Smartcab
Udacity Machine Learning Nanodegree

Chiel Peters

October 24, 2016

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?

When the enforce deadline parameter is set to false the smartcab does eventually reach its destination. However it does so very slowly without any learning. Other interesting observations I made: the next waypoint is not always the most optimal route to take as it does not take into account the current state of the stop light/traffic however it is a good indicator for direction, the grid does not have boundaries that is falling off on the left returns the car on the right. There is no way to ignore a red if the action forward is chosen and the stoplight is red then the car will just stay in place.

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?

In order to model the smartcab and the environment three pieces of information are needed: stoplight status, traffic situation and route to destination. The first two can be found in the inputs variable. However to simplify and reduce the state space I inserted some domain logic. That is states that are equivalent in allowed actions are joined. This is done by transforming the oncoming variable which has four possibilities (None,left,right,forward) into a boolean variable. This boolean variable signifies if the smartcab is allowed to turn left which is equivalent to if there is incoming traffic going forward or to the right. The same is done to the traffic on the left a new variable is created which signifies if the smartcab can turn right according to the incoming traffic. The variable is

made by the condition not(left = 'forward') that is 'Is there no incoming traffic from the left going forward?'. This rules simplify the state space significantly as they reduce the traffic situation from 4 x 4 = 16 possibilities to 2 x 2 = 4 possibilities without a loss of information. For the route to destination I used the next waypoint variable. This gives a good indication of direction for the smartcab.

To summarize the state space consists of four variables [(State of stoplight),(Can I turn Left according to the traffic), (Can I turn right according to the traffic), (next waypoint)]. I believe these states sufficiently describe the problem as they convey information on all aspects of the smartcab and the environment. It should reach the destination by means of the next waypoint and learn the rules of the road by the stoplight and traffic variables.

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?

In total there are 2 (green,red) x 2 (True,False) x 2 (True,False) x 4 (None,left, right,forward) = 32 states. The None state is actually never reached for the current implementation of the next waypoint so in reality only 2 x 2 x 2 x 3 = 24 states are present, however for completeness and generality the other states are also included. It seems reasonable to assume that all 24 states are reached a number of times when the number of trials is high. Although the number of states is small the probability of each state is not uniformly distributed. With the current setup with two other cars on the grid there is only a very small probability of having both incoming traffic from the front and left at the same time. Therefore the Q-learning will have difficulties learning in these states as they do not occur very often however I believe that with a high number of trials the algorithm can also learn in these states.

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?

In the beginning smartcab is still acting random. This is because the Q-learning algorithm has not yet figured out the optimal move in every state. After a few trials however the agents behavior starts to resemble that of a optimized driver. It stops when the stoplight is green and the correct direction is not right and

it follows the directions given by the next waypoint. This behavior is occuring because of the Q-matrix. In the beginning for every action in every state the Q-matrix will be 0. Therefore for a state the smartcab still acts random as there is no optimal action with a high utility. After a while it has explored a number actions for a number of states and the algorithm starts learning from the utilities, that are formed by the (future) rewards, which actions are optimal in which states. For a state the smartcab observes the past utilities of all actions in that state and is able to pick the action with the highest utility. To avoid local optima and to allow exploration the algorithm does sometimes still act random, this occurs epsilon percent of the time. After a large number of trials every state and action have been explored and the Q-matrix has learned the optimal action, that is the one with the highest utility, for every state.

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 succes rate is defined as the number of times the smartcab reaches its destination within the deadline. Although we could deploy 'smarter' metrics such as reward-per-action or actions over distance between start and end point in this case we prefer simplicity. There are three parameters which can be tuned: gamma (discount factor for future rewards), epsilon (exploration) and alpha (learning rate). The algorithm is run for 3 different values for each parameter (27 times in total). The values are based upon the values found in the lectures and online resources. The succes rate for each run is stated in the tables below. The optimal configuration is $\gamma = .95$, $\alpha = .03$ and $\epsilon = .01$ with a succes rate of 86.6 %.

