Practical 4: Reinformant learning

Jing Yuan(jyuan@hsph.harvard.edu)

May 6, 2019

1. Game Introduction

In this practice, we need to "teach" the monkey how to pass the tree in the game named "Swingy Monkey Game". Manually, in this game the user control a monkey that is trying to swing on vines and avoid tree trunks, where the gravity and the distance between trees vary from game to game. The reward will be given based on the monkey's most recent action and states, the rules are listed below:

- Reward of $+1$ for passing a tree trunk.
- Reward of -5 for hitting a tree trunk.
- Reward of -10 for falling off the bottom of the screen.
- Reward of -10 for jumping off the top of the screen.
- Reward of 0 otherwise.

So our learning task is to estimate the optimal policy function $\pi : S \to A$ such that the expectation of reward function $R : S \to A$ is maximized. If define a stochastic process of game state $\{s_t\}$ with unknown transition probability $P(s_{t+1}|s_1, ..., s_t, a_1, ..., a_t)$, we aim to identify a π^* such that:

$$\pi^* = \arg\max_{\pi} E\big(\sum_{s\in\mathbf{p}} R(s, \pi(s))|\mathbf{p}\big)$$

where $\mathbf{p}$ is a possible path of $\{S_t\}$

2. Technical Approach

2.1 Model selection rationale

In the previous section we identified below characteristics of the task at hand. So, based on these facts, I assume Markovian property for the process $\{s_t\}$, i.e. $p(s_{t+1}\big|\{s_i\}_{i=0}^t, \{a_i\}_{i=1}^t) = p(s_{t+1}\big|s_t, a_t)$, such that the transition information can be described by a transition matrix $\mathbf{P}$, the limited available information put this problem into a Reinforcement Learning setting. Furthermore, it is not a good idea to deploy model-based approach due to the larger $S \times A$ space. More specifically, since the model-based approach requires reasonably accurate estimation of p ($|S| \times |S| \times 2$ matrix), we need to visit each nontrivial position of P sufficiently often in order to achieve reliable $\hat{P}(s'|s, a)$ estimates. Such computation, even after the discretion of the state space, is intractable in terms of both complexity and storage.

Based on above consideration, I focused on Q-learning in my implementation, where the estimation

task is greatly simplified by considering only the Q matrix. Specifically, by expanding the Bellman equations as following:

$$\begin{aligned} Q(s,a) &= R(s,a) + \gamma \sum p(s'|s,a) max_{a' \in A} Q(s',a') \\ &= R(s,a) + \gamma E_{s'}[max_{a' \in A} Q(s',a')] \\ &= E_{s'}[R(s,a) + \gamma max_{a' \in A} Q(s',a')] \end{aligned}$$

and estimation may proceed through Temporal Difference Learning as below updating algorithm:

$$Q(s,a)^{new} \leftarrow Q(s,a)^{old} + \alpha[r + \gamma max'_a Q(s',a')^{old} - Q(s,a)^{old}]$$

where $\gamma \in (0,1)$ denotes the discount factor, and $\alpha \in (0,1)$ denotes the learning rate.

1) Simple Q-learning

Specifically, I used σ-greedy algorithm for choosing the new action, that is, at each time t, I chose a random action as our next move with probability σ, and I chose an action that has the highest Q value from the Q table with probability $(1-\sigma)$. Tuning the value of σ changes the exploitation and exploration ratio. Based on the non-change learning rate model, I decided to change the learning rate α to 0.1 original α if the number of iterations exceed 100. Then, we decided to use σ-greedy factor=0.001 and tune the value of the learning rate (α) and the discount rate (γ), since they determine the convergence speed of the program. In this analysis, I choose a wide range of α from 0.2 to 0.9 and relatively large γ values from 0.6 to 0.9. In the result, I first compared the learning performance with different learning rate and a fixed σ and $\gamma(\gamma = 0.6)$. Then I compared the scores with different discount rates at optimal learning rate ($\alpha = 0.2$) and a fixed σ.

2) Upgraded Q-learning

I also applied more complex method with more customized learning rate value. The α values in this new method depends on how many times the corresponding $Q(s,a)$ has been filled, which means the number of times the monkey finds itself taking action a in state s, denoted as $k(s,a)$. I Set the learning rate for each state equals to $\frac{1}{k(s,a)}$. In this way, the higher the number of times the monkey visit $Q(s,a)$ the lower α will be, and the learning rate will reach the point with the values of k increases leading to convergence of Q.

