2018-1 Machine Learning Final Exam (Due: Dec. 19)

Dae-Ki Kang

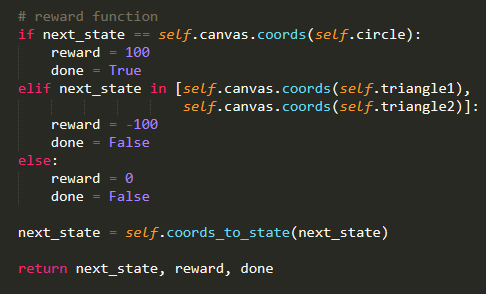
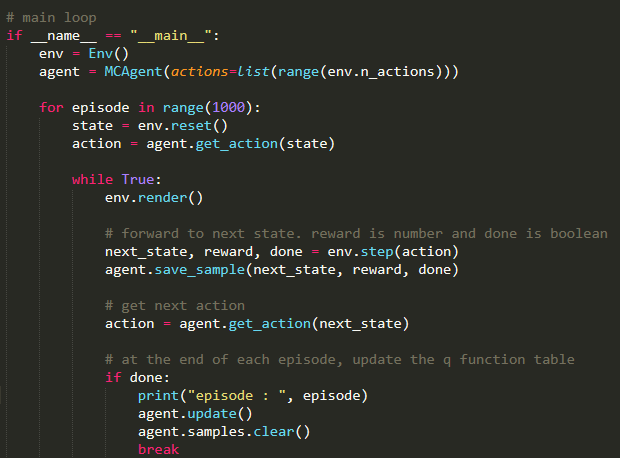
December 4, 2018

Visit and check the following URL. https://github.com/rlcode/reinforcement-learning Setup a github for your project.

1. Run the monte carlo code in the following URL.

https://github.com/rlcode/reinforcement-learning/tree/master/1-grid-world/3-monte-carlo

Analyze the programs (environment.py and mc\_agent.py) and write a report. Add one more triangle with -1 reward at a location you choose. Capture two screenshots of the program execution and include your analysis in the report with the captured images. Place this report in your github.

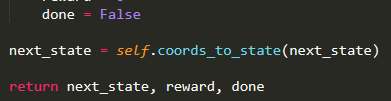
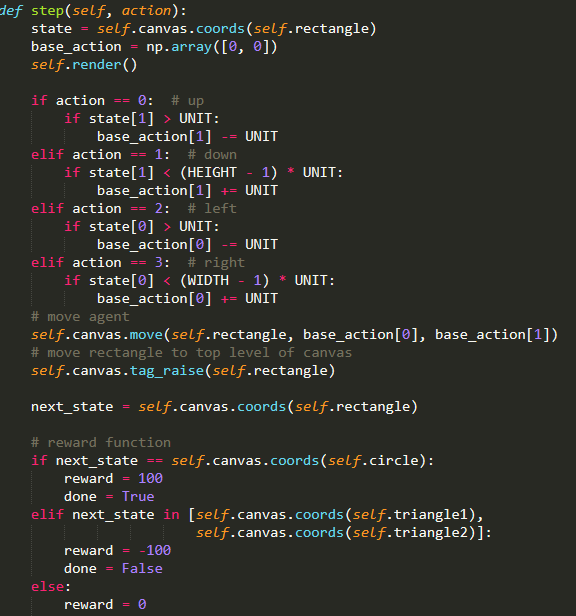


There are two layers loop. Inner loop is infinite, it will be stopped when rectangle find circle (done=ture). In every loop, “next\_sate”, “reward” and “done” will be get frome step(), and then save the three variable, and then get “next\_action” based on get\_action().

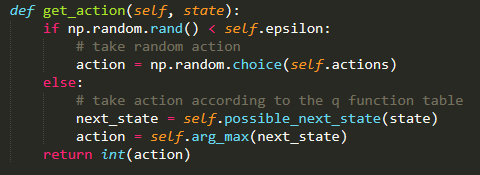
There was an error in environment.py. Where next\_state reach the circle, the reward should be “False” not “Ture”.

The episode of Outside loop is 1000. For every episode, it will be end of finding circle, and then undate the agent.

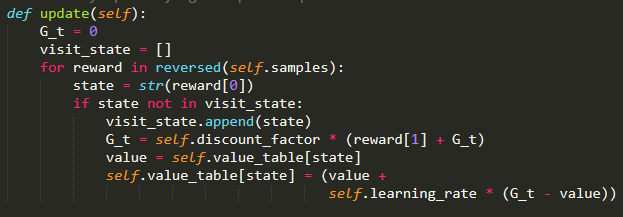
Now look at step() and get\_action() and update().



