Train a Smartcab to Drive

Friday, July 01, 2016 10:53 PM

1. 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?

The cab drives aimlessly (without determined driving direction and ignoring all the traffic laws) and only in a few times it will reach the destination by coincidence. Also, because it's random driving manner, it always disobey traffic laws like stop at the red light.

2. 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?

Traffic light, oncoming, left, next_waypoint.

The grid world is composed of 6 rows and 8 columns meaning there are 48 different locations in this grid world. However, in the real world there are numerous locations which makes it impossible to use location as a state.

In addition, deadline also shouldn't be included in the state because it has so many different numbers which is hard to fully train the data in 100 trials.

In fact, no matter where the cab is, it will face limited traffic conditions including traffic light and other lanes' traffic condition. According to the real world's traffic laws, "On a green light, a left turn is permitted if there is no oncoming traffic making a right turn or coming straight through the intersection; On a red light, a right turn is permitted if no oncoming traffic is approaching from your left through the intersection. To understand how to correctly yield to oncoming traffic when turning left." which means the cab should be careful to turn right on a green light because it might interrupt oncoming car's going straight or turning right actions. On a red light, the cab should be careful when it is turning right since there might be cars driving from left going straight. However, the cab shouldn't be worried of the cars from the right lane under normal condition.

As a result, I choose the traffic light, oncoming and left to be components of the state. Also, a cab should provide hints for the cab to drive to the right direction of the destination instead of just driving safely without knowing where to go so I add next_waypoint. In addition, the cab might take different actions when facing these states.

OPTIONAL: How many states in total exist for the smartcab in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?

Traffic light: red, green
Oncoming: None, Forward, Left, Right
Left: None, Forward, Left, Right
Next_waypoint: Forward, Left, Right

So there are 2 * 4 * 4 * 3 = 96 states existing and for each state, the cab can take 4 different actions. 96 states multiply 4 actions equal to 384 Q-values where the 100 trials provide more than 3000 updates for these 96 Q-values so each state should be sufficiently informed.

3. 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?

At the first few initial trials, both the random agent and Q-Learning agent behave in the same manner - moving randomly. However, after a few trials, the Q-Learning cab gradually has the sense to go the direction of destination. At the last few trials, the cab has high success rate (almost 100%) to reach the destination on time.

The accumulated q value and reward give the cab hints to go to the direction of destination faster and the penalty (negative reward) will make cab drive safer by giving negative credits for invalid moves.

4. 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?

In order to get more accurate result, I set the trails to 1000 and use 100 as an interval.

Alpha = 0.6; gamma = 0.5
Success Trials per 100:
[34, 42, 71, 82, 83, 95, 94, 96, 95, 95]

Number of Invalid Move per 100 Trials:
[666, 463, 304, 204, 150, 111, 78, 64, 62, 51]

Average Time Usage per 100 Trials:
[1.499381, 1.399729, 0.983359, 0.786376, 0.739905, 0.546380, 0.507797, 0.508761, 0.513697, 0.485262]

Alpha = 0.8; gamma = 0.5
Success Trials per 100:
[29, 52, 66, 66, 80, 85, 90, 92, 94, 98]

Number of Invalid Move per 100 Trials:
[628, 436, 297, 253, 203, 145, 86, 79, 53, 45]

Average Time Usage per 100 Trials:
[1.590487, 1.250403, 1.041635, 1.003517, 0.810340, 0.711136, 0.615363, 0.605920, 0.517956, 0.516350]

Alpha = 0.4; gamma = 0.5
Success Trials per 100:
[32, 53, 59, 73, 89, 90, 90, 95, 92, 97]

Number of Invalid Move per 100 Trials:
[617, 383, 298, 216, 108, 108, 84, 62, 59, 48]

Average Time Usage per 100 Trials:
[1.554174, 1.212364, 1.139493, 0.919689, 0.698329, 0.644856, 0.617790, 0.574399, 0.596553, 0.514878]

Alpha = 0.6; gamma = 0.2
Success Trials per 100:
[21, 50, 79, 80, 93, 99, 98, 98, 97, 99]

Number of Invalid Move per 100 Trials:
[636, 447, 277, 212, 122, 90, 89, 59, 60, 35]

