# Train a Smartcab using Reinforcement Learning

**Q1**. *In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?*

**Answer**: Yes, it was able to reach destination ultimately, but after large number of steps. Also the number of steps was unpredictable.

**Q2.** *Justify why you picked these set of states, and how they model the agent and its environment.*

**Answer**: Initially I choose only other vehicles and traffic light to define the state. But after multiple trials, I figured out that this was not enough. There was a temptation of using coordinates as part of state, but than it would have increased dependence of accumulated knowledge on starting point, and would have rendered knowledge useless if this car would go to some other area.

I then decided to choose, deadline and navigation inputs as part of state.
For this I have created a 5 dimensional panel to hold State and Actions:
Resulting State-Action space had dimensions::4 X 4 X 100 X 512 X 2
But this resulted in a very sparse matrix, and given limited (10) trials, there was about 40% success rate

Then I removed deadline from state space, so that Q-Table reduced by a multiplier of 100.
This considerably improved the success rate, to about 95%.

**Q3:** *What changes do you notice in the agent's behavior?*

**Answer**: Following were success rates for different values of Alpha and Gamma:

**Q4:** *Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform?*

**Answer**: Have parameterized exploration / exploitation ratio, and also tried with different values of alpha and gamma.

Observations:
1. In absence of Simulated Annealing:
    Initial Q values should be greater than 1 for optimal performance.
    If Q values were less than 1, Agent was not able to learn anything.
2. Simulated Annealing:
    Even if the initial Q values are less than 1, with annealing enabled, agent starts reaching destination within 7-10 trials.

3. Higher epsilon, higher alpha and lower gamma yield better success rate.
    Log attached for following values:
    *alphas = [0.7, 0.5,0.3]*
    *gammas=[0.005,.05, 0.1]*
    *epsilon_probabilities=[0.5,0.2,0.1]*

*Epsilon was found to most important parameter, as it allowed agent to learn initially.*
*Epsilon in my code reduces with time or trial count, which means agent depends more on knowledge once it has aggregated it.*

*Tuning Epsilon function was difficult, as a rapidly decreasing probability function = 1/ trial_count did not allow agent to learn for more trials.*
*Hence, I have used 10/ trial_count, which gives more trails to agent to explore. And then it stops gradually.*

*I initially tried with higher initial Q value > 1.0, for example 2. And this gave best results even without annealing, since higher Q value encouraged agent to explore more.*

*But challenge was to learn with low or ZERO initial Q value. And this when simulated annealing has helped build knowledge.*

*Have also reduced state space for traffic; better use of dimensions for traffic from 3 directions.*

**Q5:** *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?*

**Answer**: Yes, towards the end, rewards increased and steps required to reach destination decreases.
Have increased minimum reward points to consider success as 10.
And have achieved rewards of up to 95.

Towards the end, I have converted QTable into verbal output.
Analyzing this output, it is clear that there is a certain policy learnt.

For example, For state:
Left: None -+- Right: None -+- Oncoming: None | Light: Red | Action: Left
there is negative Q value.

Output is in annealing_support.txt file same folder.
Reward points, for optimal cases, are mostly in range of 90-95.

Since we have moved out deadline from State space, we loose out on knowledge to change when there is less time left. Would try to do that with more trials, say 2000.
Including deadline could improve Q table entries in case of 'None' traffic situations, which I guess is not optimized yet.