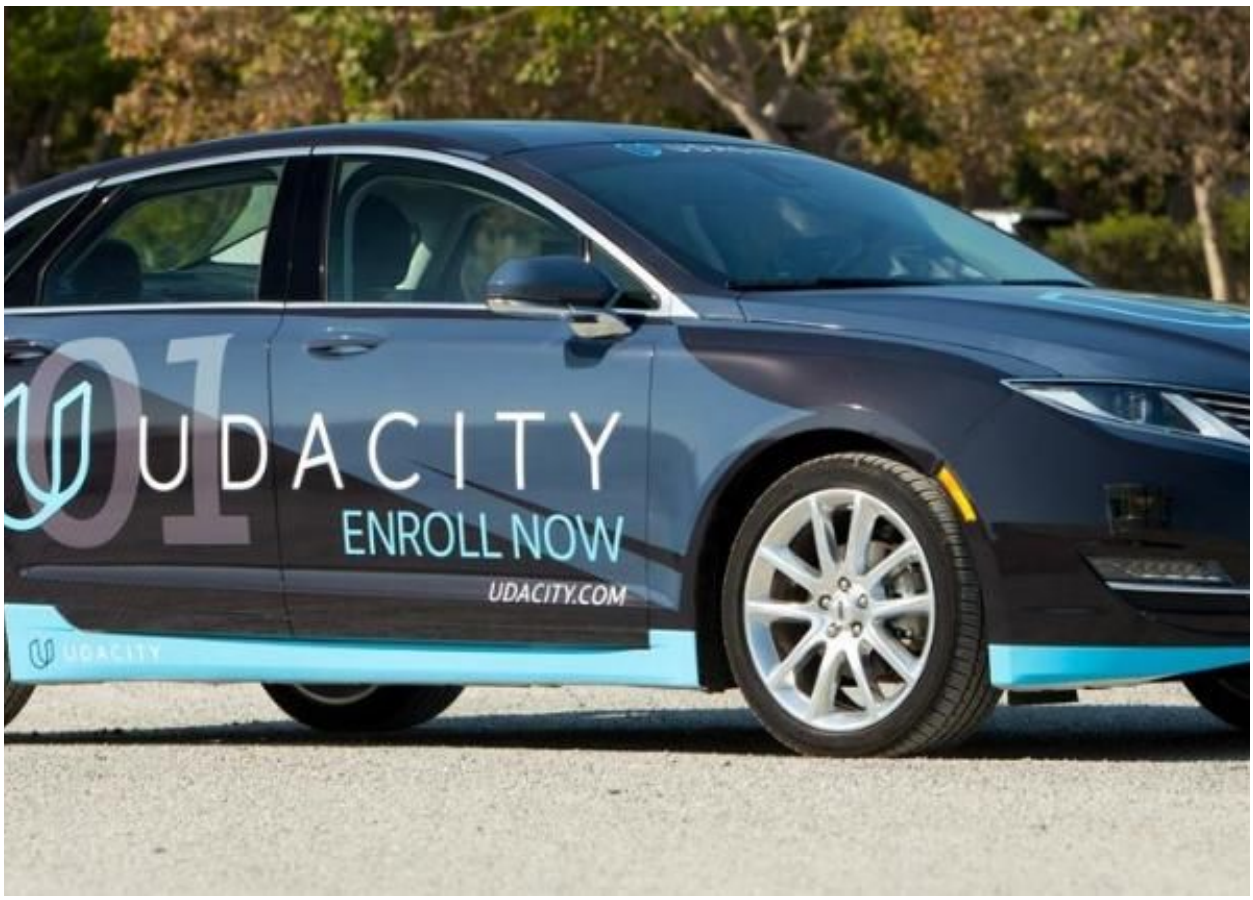
Udacity Machine Learning Engineer Nanodegree

How to train a smart cab

Report by Dominic Wong



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

The agent's behavior is, as expected, random. Sometimes the agent makes a correct move and sometimes it doesn't. Each move has no correlation with the previous/next move whatsoever. After many many rounds of moving towards all possible directions, I luckily saw the smart cab to arrive at its destination. It is interesting to see the smart cab making a streak of good moves and getting really close to its destination, then suddenly it would take a sharp turn and go astray. However, it is an expected behavior because I have not yet implemented a policy that determines the best action to take by looking at the direction of the next waypoint, traffic on the road and the time remaining.

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

I have chosen 'light', 'oncoming', 'right', 'left', and 'next_waypoint' as the appropriate states for modeling the smart cab and environment. I believe these states are appropriate for this problem because

I. Traffic lights almost always matter. They tell you whether you can hit the gas pedal or not. However, a red light doesn't always mean you can't move and a green light doesn't always mean you can move. One can turn right in a red light (unless told not to) when there is no forward traffic from the left or left traffic from oncoming direction. One cannot turn left in a green light when there is forward traffic from oncoming direction. While traffic light itself is an important state, the policy cannot solely rely on it to make a decision. It also needs the traffic information at the intersection to determine whether a move is valid or not.

Ii. Traffic conditions from various directions matter not only because the reason aforementioned, but also the fact that collisions are fatal (well, this is why we have the turn/no-turn traffic rules in red/green lights in the first place). Here is another scenario. Let's say the cab abides by the rule and is about to move forward after seeing the green light. However, there is a law-violating vehicle that ignores the red light and keeps moving forward from the left. It is then a better decision for the cab to not move because avoid a collision should receive more reward than following the traffic rules.