2.2 Choice of State Space

In the given state, the state and action spaces are defined as:

$$\begin{aligned} S &= [Tree\_dist, Tree\_bot, Tree\_top, Monkey\_vel, Monkey\_top, Monkey\_bot] \subset R^6 \\ A &= [Jump, None] \end{aligned}$$

Note that $[Tree\_top, Tree\_bot, Monkey\_top, Monkey\_bot]$ are in fact bounded by screen size (width:400pxls).Due to the continuous and high-dimensional nature of original S, it is essential to identify S_p a minimal-information-loss projection of S in discrete, lower dimensional space. Specifically, we impose below criteria for our projection $s_p \in S_p$:

1. S_p contains maximal possible information from S for the purpose of optimizing Q;
2. S_p reasonably satisfies Markov assumption, i.e. $P(s_{p,t+1}|s_{p,t}, a_t) = P(s_{p,t+1}|\{s_{p,i}\}_{i=1}^t, \{a_{p,i}\}_{i=1}^t)$

Although above criteria are difficult to verify theoretically, they did provide some intuitive guidance in terms of selecting minimal-information-reduction transformation of S. So, in order to representing a lower-dimension space, in the following model construction process, firstly, I have three strategy for reducing state that will be used in the model (S):

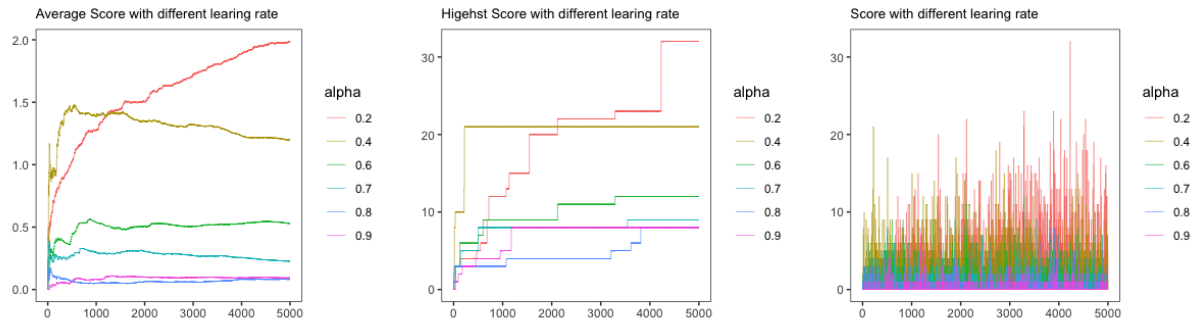
1. 3 states: $\{Tree\_dist, Monkey\_vel, (Monkey\_top - Tree\_top)\}$
2. 4 states: $\{Tree\_dist, Monkey\_vel, (Monkey\_top - Tree\_top), Tree\_bot\}$
3. 3 states+gravity: $\{Tree\_dist, Monkey\_vel, (Monkey\_top - Tree\_top), Gravity\}$

Secondly, I apply discretization method to transfer continuous variables into discrete counterparts and finally create a state space for the learning process which will reduce the iteration times.
In my model learning process, I used the following 4 combination of method and state spaces:**A)** Simple Q-learning +3 states space; **B)** Upgraded Q-learning+3 states space; **C)** Upgraded Q-learning+4 states space; **D)** Upgraded Q-learning+3 states+gravity.

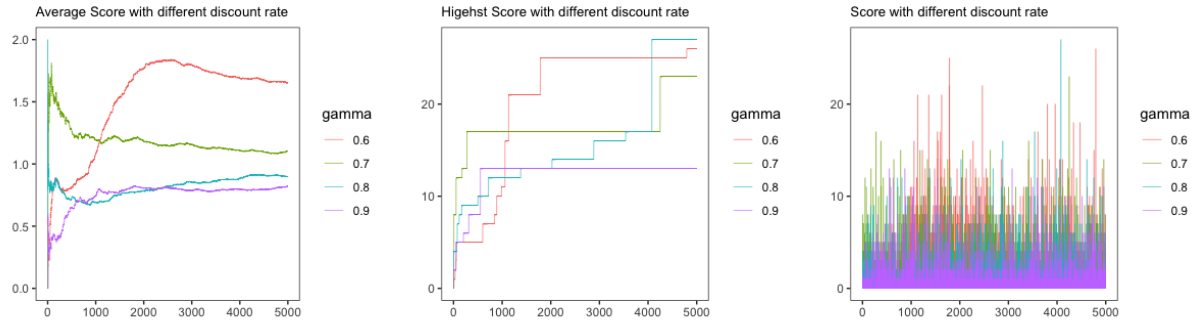
3. Results

3.1 Tuning parameters

The following 3 figure shows the average, highest and raw score applying the combination A with different learning rate α given the other parameters fixed. So, I could conclude that with the lower values of α (0.2), the learning performance is better than it with higher α. This indicates that the states and action in the future has moderate relationship compared to the previous one.