ϵ/α	.01	.03	.1
.01	94.8 %	93.8 %	95.8 %
.03	95.8 %	93.8 %	94.4 %
.05	94.2 %	93.2 %	93.8 %

Table 1: Success rates for gamma (γ) equals .85

ϵ/α	.01	.03	.1
.01	96.2 %	94.2 %	94 %
.03	96.6 %	92.8 %	93.8 %
.1	96 %	93.6 %	93.6 %

Table 2: Success rates for gamma (γ) equals .9

ϵ/α	.01	.03	.1
.01	97 %	98.6 %	74 %
.03	94.6 %	92.6 %	61.2 %
.1	94.4 %	91.6 %	93 %

Table 3: Success rates for gamma (γ) equals .95

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 will not find the optimal policy 'reach the destination in the minimum possible time and not incur any penalties'. One reason for this is that the smartcab follows the waypoints untill it reaches the destination. However the waypoints do not account for the current traffic and stoplight status. It therefore sometimes misses that it could take a right turn on a red signal to get closer to the destination and advises the smartcab to go forward which it cannot do because of the red light.

However the smartcab does find another optimal policy, because of the way the state space and learning are defined. Note that this only holds for the proportion of time when the smartcab performs exploitation on what is has learning instead of performing a random action (exploration). In the optimal policy it will follow the waypoints towards the destination while not violating any traffic lights or collisions.

However it takes a long time to learn this. The Q-matrix is printed below for 500 trials. All the rows where there is traffic from the front (Oncoming equals false) and traffic from the left (Left is false) are 0 since the smartcab has not observed traffic from both sides at the same time. Also as mentioned earlier the waypoint case of None is never observed. For the other states it can be observed that in case of no traffic (Oncoming and Left are true) the smartcab has learned to follow the waypoint when the stoplight is green and it turns right when the stoplight is red. For cases with traffic from one side the smartcab has not learned enough and still does suboptimal actions.

light	Oncoming	Left	Waypoint	None	left	right	forward
green	True	True	None	0.000000	0.000000	0.000000	0.000000
			left	0.099074	6.437750	0.028832	0.000000
			right	0.264628	0.059053	7.465507	-0.002250
			forward	0.158446	0.294394	0.017115	6.851575
		False	None	0.000000	0.000000	0.000000	0.000000
			left	0.000000	0.000000	0.272817	0.000000
			right	0.000000	0.000000	0.823292	0.000000
			forward	0.000000	-0.002528	0.554670	0.000000
	False	True	None	0.000000	0.000000	0.000000	0.000000
			left	0.000000	0.000000	0.057956	0.000000
			right	0.000000	0.000000	0.000000	0.377701
			forward	0.000000	0.000000	0.000000	0.310074
		False	None	0.000000	0.000000	0.000000	0.000000
			left	0.000000	0.000000	0.000000	0.000000
			right	0.000000	0.000000	0.000000	0.045516
			forward	0.000000	0.000000	0.000000	0.000000
red	True	True	None	0.000000	0.000000	0.000000	0.000000
			left	0.042563	-0.001768	5.075248	0.013391
			right	0.297940	0.016283	7.508185	0.184644
			forward	0.070744	0.011598	4.585155	0.013394
		False	None	0.000000	0.000000	0.000000	0.000000
			left	0.000000	0.000000	-0.010000	-0.010000
			right	0.000000	0.000000	0.000000	-0.010000
			forward	0.000000	-0.010000	-0.010000	-0.010000
	False	True	None	0.000000	0.000000	0.000000	0.000000
			left	0.000000	-0.010000	-0.002694	-0.010000
			right	0.000000	-0.010000	0.650707	0.000000
			forward	0.000000	-0.010000	-0.003994	-0.010000
		False	None	0.000000	0.000000	0.000000	0.000000
			left	0.000000	0.000000	0.000000	0.000000
			right	0.000000	0.000000	0.000000	0.000000
			forward	0.000000	0.000000	0.000000	0.000000

Table 4: Q-matrix found for optimal parameter set configuration