**[172158] Ali Almajed**

CS 229 Machine Learning, spring 2020

Homework 5:

Reinforcement Learning

Due Saturday May 9, 11:59pm

**Question1: (3 pts) Difference between TD and MC**

Read the attached TD-MC chapter, and the example 6.1 Driving to home. Can you imagine a scenario in which a TD update would be better on average than an Monte Carlo update? Give an example scenario—a description of past experience and a current state—in which you would expect the TD update to be better.

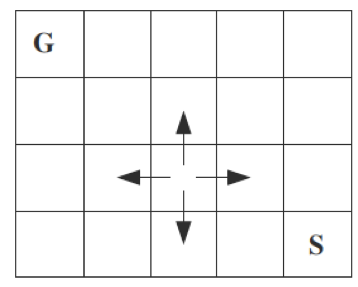
I believe TD helped me a lot when cooking cake. When I cook cake, especially when using a new oven, or new recipe, I do not know how long each step will take and when it will be done, but I keep doing multiple tests to the dough and the toothpick test during the cooking, which is inserting a toothpick in the cake, and removing it. So, when I make dough, I check every few minutes to check how much it rose so far, and adjust my guess on when it will be ready. Furthermore, during the cooking process, I do the toothpick test, and if there is cake batter stuck to the toothpick, then the cake is not ready and I adjust the completion time accordingly, otherwise it is ready.

So while the cake usually takes about 15-25 minutes, I would start inserting toothpicks from minute 15, every couple of minutes and I would adjust the expected completion time based on how much batter is stuck to the toothpick. By doing these tests, in TD fashion, I keep adjusting little by little until the cake is perfectly cooked.

On the other hand, if I were to use the monte carlo method, I would test but not make a decision until minute 25 to update my estimated time, and by then, the cake would be burned maybe. So in the cake scenario, I believe TD is better than monte carlo.

**Question2: (7 pts) Off policy Q-learning and Sarsa algorithm**

Given the grid world in Figure 1, there are 4 deterministic actions: *up*, *down*, *left* and *right*. The goal is to reach the G, starting at S.



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (0, 0) | (0, 1) | (0, 2) | (0, 3) | (0, 4) |
| (1, 0) | (1, 1) | (1, 2) | (1, 3) | (1, 4) |
| (2, 0) | (2, 1) | (2, 2) | (2, 3) | (2, 4) |
| (3, 0) | (3, 1) | (3, 2) | (3, 3) | (3, 4) |

Figure 1 Grid world

The reward on reaching on the goal (G) is 10.

The reward on actions that would take the agent off the grid is -1 (agent stays still in this case).

The reward on entering F is -5

The reward on other actions is 0.

The discount factor γ = 0.9.

If you need, take a look at Sarsa-OffPolicy-Q-learning.pdf in the attached file.

Use off policy **Q learning** to learn the optimal values of Q\* (s, a). Please submit your own code on calculating the Q\*(s,a).

Code is attached to the submission

--1 (1pts) What is the Q\*(s,a) for each pair of s and a?

Q\* Summary:

Q\*((0, 0)) => Up: 0.00, Right: 0.00, Down: 0.00, Left: 0.00

Q\*((0, 1)) => Up: 8.00, Right: 8.10, Down: 8.10, Left: 10.00

Q\*((0, 2)) => Up: 7.10, Right: 7.29, Down: 7.29, Left: 9.00

Q\*((0, 3)) => Up: 6.29, Right: 6.56, Down: 1.56, Left: 8.10

Q\*((0, 4)) => Up: 5.56, Right: 5.56, Down: 5.90, Left: 7.29

Q\*((1, 0)) => Up: 10.00, Right: 8.10, Down: 8.10, Left: 8.00

Q\*((1, 1)) => Up: 9.00, Right: 7.29, Down: 7.29, Left: 9.00

Q\*((1, 2)) => Up: 8.10, Right: 1.56, Down: 6.56, Left: 8.10

Q\*((1, 3)) => Up: 7.29, Right: 5.90, Down: 5.90, Left: 7.29

Q\*((1, 4)) => Up: 6.56, Right: 4.90, Down: 5.31, Left: 1.56

Q\*((2, 0)) => Up: 9.00, Right: 7.29, Down: 7.29, Left: 7.10

Q\*((2, 1)) => Up: 8.10, Right: 6.56, Down: 6.56, Left: 8.10

Q\*((2, 2)) => Up: 7.29, Right: 5.90, Down: 5.90, Left: 7.29

Q\*((2, 3)) => Up: 1.56, Right: 5.31, Down: 5.31, Left: 6.56

Q\*((2, 4)) => Up: 5.90, Right: 4.31, Down: 4.78, Left: 5.90

Q\*((3, 0)) => Up: 8.10, Right: 6.56, Down: 6.29, Left: 6.29

Q\*((3, 1)) => Up: 7.29, Right: 5.90, Down: 5.56, Left: 7.29

Q\*((3, 2)) => Up: 6.56, Right: 5.31, Down: 4.90, Left: 6.56

Q\*((3, 3)) => Up: 5.90, Right: 4.78, Down: 4.31, Left: 5.90

Q\*((3, 4)) => Up: 5.31, Right: 3.78, Down: 3.78, Left: 5.31

--2 (1pts) What is the V\*(s) for each s?