In this function we will know how to get “next\_sate”, “reward” and “done”. Rectangle moves by certain action( 0 means up, 1 means down, 2 means left, 3 means right). Then we will check the next state, if next state is circle , “done” will be “Ture”, else will be “False”.



If np.random.rand() < self.epsilon( epsilon=0.1), action will choose arbitrary action. Epsilon gives agent chance to have other choices. But in greater possibility , the agent will find next state in greedy policy.



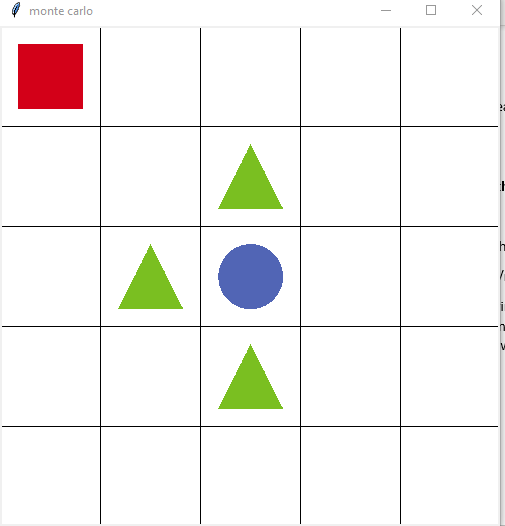
The update rule will be:

G\_t=λ(Reward(1)+G\_t

V=V+α(G\_t-V)

λ is the discount, α is the learning rate. G is the goal, V is the value. Rerward[0] is state, reward[1] is reward.

**Add one more triangle with -100 reward at a location:**

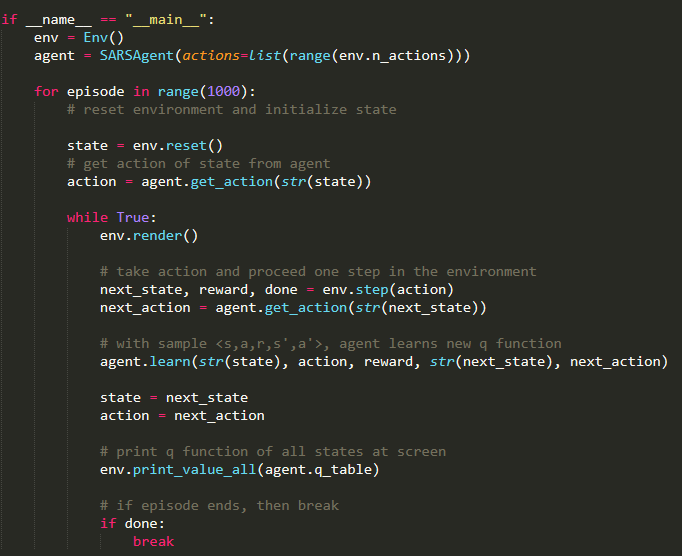


When I add the third triangle to the game, the rectangle becomes stupid. The time it spend on finding circle will depend which function the rectangle chooses. If rectangle follow the route (right-right-down) for the first time, and follow the route (down-down-right), then the rectangle will consider the values of (1,2) and (2,1) will be low, and don’t know where to go. The rectangle still can break the difficult, because there was a random.rand() at 0.1, it will give the rectangle other choice regardless of rule. If the rectangle find the circle easily for the first time, the next several times it will be easy to find the circle.

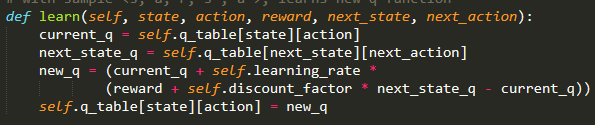
2. Run the SARSA code in the following URL.

https://github.com/rlcode/reinforcement-learning/tree/master/1-grid-world/4-sarsa

Analyze the programs (environment.py and sarsa\_agent.py) and write a report. Add one more triangle with -1 reward at a location you choose. Capture two screenshots of the program execution and include your analysis in the report with the captured images. Place this report in your github.



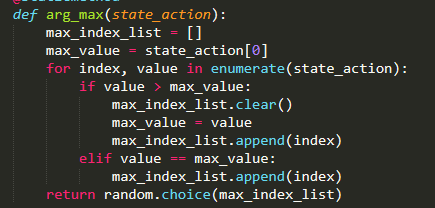
The difference of main function between MC and SARSA is that “action” is needed in SARSA. I will explain it in learning function.



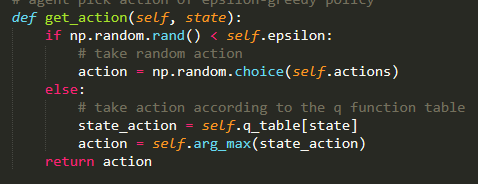
In update function, we will use Q function with action instead of value function.

