report

October 9, 2016

1 SmartCab Solution Report

1.1 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?

Smartcab eventually reaches its destination but takes a longer time. Also notice that it does not stop on red lights sometimes. Thirdly, as expected, it is not learning from past mistakes. With DeadLine NOT enforced, it misses trial and does not reach destination in 100 attempts or less.

1.2 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?

State I chose is a combination of Light, Oncoming and Next WayPoint. The selection of state seem reasonable as smarcab reaches destination with given constraints. **Light** This is important as a basic fundamental rule of driving is to follow light signal. In this case there are two choices, True and False meaning if it is Green or Red. **OnComing** This state tells us whether there is oncoming traffic or not. It is also important as cab can turn right on a red light but needs to watch oncoming traffic. Other way it is important is Oncoming traffic gets a priority when light is green and cab needs to turn left. **NextWayPoint** where to go from here? This state sums the last two states and defines the next action. Once the action is acted upon, a reward is assessed. Without this state, cab will only learn half the conditions and may start circling around the block.

- Left Out States: * There are three states left out from the choice of states. 1) **Left**: This state tells the direction left traffic is turning toward. There is a situation when rewards are affected from the left traffic. If left traffic is coming forward, and agent decides to turn right on a red light, there is a chance of collision. The agent needs to remember to yield to oncoming traffic from left. If not yielded, agent should be penalized with negative rewards whereas turning right when left is not coming forward should be rewarded. However, leaving this state as part of Q matrix is feasible. Adding this state would make the number of states agent has to scan and learn will bloat making the learning process longer and making it more likely to miss deadline. learning from not yielding to traffic coming from left is simple and cheap. Hence, leaving it from set of states in Q matrix is justified.

2) **Right**: Right traffic is controlled by lights and is not important to learn.
3) **Deadline**: Influenced by deadline, an agent would be rushed into reaching destination not minding penalties. Deadline is subjective and can be realistic or non realistic. Learning an optimum policy should be exclusive of deadline.

1.3 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?

Total States: Total Locations * Total Light States * Total Oncoming * Total Actions * Left * Right * Next Waypoint Total Locations: 48 (86 *grid) Total Light: 2 (True and False) Total Oncoming: 4 (None, Forward, Left, Right) Total Actions: 4 (None, Forward, left, Right) Total Left: 4 (None, Forward, Left, Right) Total Right: 4 (None, Forward, Left, Right) Total States = 48* 2 * 4 * 4 * 4 * 4 Total States: 24576

We do not need Location on Q Learning states as it will take a very long time learn. Although, getting a complete state picture is ideal but is not possible in a limmited number of trials. Similarly, Left and Right are not needed.

1.4 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?

Noticed three important changes. 1) When picking actions from Q matrix, agent still explores a little bit. Upon exploration if it finds a low reward and when it comes across a similar situation, it picks a different route until it knows the best action to take based on rewards and Q policy. 2) Agent updates Q matrix incrementally untill all actions are explored. This makes every subsequent trial likely to be better than last ones. 3) Once converged, agent picks up the right path quickly and reaches to destination almost every single trial within the deadline.

1.5 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?

Best Combination Agent Performed 80+ successful trials for its best trials with a minimum penalty of 26. On an average, across all combinations, agent was successfull 70+ trials with penalties ranging from 26 - 200. * gamma = 1.0 * alpha = 1.0 * epsilon = 0.0 ***Other Combinations*** * gamma = 0.9, alpha = 0.6, epsilon = 0.0 * gamma = 0.8, alpha = 0.7, epsilon = 0.2 * gamma = 0.8, alpha = 0.6, epsilon = 0.2 * gamma = 0.6, alpha = 0.3, epsilon = 0.4 * gamma = 0.4, alpha = 0.3, epsilon = 0.3 * gamma = 0.7, alpha = 0.6, epsilon = 0.3 * gamma = 0.7, alpha = 0.6, epsilon = 0.2 * gamma = 0.9, alpha = 0.7, epsilon = 0.2

1.6 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?

No agent does NOT find an optimum policy. Best the agent can do is 80+ successfull trials with minimum penalty (26). On an average with all combinations of parameters agent made 70+ successfull trials with penalty ranging from 20 - 200. Max successful trials was 100 but with penalties reaching 92. Agent is, however, making progress and is converging to a stable Q matrix. ('red', None, 'left') : [0.0, -1.0, -1.0, -0.5], ('red', None, 'right') : [0 , 0 , 0 , 2.0], ('red', 'left', 'left') : [0.0, -1.0, -1.0, 0], ('green', None, 'left') : [0.0, 0 , 2.0 , 0], ('red', 'left', 'right') : [0.0, -1.0, -1.0, 0], ('green', None, None) : [0 , 0 , 0 , 0], ('red', None, None) : [0 , 0 , 0 , 0], ('green', 'left', 'forward') : [0 , 4.0 , 0 , 0], ('green', 'left', 'right') : [0.0, 0 , 0 , 12.0], ('red', None, 'forward') : [0.0, -1.0,-1.0 , -0.5], ('green', 'right', 'forward') : [0 , 0, 0 , -0.5], ('red', 'left', 'forward') : [0.0, -1.0,-1.0 , 1.5], ('red', 'forward',

‘forward’) : [0.0, 0, 0 , 0], (‘red’, ‘forward’, ‘right’) : [0.0, 0, 0 , 14.0], (‘red’, ‘right’, ‘forward’) : [0 , 0 ,-1.0 , 13.5], (‘green’, ‘forward’, ‘forward’): [0.0, 0 , 0 , 0], (‘green’, None, ‘right’) : [0.0, 0 , 0 , 14.0], (‘green’, None, ‘forward’) : [0 , 12.0, 0 , -0.5], (‘red’, ‘forward’, None) : [0 , 0 , 0 , 0]}, We can see that Q matrix is converging although not fully converged yet. That is evidence that policy is gearing towards optimum state. Although we are making progress, we see wrong actions, hence negative rewards. For e.g. **current_negative_reward= [‘Wrong Action: Light: green, Oncoming: None, NextWaypoint: forward, action: right, Reward: -0.5’]** In this case, There is no oncoming and light is green but instead of moving to the next waypoint (forward), it moves in some other direction (right). To that affect, it enjoys the negative reward (-0.5). **current_negative_reward= [‘Wrong Action: Light: red, Oncoming: None, NextWaypoint: left, action: right, Reward: -0.5’]** In this case, agent notices a red light so it does not take a left, but it takes a right instead. It probably should have done nothing. So we can see how the agent is learning and makes to the destination but with some penalties. In this way, learning through penalties and positive rewards. The policy is gearing towards optimality.