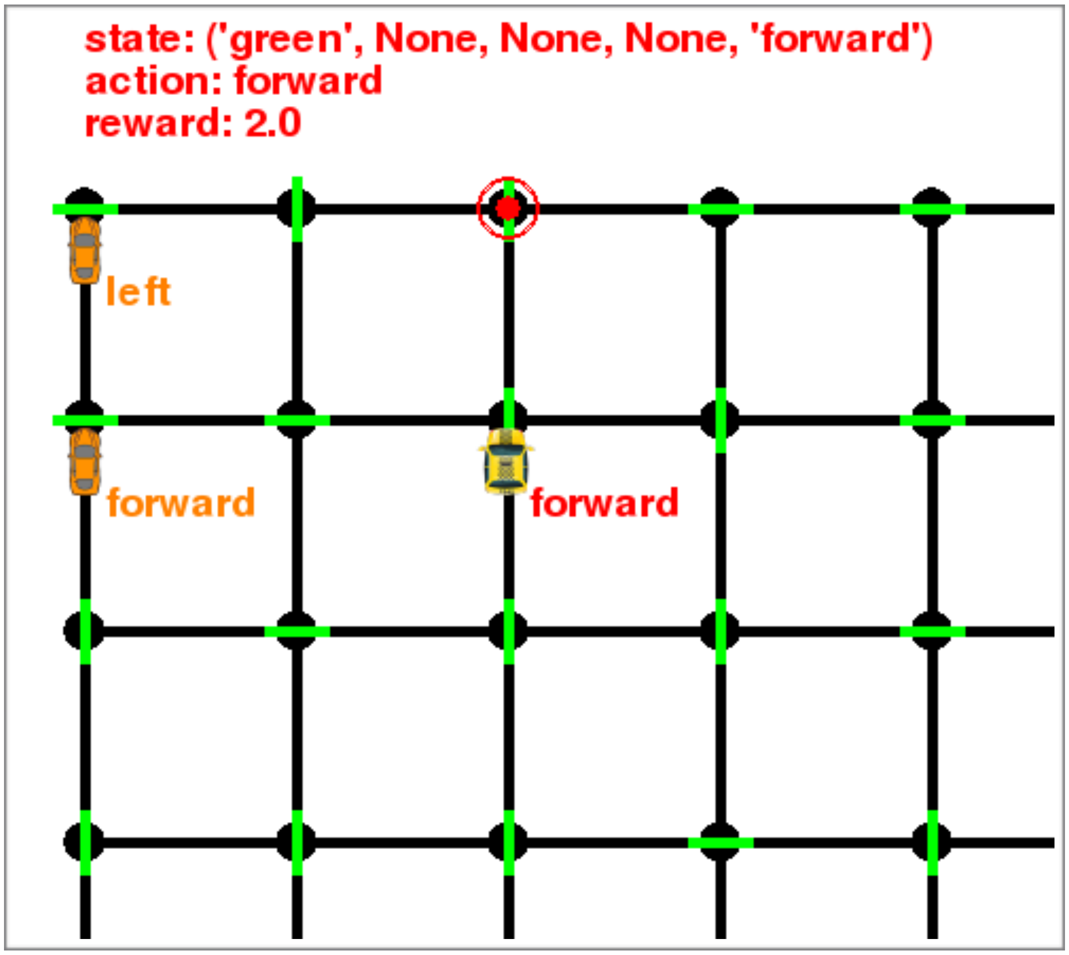
Project 4 Report

Train a Smartcab to Drive


state: ('green', None, None, None, 'forward')
action: forward
reward: 2.0

Salvatore Mitrano

July 2016

1. *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?*

Taking random actions lead two two possible result. If the smart can is lucky enough, it reaches the destination. However, 80% of the time this is not the case, as a matter of fact, the environment hits the hard time limit of -100 and the game stop to avoid deadlocks.

Furthermore, another interesting observation to note is that if we set the actions to next_waypoint, the smart cab reaches the destination. However, it will disregard all the traffic laws, crashing with vehicles and passing with 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?*

The state I have identified for modeling the smart cab and environment are a combination of input traffic light, oncoming traffic, right traffic, left traffic and planner route. I believe that this combination is the minimum needed to create a model. I was thinking also to add another boolean variable to model when the time was running out. However, the smart cab was performing really well and I did not want to add more complexity to the system, since adding that variable would have doubled the number of states.