The new Q function after updating:

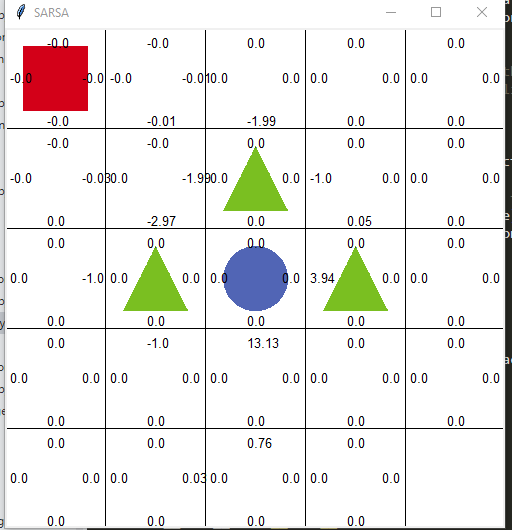
Q\_new(st, at)=Q\_current(st, at)+0.01(R+0.9\*maxQ\_current(st+1, at+1)-Q\_current(st, at))



For the four action, choose one action with bigger value. If more than one action have same bigger value, then choose arbitrary one from them.



From this function, we can see that there are exploration and exploitation in SARSA. “epsilon” is the rate for exploration. “1-epsilon” is the rate , in that the game follow the action according the updating rule.



When I add one more triangle to the game, rectangle will spent much more time to find the circle. The rectangle will choose the route (1,1)-(2,1)-(3,1)-(4,1)-(5,1)-(5,2)-(4,2)-(3,2),rather than (1,1)-(2,1)-(3,1)-(4,1)-(4,2)-(3,2). The reason is that: SARSA will consider the action of current state and next state, when rectangle reach on (4,1), there are there actions (up down and right), but up action and right action might be considered as bad behavior because of έ-greedy policy. In this reason, the rectangle might choose a far path.

3. Run the Q-learning code in the following URL.

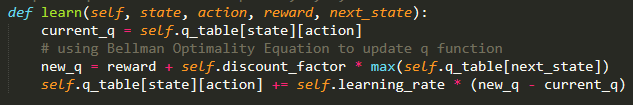
https://github.com/rlcode/reinforcement-learning/tree/master/1-grid-world/5-q-learning

Analyze the programs (environment.py and q\_learning\_agent.py) and write a report. Add one more triangle with -1 reward at a location you choose. Capture two screenshots of the program execution and include your analysis in the report with the captured images. Place the report in your github.

Send me the URL of your github by email [dkkang@gmail.com](mailto:dkkang@gmail.com).

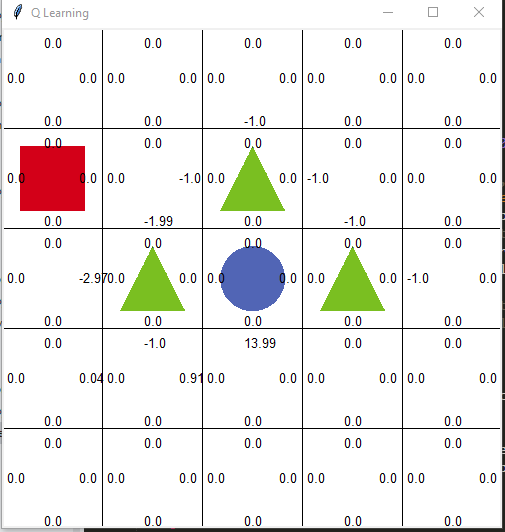
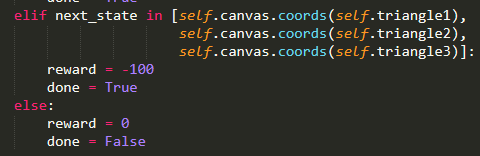
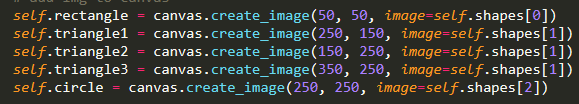


From the main function ,we can see that in Q learning , we don’t need to know the action of next state, only need the know s-a-r-s’. This point is different from SARSA.



In learn() function, we can know how to update Q function:

Q\_new(st, at)=Q\_current(st, at)+0.01(R+0.9\*maxQ\_current(st+1)-Q\_current(st, at))



In Q learing , the red rectangle will spent less time to find the circle, and the rectangle would choose the near route to accomplish the game. The reason is that, SARSA method will not afraid of coming across the triangle, because of considering the action of next state. But for Q learning, rectangle will find circle quickly by using the greedy policy instead of using έ-greedy policy.