Udacity Smartcab project

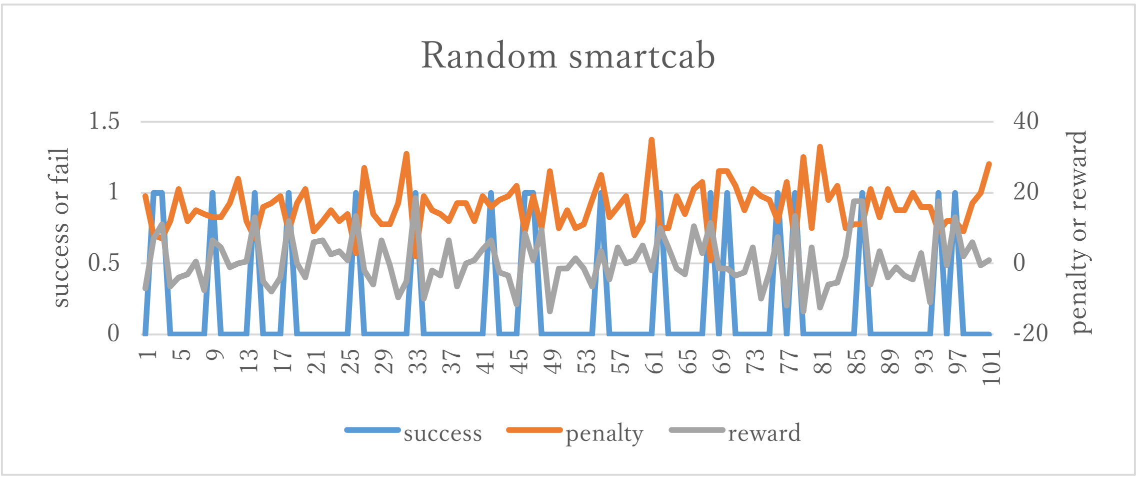
Tomoaki Tsuzuki

Implement a Basic Driving Agent

QUESTION: *Observe what you see with the agent's behavior as it takes random actions. Does the* ***smartcab*** *eventually make it to the destination? Are there any other interesting observations to note?*

ANSWER: *I have run 100 times with random actions. The agent rarely reached the goal within the time with* ***enforce_deadline*** *to be* ***True****. It is 19/100 times. Average reward for those trials is 0.67. Average penalty is 15.8 times per trial (penalty is counted if reward is less than 0).*

The graph below shows how random smartcab performed.
Blue line shows if the agent reached to destination (0 = do not reached to goal, 1 = reached to goal). Orange line shows number of times penalty is given. Gray line shows net reward of the trial.



Obviously there is no trend for, success or fail, penalty and reward.
Hopefully with Q-learning implemented, it shows more success, less penalty, and more rewards as the agent learns.

Inform the Driving Agent

QUESTION: *What states have you identified that are appropriate for modeling the* ***smartcab*** *and environment? Why do you believe each of these states to be appropriate for this problem?*

Answer: *I have selected below as the state variables.*

- *light {red, green}*
- *oncoming {None, forward, left, right}*
- *left {None, forward, left, right}*
- *right {None, forward, left, right}*
- *next_waypoint {None, forward, legt, right)*

The state is combination of those state variables. So it is total of 512 states.
Since possible actions to take is {None, forward, left, right}, my Q-matrix would be the size of 512 x 4.

There is one more state variables that can be used, which is dead line.
However, it greatly increases the size of Q-matrix because deadline variable can take many states (distance between start and destination times 5).

For the agent to learn within the reasonable amount of time, I did not include deadline as state variables.