The variable for the traffic light is important because it allow to consider red and green possibilities, in order to avoid traffic penalties. The traffic oncoming from forward and left are important because it allows to avoid the crash penalties. The planner next way point is important because it allow to consider the right path to follow, and avoid penalties for not following the right path to destination. Finally, the traffic oncoming from the right variables can be avoided due to the simplification of the problem, which does not allow u turn on the same road, and based on the US traffic law.

3. (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?

The variable I have chosen to model a state for the Q-Learning are the following:
~ Traffic Light : {green, red}
~ Traffic oncoming forward: {none, forward, left, right}
~ Traffic oncoming left: {none, forward, left, right}
~ Planer next way point: {forward, left, right} —> none can be disregarded because final case

Since all possible states is the combination of those variables, the possible states are 2*4*4*3 = 96 possible states.

As I have stated prior, I believe that this combination of variables is the minimum needed for the Q-Learning algorithm to make informed decisions. For example, if you eliminate the traffic variables, the model is to generalized. On the other end, if you add more variables such time, the model is to complex and will need more training than expected.

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?

After starting implementing Q-Learning, the behavior of the agent changes. As a matter of fact, the agent start respecting traffic rules, follows the planer path, and make no accidents. This is due to the reward policy I have created to make the agent learn the basics. Indeed, the states I have chosen represent all possible scenarios the agent will encounter. During learning, the rewards for each possible actions related to that state are updated according to the traffic law imposed by the environment. Finally, after training, the agent uses those rewards to make the right decision, choosing the best action for the specific state it is in.

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?*

Discount Factor — Trial #	Average Net Reward	Average # of Penalties	Average # of moves	Success Rate
γ=0.0 — Trial 1	9.515	11.26	24.66	38.0%
γ=0.0 — Trial 2	9.22	12.39	26.79	35.0%
γ=0.2 — Trial 1	9.75	10.94	24.61	39.0%
γ=0.2 — Trial 2	9.78	11.24	24.82	43.0%
γ=0.4 — Trial 1	10.36	12.18	25.3	41.0%
γ=0.4 — Trial 2	10.635	11.78	24.32	39.0%
γ=0.6 — Trial 1	10.16	12.23	24.89	29.0%
γ=0.6 — Trial 2	8.835	11.93	23.42	31.0%
γ=0.8 — Trial 1	8.925	12.6	24.78	29.0%
γ=0.8 — Trial 2	9.66	13.18	25.93	31.0%
γ=1.0 — Trial 1	5.885	12.82	25.89	25.0%
γ=1.0 — Trial 2	4.415	12.95	27.57	18.0%

The parameter I have tuned for my implementation of Q-Learning are the decaying learning rate alpha, which decays over time. The possible range for the decaying learning rate are between 0.1 to 0.9. Another parameter I have tuned is the discount factor. I found that the value between 0.2 and 0.4 work best for this problem, as you can see from the table above. As a matter of fact, If I set up the parameter at 1, the agent will act weirdly, making loop around the block. Therefore, I decided to go with an average of 0.3. I tested this out and the following table shows the results.

Discount Factor — Trial #	Average Net Reward	Average # of Penalties	Average # of moves	Success Rate
γ=0.3 — Trial 1	9.27	10.49	22.35	40.0%
γ=0.3 — Trial 2	10.42	10.99	23.75	42.0%
γ=0.3 — Trial 3	9.735	11.02	23.37	41.0%

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?*

The optimal policy for this problem would be to reach the destination before the deadline with minor penalties. Indeed, the agent follows this optimal policy. As a matter of fact, the agent follow the planner direction and does incur in minor penalties. Moreover, I have noticed in 10% of the cases that the agent takes a turn in the wrong direction. I believe this is due to the hight number of possible states. Indeed, in my previous submission I have trained the agent with more training cycles and more dummy agents. However, It was rejected because I had to do it with 100 cycles. Indeed, I believe, that if we were able to train more the agent, and expose it to more edge cases, it would perform better overall.