Average Time Usage per 100 Trials:
[1.684033, 1.268496, 0.830752, 0.814963, 0.613656, 0.484067, 0.482990, 0.457665, 0.492662, 0.414971]

Alpha = 0.6; gamma = 0.8
Success Trials per 100:
[31, 54, 62, 69, 78, 78, 89, 88, 96, 92]

Number of Invalid Move per 100 Trials:
[624, 462, 337, 266, 175, 143, 121, 136, 68, 50]

Average Time Usage per 100 Trials:
[1.53421, 1.249454, 1.092473, 0.982574, 0.847400, 0.885221, 0.669927, 0.697964, 0.578363, 0.608837]

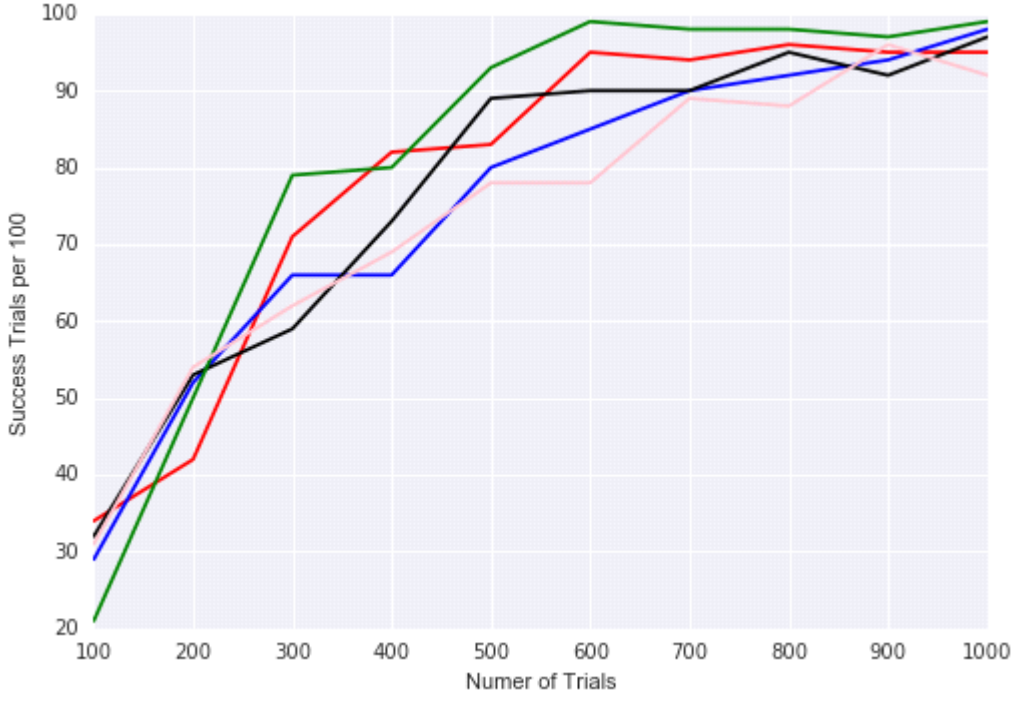
Plotting:

Color assigned for each pair of parameters:

