Project 2 Blackjack

1 Problem 1: Value Iteration

1.1 1a 1b

The MDP model is:

- State: $S = \{-2, -1, 0, 1, 2\}$
- Action: $A = \{+1, -1\}$
- Discount: γ=1
- Transition: $T = \left[\begin{array}{ccc} & S-1 & S+1 \\ +1 & 0.7 & 0.3 \\ -1 & 0.8 & 0.2 \end{array}\right]$
- Reward: {-2:20, -1:-5, 0:-5, 1:-5, 2:100}

Calculation process:

$$\begin{aligned} V_1^*(-1) &= \max\{+1 : 0.7 * (0 + 20) + 0.3 * (0 - 5), \quad -1 : 0.8 * (0 + 20) + 0.2 * (0 - 5)\} \\ &= \max\{+1 : 12.5, \quad -1 : 15\} \\ &= 15(-1) \end{aligned}$$

$$\begin{aligned} V_1^*(0) &= \max\{+1 : 0.7 * (0 - 5) + 0.3 * (0 - 5), \quad -1 : 0.8 * (0 - 5) + 0.2 * (0 - 5)\} \\ &= -5(+1) \end{aligned}$$

$$\begin{aligned} V_1^*(1) &= \max\{+1 : 0.7 * (0 - 5) + 0.3 * (0 + 100), \quad -1 : 0.8 * (0 - 5) + 0.2 * (0 + 100)\} \\ &= \max\{+1 : 26.5, \quad -1 : 16\} \\ &= 26.5(+1) \end{aligned}$$

$$\begin{aligned} V_2^*(-1) &= \max\{+1 : 0.7 * (0 + 20) + 0.3 * (-5 - 5), \quad -1 : 0.8 * (0 + 20) + 0.2 * (-5 - 5)\} \\ &= \max\{+1 : 11, \quad -1 : 14\} \\ &= 14(-1) \end{aligned}$$

$$\begin{aligned}V_2^*(0) &= \max\{+1 : 0.7 * (15 - 5) + 0.3 * (26.5 - 5), \quad -1 : 0.8 * (15 - 5) + 0.2 * (26.5 - 5)\} \\ &= \max\{+1 : 13.45, \quad -1 : 12.3\} \\ &= 13.45(+1)\end{aligned}$$

$$\begin{aligned}V_2^*(1) &= \max\{+1 : 0.7 * (-5 - 5) + 0.3 * (0 + 100), \quad -1 : 0.8 * (-5 - 5) + 0.2 * (0 + 100)\} \\ &= \max\{+1 : 23, \quad -1 : 12\} \\ &= 23(+1)\end{aligned}$$

The final result is as follow:

Table 1: Value Iterations (Optimal Policy)

State(Policy)	**-2**	**-1**	**0**	**1**	**2**
0	0 (None)	0 (None)	0 (None)	0 (None)	0 (None)
1	0 (None)	15 (-1)	-5 (+1)	26.5 (+1)	0 (None)
2	0 (None)	14 (-1)	13.45 (+1)	23 (+1)	0 (None)

2 Problem 2: Transforming MDPs

2.1 2a

The MDP model we constructed is:

- State: $S = \{-1, 0, 1\}$, goal={-1,1}
- Action: $A = \{+1, -1\}$
- Discount: γ=1
- Transition: $T = \begin{bmatrix} & S-1 & S+1 \\ +1 & 0.9 & 0.1 \\ -1 & 0.9 & 0.1 \end{bmatrix}$, and $T'(s, a, s')$ is defined as in the problem.
- Reward: {-1:-1, 0:0, 1:1}

Calculation process:

$$V_1'(0) = 0.9 * (0 + -1) + 0.1 * (0 + 1) = -0.8$$

$$V_2'(0) = 0.5 * [0.9 * (0 + -1) + 0.1 * (0 + 1)] + 0.5 * 0 = -0.4$$

So we have $V_1'(0) < V_2'(0)$.

2.2 2b

For acyclic MDPs, since the calculation for $V(s)$ only depends on $V(s')$, the values of the states that will come after s, we just need to do one iteration provided that we process the nodes in reverse topological order of the graph.

To be more specific, the algorithm is divided into following steps:

1. Find the topological order of the states, say, using modified DFS algorithm.
2. Traverse the states according to the reversed topological order, and compute $V(s)$ for each state with the formula
$$V(s) = \max_{a\in A}\sum_{s'} T(s,a,s')[V(s') + R(s,a,s')]$$

In this way, for every state, we are only computing its value once.