V\* Summary:

[[ 0. 10. 9. 8.1 7.29]

[10. 9. 8.1 7.29 6.56]

[ 9. 8.1 7.29 6.56 5.9 ]

[ 8.1 7.29 6.56 5.9 5.31]]

--3 (1pts) What are the actions of optimal policy?

from (3, 4) go UP to (2, 4) - Q\* = 5.31

from (2, 4) go UP to (1, 4) - Q\* = 5.90

from (1, 4) go UP to (0, 4) - Q\* = 6.56

from (0, 4) go LEFT to (0, 3) - Q\* = 7.29

from (0, 3) go LEFT to (0, 2) - Q\* = 8.10

from (0, 2) go LEFT to (0, 1) - Q\* = 9.00

from (0, 1) go LEFT to (0, 0)

[['G' 'L' 'L' 'L' 'L']

['U' 'U' 'U' 'F' 'U']

['U' 'U' 'U' 'L' 'U']

['U' 'U' 'U' 'U' 'U']]

Use **Sarsa** algorithm to learn the optimal values of Q\* (s, a). Please submit your own code on calculating the Q\*(s,a).

--1 (1pts) What is the Q\*(s,a) for each pair of s and a?

Q\* Summary:

Q\*((0, 0)) => Up: 0.00, Right: 0.00, Down: 0.00, Left: 0.00

Q\*((0, 1)) => Up: 8.00, Right: 8.10, Down: 8.10, Left: 10.00

Q\*((0, 2)) => Up: 5.48, Right: 0.28, Down: 5.90, Left: 9.00

Q\*((0, 3)) => Up: 0.20, Right: 0.22, Down: -3.78, Left: 8.10

Q\*((0, 4)) => Up: -0.48, Right: -0.48, Down: 0.09, Left: -3.41

Q\*((1, 0)) => Up: 10.00, Right: 8.10, Down: 6.56, Left: 8.00

Q\*((1, 1)) => Up: 9.00, Right: 7.29, Down: 7.29, Left: 9.00

Q\*((1, 2)) => Up: 5.31, Right: 1.56, Down: 5.31, Left: 8.10

Q\*((1, 3)) => Up: 0.47, Right: 0.03, Down: 1.40, Left: 3.87

Q\*((1, 4)) => Up: 6.56, Right: -0.95, Down: -4.02, Left: -3.78

Q\*((2, 0)) => Up: 7.20, Right: 7.29, Down: 5.90, Left: 7.10

Q\*((2, 1)) => Up: 8.10, Right: 1.26, Down: 5.31, Left: 6.56

Q\*((2, 2)) => Up: 7.29, Right: 1.14, Down: 4.78, Left: 5.90

Q\*((2, 3)) => Up: -3.74, Right: -0.20, Down: 0.70, Left: 6.56

Q\*((2, 4)) => Up: -3.62, Right: -2.52, Down: -3.62, Left: -4.47

Q\*((3, 0)) => Up: 6.56, Right: 1.90, Down: 1.88, Left: 1.88

Q\*((3, 1)) => Up: 2.54, Right: 3.24, Down: 2.60, Left: 1.69

Q\*((3, 2)) => Up: 6.56, Right: 4.30, Down: 4.90, Left: 3.60

Q\*((3, 3)) => Up: 3.87, Right: 3.87, Down: 0.78, Left: 4.78

Q\*((3, 4)) => Up: -4.02, Right: -1.70, Down: -0.80, Left: 4.30

--2 (1pts) What is the V\*(s) for each s?

V\* Summary:

[[ 0. 10. 9. 8.1 0.09]

[10. 9. 8.1 3.87 6.56]

[ 7.29 8.1 7.29 6.56 -2.52]

[ 6.56 3.24 6.56 4.78 4.3 ]]

--3 (1pts) What are the actions of optimal policy?

from (3, 4) go LEFT to (3, 3) - Q\* = 4.30

from (3, 3) go LEFT to (3, 2) - Q\* = 4.78

from (3, 2) go UP to (2, 2) - Q\* = 6.56

from (2, 2) go UP to (1, 2) - Q\* = 7.29

from (1, 2) go LEFT to (1, 1) - Q\* = 8.10

from (1, 1) go LEFT to (1, 0) - Q\* = 9.00

from (1, 0) go UP to (0, 0) - Q\* = 10.00

[['G' 'L' 'L' 'L' 'D']

['U' 'L' 'L' 'F' 'U']

['R' 'U' 'U' 'L' 'R']

['U' 'R' 'U' 'L' 'L']]

Please discuss your observed difference between Sarsa algorithm and off-policy Q-learning (1pts) in your game.

Sarsa sometimes is able to converge to a solution faster than the off-policy Q-learning. Furthermore, the results are variable each time because the algorithm uses estimates for both a and a’. Which is the essence of TD, implied by combining the positive sampling of Monte Carlo, and the bootstrapping of DP. Furthermore, in this scenario, Qlearn off-policy provided the direct optimal path, where on-policy SARSA tried to provide a path that avoids going near the fail state. That being said, I think for a simple small game like this one, it is more efficient to go with the q-learning off-policy since the goal is to reach the goal state, and there is no reward for getting there in the fastest way. Also, since there is no step penalty, or reward, the algorithm will have a constant runtime to converge with a reasonable solution.