Iii. Finally, knowing where the next waypoint is important because a cab can be making all the valid moves and will still end up not making it to the destination if it does not know where it should go. As I saw by experimenting with making random actions, the cab can make some good moves with positive rewards, and then it suddenly goes elsewhere. The cab should move towards the next waypoint whenever possible.

Deadline is not included in the states because it has too many variations. From what I observed, it can have between 20 to 50 steps. In the best scenario, it will still increase the number of states 20 times, which makes a successful learning much harder to achieve.

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?

A traffic light has 2 states, red and green.

Oncoming, left and right traffic each has 4 states, forward, left, right and none.

Next waypoint has 3 states, left, right and forward.

The total number of states is 2 * 4^3 * 3 = 384.

It seems like a reasonable number. The hard_time_limit is set to 100 in the class Environment. While it is unlikely to learn all 384 states in the 100 trials, I believe the car will end up not running into most of the states, especially there are only 3 other cars on the road in a 6x8 grid. The chance of getting something other than ‘none’ for ‘oncoming’, ‘left’ and ‘right’ traffic is quite low. Q-learning is most likely able to learn and make informed decisions about each state.

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?

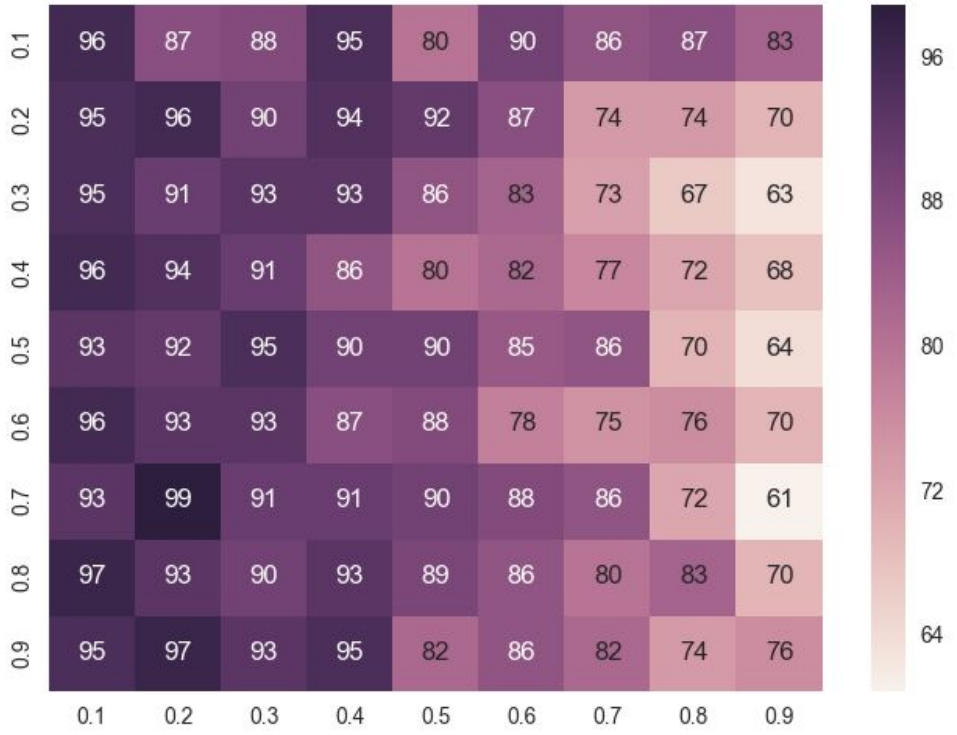
When random actions were always taken, it was almost impossible to reach the destination. After implementing the Q-learning algorithm, the driving agent has a very high chance (99% was the best result) of reaching the destination before the deadline.

After a few trips, I noticed the driving agent started learning to do a few things

1. Stop in front of a red traffic light when it is needed
2. Actually move towards the next waypoint
3. Avoid colliding with cars

The driving agent's behavior has changed because it now attempts to maximize its reward. Every time the agent makes a move, it first checks the available actions for its current state and chooses the action with the highest Q-value. Making the 'optimal' move does not guarantee a positive reward because there might be situations the driving agent has not encountered before. Whether positive/negative, the driving agent always adds the reward/punishment to the existing Q-value so that next time when it is in the same state, it learns better. Over time, the driving agent should have enough experience with a few of the common situations and is more likely to make a move that results in positive rewards.

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?



The best combination was
Alpha: 0.7
Gamma: 0.2

It has 99 successful trips with a net reward of 2224.5. This is a very impressive result as it means the agent only took 1 trip to learn what a good move is.

The best result occurs with a relatively high alpha value and a relatively low gamma value. The long-term reward is taken into account less importantly than the current reward.

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?

My agent ended up finding an optimal policy. The success rate is very high (99%) but it will not be 100% perfect because there are some states that rarely occur and it will be hard to get meaningful Q-values from these states. These rarely occurring states are the ones with other cars' presences. There are simply too few cars in the grid for these states to occur frequently. An example state would be (light: 'green', left: 'forward', right: 'forward', oncoming: 'forward', next_waypoint='forward')

An optimal policy would know how to do a few things,

- Avoid colliding with another car
- Follow the traffic rules
- Move towards the correct next waypoint instead of a random direction