2.3 2c

Set the new state o as an end state, then $V'(o) = V_{opt}(o) = 0$. The current MDP model is:

- State: States'=States $\bigcup\{o\}$
- Action: Actions'(s)=Actions(s), Actions'(o) = None (end state)
- Discount: γ'=1
- Transition: $T'(s,a,s') = \begin{cases} \gamma T(s,a,s') & s \in \text{States} \\ 1-\gamma & s = o \end{cases}$
- Reward: Reward'(s,a,s')=$\begin{cases} \frac{1}{\gamma}\text{Reward}(s,a,s') & s \in \text{States} \\ 0 & s = 0 \end{cases}$

Then check the bellman function:

$$\begin{aligned} V_{opt}^{'} &= \max_{a\in\text{Actions'(s)}} \sum_{s'\in\text{States'}} T'(s,a,s')[V'(s') + \text{Reward'}(s,a,s')] \\ &= \max_{a\in\text{Actions(s)}} \sum_{s'\in\text{States}} \gamma T(s,a,s')[V'(s') + \frac{1}{\gamma}\text{Reward}(s,a,s')] + (1-\gamma)[V'(o)+0] \\ &= \max_{a\in\text{Actions(s)}} \sum_{s'\in\text{States}} T(s,a,s')[\gamma V'(s') + \text{Reward}(s,a,s')] \end{aligned}$$

Which has the exactly same formula as in original model except for $V'(s')$. But this equivalence is held from the beginning of the algorithm, we conclude $V'(s) = V(s)$ and thus $V_{opt}(s)$ $\forall s \in$ States are equal under the original MDP and the new MDP.

3 Problem 4: Learning to Play Blackjack

3.1 4b

I've tried several experiments and get results as follow:

Table 2: Comparison Results

Q-learning	Similarity	
Experiments	small MDP	large MDP
1	96.30%	68.42%
2	100.00%	68.63%
3	100.00%	67.50%
4	88.89%	66.41%
5	96.30%	67.98%

For smallMDP, the difference is about 3.7%, while for largeMDP, the difference is around 32% (Results may fluctuate due to randomness).

One of the problems is that our value approximation is too much simple, that is, the implementation of $featureExtractor$ assigns each $(state, action)$ pair a specific weight, resulting in a large amount of parameters to be estimated. This won't become a problem once we simulate enough times according to the total number of states, but when the number of states increases, poor estimation will slow down the convergence.

For smallMDP trails it works well, because the total number of states is quite small, say, 27 different states in this case. Then the probability that $Q - learning$ will traverse all (or most) possible states is relatively large even we select actions with $\epsilon - greedy$ method, a seemingly random way. Therefore the final values we get are rather similar to $Value\ Iteration$, since this algorithm will converge in the specific case.

However, since the total number of states for largeMDP is 2745, it is hard for us to cover all cases. Therefore, due to the randomness of the $Q - learning$, we will only visit part of the states and this will give bad estimations of the $Q_{opt}(s, a)$.

During the experiment, we've set the number of trials to 30,0000, and find the similarity between $Value\ Iteration$ and $Q - learning$ keeps going down. However, with the knowledge that "the more samples will generate more accurate estimation", we think this phenomenon doesn't necessarily indicate the poor performance of $Q - learning$.

3.2 4d

I've tried several experiments and get results as in the table.

Table 3: Comparison Results

NumTrials	300 Trails		30000 Trails	
Experiments	Value Iteration	Q-learning	Value Iteration	Q-learning
1	6.96	7.58	6.84	9.53
2	6.73	8.62	6.83	9.52
3	6.80	7.43	6.84	9.56
4	6.86	**6.15**	6.85	9.51
5	6.87	9.14	6.84	9.56

The rewards under $Value\ Iteration$ is about 6.9 when take 300 and 30000 trials, while the average rewards under $Q-learning$ increases from 7.58 (300 trials) to 9.53 (30000 trials).

We note that on average, we get the lower expected(average) reward for $Value\ Iteration$ and compared to the reward for $Q-learning.$ However, when the number of trials is small, the reward for $Q-learning$ isn't always guaranteed.

On one hand, for $Value\ Iteration$, this is equivalent to an off-policy case with respect to value iteration implemented on original mdp problem and then get the rewards on adjusted mdp problem. This may cause problems that the optimal policies for original MDP can't handle the new MDP, since $FixedRLAlgorithm$ won't adapt to the new situations. For $Q-learning$, this won't happen since each iteration it will update the weights of different features during the simulation, which means it will adapt to new problems in time.

On the other hand, due to the randomness included in $Q-learning$, we have to take enough trials to get more samples to estimate the parameters for more accurate results.