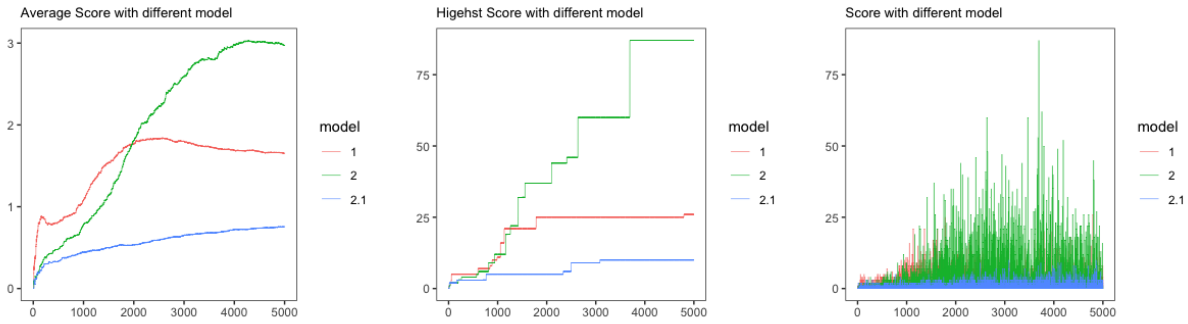
The following 3 figure shows the average, highest and raw score applying the combination A with different discount rate γ with optimal $\alpha = 0.2$ given the other parameters fixed. So, I could conclude that with the lower values of γ (0.6), the learning performance is better than it with higher γ. Thus, I used $\alpha = 0.2$ and $\gamma = 0.6$ for the following learning.



3.2 Different Q-learning method

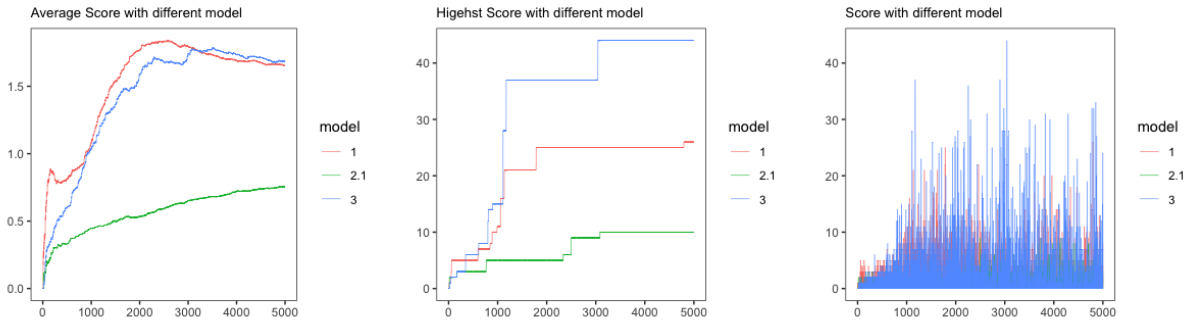
The following 3 figure shows the average, highest and raw score applying the combination B and C with optimal parameters($\alpha = 0.2$, $\gamma = 0.6$ and $\sigma = 0.001$). In the figure, Model 1 represent simple

Q-learning with 3 state spaces; Model 2 represent upgraded Q-learning with 3 state spaces; Model 2.1 represent upgraded Q-learning with 3 state spaces. Model 2 has better learning performance compared to model 1, but Model 2.1 has lower score compared to model 1 and 2, which may due to larger states spaces. If applied more iteration, we may see better performance.



3.3 Considering gravity

The following 3 figure shows the average, highest and raw score applying the combination D with optimal parameters($\alpha = 0.2$, $\gamma = 0.6$ and $\sigma = 0.001$). Since, the gravity changes every time when we run the game, so it is necessary to include gravity into state spaces. Model 3 represent upgraded Q-learning model with 3 states space including gravity. Model 3 has the highest score compared to other model, however with relative same average score as model 1 which may due to the enlarged state spacse. But, Model 3 perform better than Model 2.1 in both average and highest scores.



4. Discussion

To improve this learning process, there are a lot of ways to deal with it. The most effect way is to redefine state spaces. For example, one method has effectively dealt with practical situations with large state spaces is locally weighted regression (LWR). It create a local model of a function around a query point. To deal with the large state spaces, besides discretizing the position space into bins and locally weighted regression, another strategy is implementing function approximation, such as neural network. Another way to find the optimal method is to increase the number of iterations. Because, the more states the model has experienced or the more Q-matrix values to be filled, the more complete the matrix is. We will get a better model.