Project3 - Blackjack

LiuBaichuan
School of Data Science
Fudan University
16307130214

May 18, 2019

1 Value Iteration

1.1

Normal **MDP** consists of initial states(S_0),actions,transition model $P(s\prime|s,a)$ and reward R(s). However, some MDPs allow to definite reward by action and successor, so the reward function is R(s,a,$s^{'}$). In our game, we exactly use the latter.

So we change Bellman Update Equation $V_{k+1}(s) \leftarrow R(s)+\gamma \max_{a\in A(s)} \sum_{s'} P(s^{'}|s,a)V_k(s^{'})$ to $V_{k+1}(s) \leftarrow \max_{a\in A(s)} \sum_{s'} T(s,a,s^{'})[R(s,a,s^{'})+\gamma V_k(s^{'})]$

0 iteration

$$v(-2)=0\ v(-1)=0\ v(0)=0\ v(1)=0\ v(2)=0$$

1 iteration

$$\begin{gathered} v(-2)=0 \\ v(-1)=\max(0.8(20+0)+0.2(-5+0),0.7(20+0)+0.3(-5+0))=15 \\ v(0)=\max(0.8(-5+0)+0.2(-5+0),0.7(-5+0)+0.3(-5+0))=-5 \\ v(1)=\max(0.8(-5+0)+0.2(100+0),0.7(-5+0)+0.3(100+0))=26.5 \\ v(2)=0 \end{gathered}$$

2 iteration

$$\begin{gathered} v(-2)=0 \\ v(-1)=\max(0.8(20+0)+0.2(-5-5),0.7(20+0)+0.3(-5-5))=14 \\ v(0)=\max(0.8(-5+15)+0.2(-5+26.5),0.7(-5+15)+0.3(-5+26.5))=13.45 \\ v(1)=\max(0.8(-5-5)+0.2(100+0),0.7(-5-5)+0.3(100+0))=23 \\ v(2)=0 \end{gathered}$$

1.2

After value iteration converges, we get non terminal states' optimal policy. $\pi_{opt}(-1) = -1, \pi_{opt}(0) = +1, \pi_{opt}(1) = +1$

2 Transforming MDPs

2.1

This modified transition function aims to balance transition probabilities, so it is easy to construct a counterexample. Let game has high probability to gain low reward, while low probability to gain high reward. Please check my code in detail.

2.2

In original value iteration method, we have to do multiple iterations because we try to use all the successors from last time slice. However, if we have specific order over states, we can just run an acyclic MDP without visiting a state second time. We construct a tree to store the order of states, by doing this, we can simply compute $V(s)$ using its specific successors $V(s^{'})$. We only go over each $(s, a, s^{'})$ once.

2.3

$$V_{k+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} T(s, a, s^{'})[R(s, a, s^{'}) + \gamma V_k(s^{'})] =$$
$$V_{k+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} \gamma T(s, a, s^{'})[\tfrac{1}{\gamma} R(s, a, s^{'}) + V_k(s^{'})]$$

In order to keep correctness of probability, we use new state o.

$$\begin{aligned} T_{new}(s, a, s^{'}) &= \gamma T_{old}(s, a, s^{'}) \\ R_{new}(s, a, s^{'}) &= \tfrac{1}{\gamma} R_{old}(s, a, s^{'}) \\ T_{new}(s, a, o) &= 1 - \textstyle\sum T_{new}(s, a, s^{'}) \\ R_{new}(s, a, o) &= 0 \\ T_{new}(0, a, 0) &= 1 \\ T_{new}(0, a, s) &= 0 \end{aligned}$$

3 Peeking Blackjack

3.1

Check my code in detail.

3.1.1

My idea is making a high probability of exceeding the threshold forces players to peek. Check my code in detail.

4 Learning to Play Blackjack

4.1

Check my code in detail.

4.2

In small MDP, value iteration has 5 iterations and the policy learned has 96 percent as the same with the policy learned by value iteration. In large MDP, value iteration has 16 iterations and the policy learned has 68 percent as the same with the policy learned by value iteration. In small MDP, rl method is capable to explore enough states, while in large MDP, it is difficult to learn all states, so it has low similarity with value iteration. Also, *identityFeatureExtractor* can only return a singleton list containing indicator feature for the (state,action) pair and it provides no generalization.

4.3

Check my code in detail.

4.4

OriginalMDP game settings:$cardValues = [1, 5], multiplicity = 2, threshold = 10, peekCost = 1$

I get relatively low sum rewards 205103 for FixedRLAlgorithm when I use the policy learned for originalMDP in it. We also can observe each reward is below 10, because it can not beyond original threshold. Since FixedRLAlgorithm can not adapt, the actions taken are not optimal actions for newThresholdMDP.

newThresholdMDP game setting:$cardValues = [1, 5], multiplicity = 2, threshold = 15, peekCost = 1$ Qlearning has higher sum rewards 288771 because it is able to adapt to newThresholdMDP.