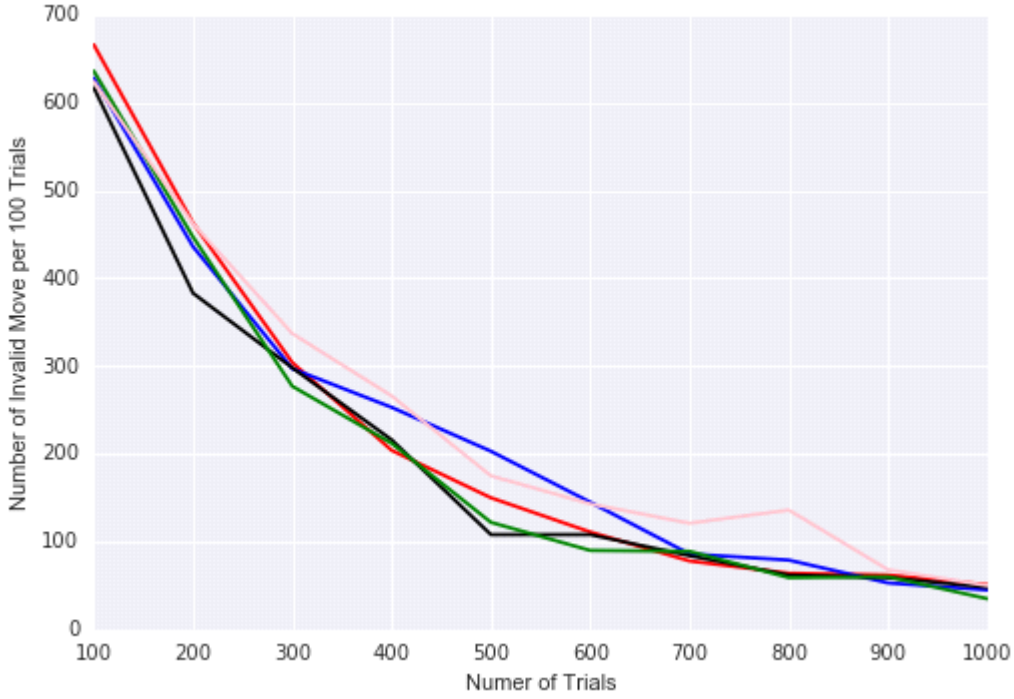
Alpha = 0.6; gamma = 0.5 red
Alpha = 0.8; gamma = 0.5 blue
Alpha = 0.4; gamma = 0.5 black
Alpha = 0.6; gamma = 0.2 green
Alpha = 0.6; gamma = 0.8 pink

Plotting Results:

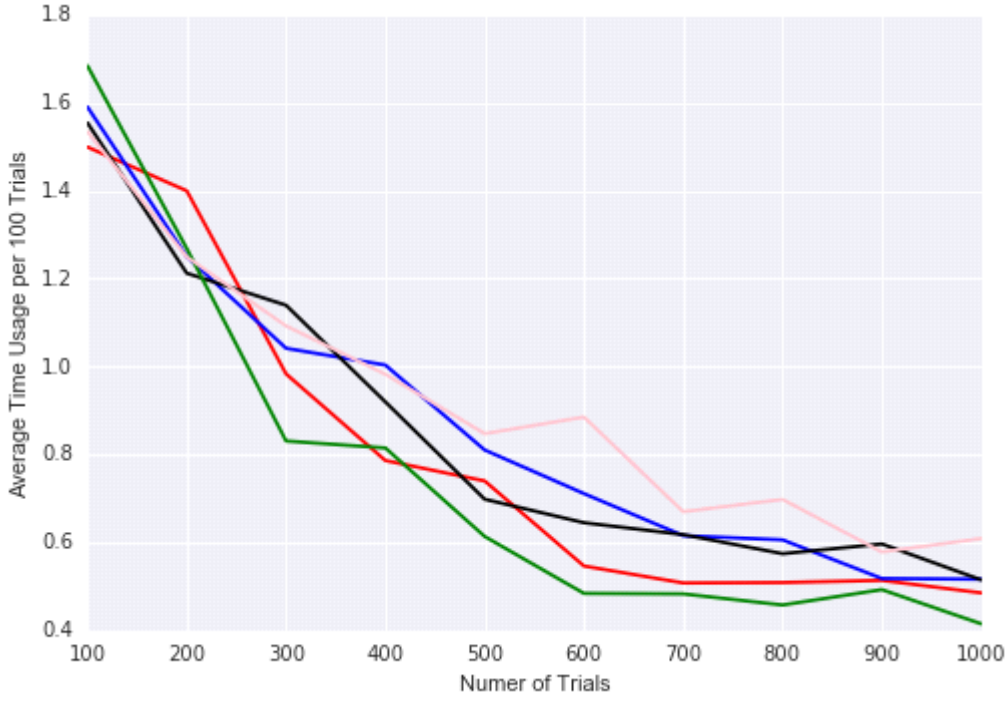
Success Trials per 100:



Number of Invalid Move per 100 Trials:



Average Time Usage per 100 Trials:



Conclusion:
For all the 5 pairs of alpha and gamma, it all use the same epsilon decay rate as epsilon = epsilon * 0.97 for each trial in the 100 trials (for each 10 trials in 1000 trials).

From the above diagrams, alpha = 0.6; gamma = 0.2 , the green line is the best. Since alpha is learning rate and gamma is discount rate, this pair of parameters mean the agent is paying more attention to its recent/current reward instead of old accumulated values.

