Report Of Q-learning Program

Author: Lingbo Tang, Yufei Zhang

Introduction:

This project is aimed at using Q-learning to derive the best game policy to win the black jack poker game. Blackjack, also known as twenty-one, is the most widely played casino banking game in the world. The rules to win is: 1. Get 21 points on the player's first two cards (called blackjack), without a dealer blackjack. 2. Reach a final score higher than the dealer without exceeding 21. 3. Let the dealer draw additional cards until his or her hand exceeds 21.[1] Since the card you have been dealt is random and the actions taken by both dealer and customer are also random, so we can learn the black jack game as a Markov Decision process. We are using Q-learning algorithm (off-policy) throughout the whole project. Therefore, for each run, we will run at least 1.0*e+06 times to derive an approximately deterministic policy.

Procedures and Results:

1. Specified settings:

When α =0.001 and ϵ = 1 and episode =1.0*e+06, we get the average return as -0.306911, this value is approximately equal to the average return that we get from the random policy. When α =0.001 and ϵ = 0.1 and episode =1.0*e+06, we get the average return as -0.064448, this value is much better than the value we derive from Q-learning algorithm when ϵ = 1, which means the epsilon greedy policy we learned from Q-learning algorithm is much better than the random policy.

2. Finding the best policy:

Now we have to use the variable control method to find the best policy since we have 3 different variables that has effect on the policy. When we want to test the effect of ϵ, we have to make the number of episode and the step size all the same but just change the value of ϵ. When we are testing other two variables, the rules is the same as what we stated above. It's obvious that larger number of episodes means better number that is converged to the best value. Then most significantly, we should do experiment for ϵ and α, and always make the number of episodes as 1.0*e+06.

Here is what we got when α =0.001 and ϵ = 0.1, α =0.001 and ϵ = 0, α =0.001 and ϵ = 0.01:

alpha	epsilon	episodes	Return	ave-return
0.001	0.1	1000000	-0.064743 -0.064304 -0.066059 -0.063826 -0.064284 -0.062602 -0.066652 -0.062968 -0.064185 -0.064853	-0.0644476

alpha	epsilon	episodes	Return	ave-return
0.001	0	1000000	-0.08435 -0.064923 -0.063971 -0.069449 -0.065586 -0.071614 -0.083696 -0.077429 -0.074136 -0.067007	-0.0722161

alpha	epsilon	episodes	Return	ave-return
			-0.055398	
			-0.047165	
			-0.054625	
			-0.051072	
0.001	0.01	1000000	-0.052443	-0.0525088
			-0.053277	
			-0.052508	
			-0.050369	
			-0.055341	
			-0.05289	

From these results sets we can see that when α does not change and episodes equals to 1.0*e+06, the best policy should be generated when epsilon is approximately equal to 0.01.

When we are exploring the effect of α (step size), we follow the same method that we used above. As well, when executing the experiment, we use the same method above. Instead of doing sequential search all the time, we use golden section search to increase the speed of exploring. Sample data will be displayed by the following chart:
Comparing the result when alpha and epsilon stays the same but the episodes increased from 1.0*10^6 to 1.0*10^7 (Chart in next page):

alhpa	epsilon	tries	result	ave-result
0.001	1/n	1000000	-0.050355 -0.046708 -0.047359 -0.047214 -0.048419 -0.04583 -0.047377 -0.047799 -0.048798 -0.048798	-0.0478657

Where n is the number of episodes accumulated from 1 to the total episodes executed.

alhpa	epsilon	tries	result	ave-result
0.001	1/n	10000000	-0.0376685 -0.0366644 -0.0378312 -0.0381894 -0.0368596 -0.0367296 -0.0379569 -0.0370632 -0.0378986 -0.0366743	-0.0373536

Comparing the result when epsilon and the:
(Since 1.0*10^7 is too large, we don't have enough time to run, we just show one significant sample compare)
When alpha = 0.001 and epsilon = 1/n, and episodes = 1.0*10^7:
ave-return = -0.0373536
When alpha = 0.01 and epsilon = 1/n, and episodes = 1.0*10^7
ave-return = -0.0408879

Analysis:

We observed that if $\alpha \in (0.0005, 0.001)$, the converge process will be safe and fast, when the step size is larger than the upper bound or lower than the lower bound the policy might not be perfect. It means if the step size is too large, the error caused by each step is large, and the effect will accumulate while more number of episodes are learned by us. If the step size is too small, it will cause more error on each step, and we call it hysteresis effect.
While we are exploring the effect of ϵ, we found that if ϵ is changing with the number of episodes executed such as (1/number of episodes), it will produce the best policy most of the time if the episodes is large enough, ex. episodes>2.5*10^6. It can be explained by: Once we get enough experience from the model, we don't need to explore as frequently as before, we should exploit the best policy that we have been explored. Then the smaller epsilon, the more chance we can select the best policy. However, when the episodes are less than 2.5*10^6, the epsilon might be too large which can involve worse policy.
For the policy that we explored by Q-learning, it make sense in the practical scenario. For example, when the customer gets total card 12, and the dealer gets total card 4, the customer should stick to its card and wait for dealer explode. When customer gets total card 20, whatever the dealer gets the customer should stick to its card, and compare it to the dealer's card

Conclusion:

If we set α = 0.001, ϵ = 1/(number of episodes) and number of episodes greater than 1.0*10^7, we can get the best policy as follow:

Average return: -0.0365098

Usable Ace:
S S S S S S S S S S 20
S H S H H H S S S S 19
H S S H H H S S H H 18
H H H H H H H H H H 17
H H H H H H H H H H 16
H H H H H H H H H H 15
H H H H H H H H H H 14
H H H H H H H H H H 13
H H H H H H H H H H 12
1 2 3 4 5 6 7 8 9 10

No Usable Ace:
S S S S S S S S S S 20
S S S S S S S S S S 19
S S S S S S S S S S 18
S S S S S S S S S S 17
H S S S S S H H S S 16
H S S S S S H H H H 15
H S S S S S H H H H 14
H S S S S S H H H H 13
H S S S S S H H H H 12
1 2 3 4 5 6 7 8 9 10

But it means, if we want to win the game, we should always be the dealer or never try too many times.

Reference:

[1] http://en.wikipedia.org/wiki/Blackjack