the data size. Also, from both examples, we see that the converged result varies when different algorithms were applied. In that case, I would recommend applying different algorithms if time allows and compare the results using different metrics.