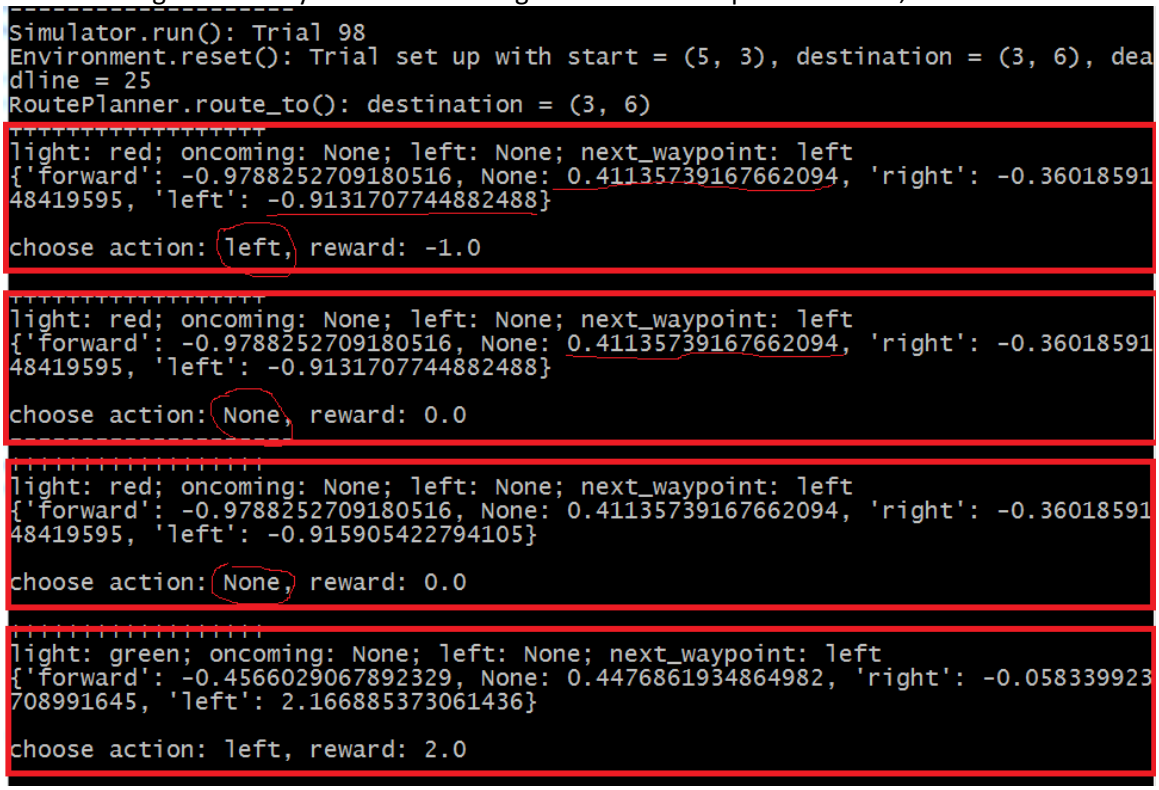
The pair of parameters alpha = 0.6; gamma = 0.2 can arrive destination on time around 99% of all the trials using around 45% of given deadline time.

5. 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?

The agent has a very high success rate (99%) and only use 45% of given deadline time. However, it is easy to find in the diagram of Number of Invalid Move per 100 Trials that there are still around total 50 invalid moves in 100 trials (one trial can have many invalid moves) which means by average there will be one invalid move every two trials.

The optimal policy for a driving agent should be first safe, then successfully arrive at destination and the last is fast. As to this agent, it has a very high rate arriving at destination as desired and also very fast. However, as mentioned above, there are invalid moves which is unsafe. Considering the serious consequences of car accidents in the real world, the optimal policy's number of invalid moves should close to 0.

In order to figure out why sometimes the agent choose sub optimal actions, below is the screen shot.



As is shown above, in the first action, the agent choose left which is really hard to understand since None actually has the maximum value. In the second action, facing the same condition, the agent makes a right choice this time and so as the following actions.

Some other invalid moves are due to insufficiently exploration. For example, with a q table {forward: 0, None: 0, right: 0, left: -0.97343}. Because the max q value here is 0 and there are three 0, so the program randomly choose one action and later it violates traffic laws.