Implement a Q-Learning Driving Agent

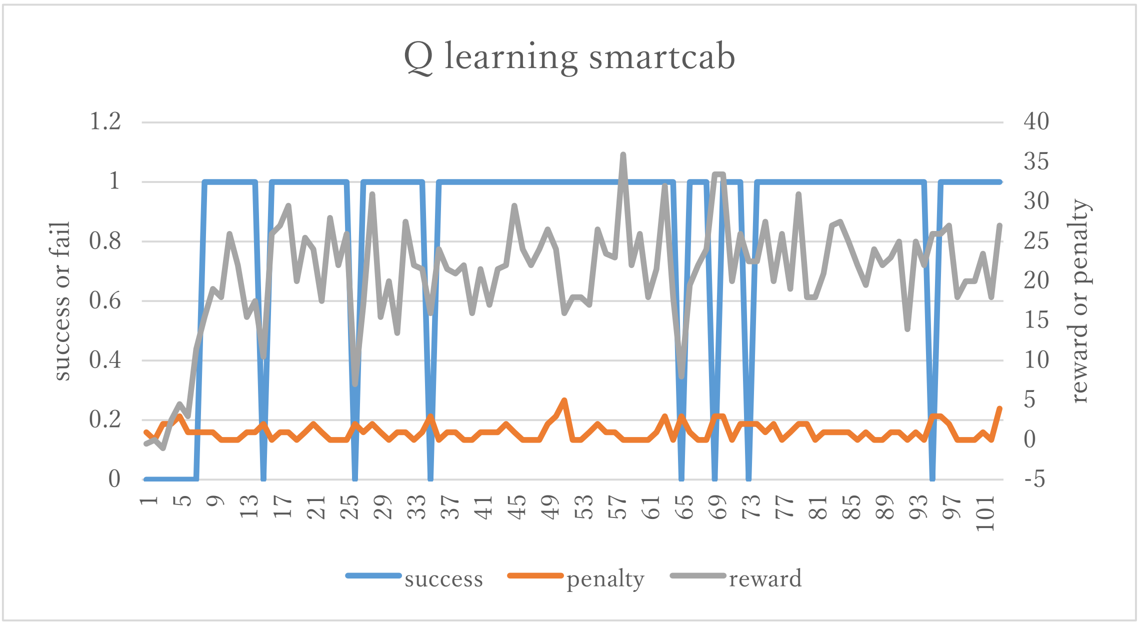
QUESTION: *What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?*

Answer: *With the Q learning implemented, the agent starts to reach goal after few times of trial. This is expected as the agent has learnt how to take action for the first few trials and then trained well enough to get to the goal within the time limit.*

The selected parameter for this trial is

- *Learning rate = 0.6*
- *Discount factor = 0.3*
- *Exploration rate = 0.1*

The graph below shows how Q learning smartcab performed. Again, Blue line shows if the agent reached to destination, orange line shows number of penalties, and gray line shows net reward of trial.



There are great improvements seen by using Q learning. The agent gets to the goal

89/100 times and average reward is about 20.8. Average penalty per trial is about 1. Compare to random smartcab, the result improves dramatically. However, not only the agent does not get to the goal from time to time, but also there are some penalty even around 100th trial.

I think this behavior might be due to the fact that exploration rate is static. I will implement E-greedy exploration and see how it looks like.

Improve the Q-Learning Driving Agent

QUESTION: *Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?*

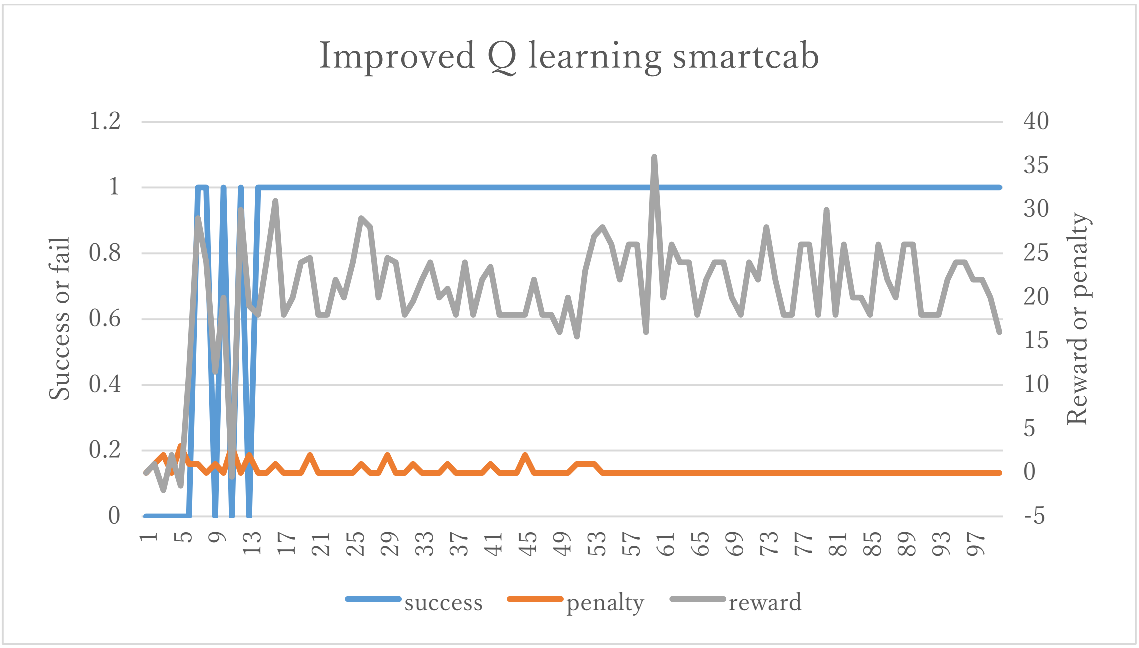
Answer: *To improve performance of the agent, I have modified below things.*

- *Implement greedy exploration*
- *Set Learning rate to be 0.8 from 0.6*

In the non-improved Q learning, exploration rate is statically set to be 0.1. This means the agent randomly choose action once in a 10 times. With greedy exploration, the chances for the agent takes random action reduced by time (in my implementation, by number of learning steps). This will allow the agent to behave very safe after a while as they will only take the action which does not break traffic rules.

Other things I have done is to reduce state space. As input/right state is not relevant because of US traffic rule, I have removed this state variable. This will reduce my states to 128, which was originally 512.

Below is the improved Q learning smartcab performance. All lines represent same as in previous section.

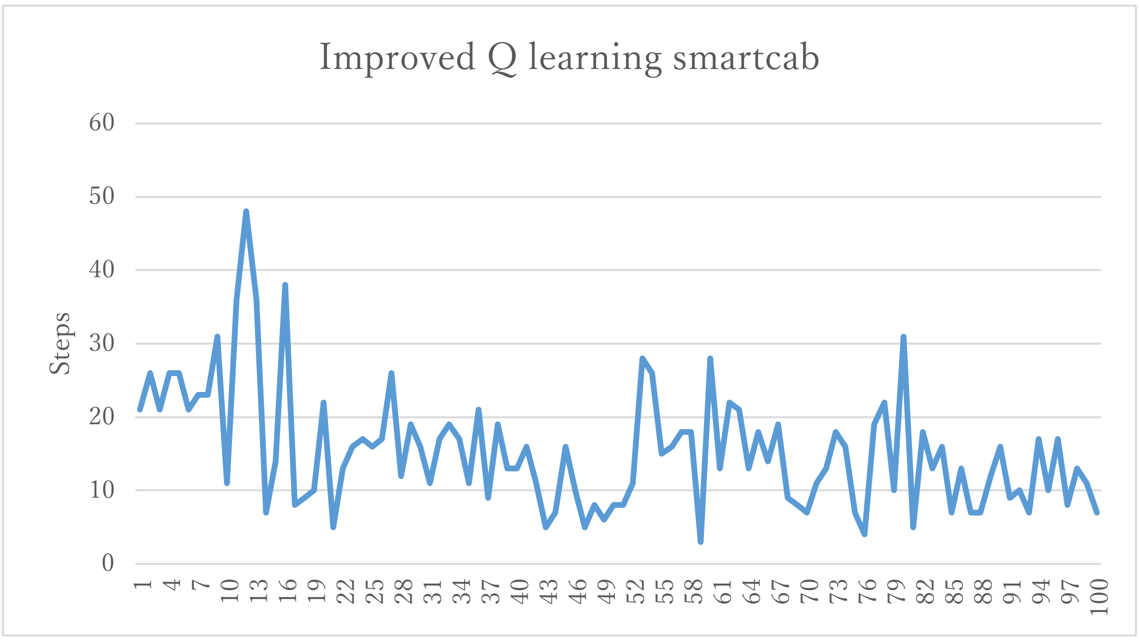


The agent gets to the goal 91/100 times and average reward is about 20. Average penalty per trial is about 0.28.

QUESTION: *Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?*

Notable point here is that after about 50th trial, there is no penalty. This is great improvement because being safe should be the most important feature of smartcab. The agent has learned enough to always get to the goal after about 13th trial.

Below are the steps for the agent taking to get to the goal



Within 100 trials, the agent moves at 1545 times. The action which leads to penalty is 28 times. This means 98% of the time, the agent did not violate traffic rule.
It can also be seen that there would be a trend that the agent need less and less steps to get to the goal.

Optimal policy of this problem would be

- *No penalty (do not break traffic rule)*
- *High success rate (always get to the goal)*
- *Take as small action as possible when getting to the destination*

With success rate, penalty rate, and steps required to get to the goal of my agent, I think it is safe to say that the agent has converged to the optimal policy of this problem.