

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

With random action choice and `enforce_deadline=False` agent eventually makes it to target location but due to randomness of his moves completely misses deadline and gets a lot of negative rewards for bumping into other cars and not stopping on traffic lights.

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

State variables:

'light' - you need to know this to follow traffic rules on intersections

'oncoming' - same as above

'left' - same as above

'next\_waypoint' - determines where he should go

'right' was left out because it doesn't matter to follow traffic rules

'deadline' was left because there are too much possible states and it seems unrealistic that agent could learn them in 100 runs

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

Agent behaves much more intelligently. It quickly learns right policies and it looks like it follows traffic rules

*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?*

Tuned hyperparameters: learning rate (alpha), discounting rate (gamma) and random action selection rate (epsilon)

Results sorted by net reward (for 100 trials):

alpha	gamma	epsilon	% of successes	net reward
0.15	0.9	0.15	0.92	2252
0.2	0.8	0.1	0.91	2247
0.15	0.9	0.05	0.94	2235.5
0.1	0.7	0.2	0.92	2229.5
0.1	0.8	0.05	0.92	2215.5
0.2	0.6	0.15	0.96	2202.5
0.05	0.8	0.15	0.98	2198
0.05	0.7	0.2	0.93	2194

0.15	0.9	0.1	0.91	2182.5
0.15	0.6	0.15	0.89	2174
0.2	0.7	0.1	0.95	2166.5
0.2	0.8	0.2	0.89	2157.5
0.1	0.8	0.15	0.93	2156
0.05	0.6	0.2	0.9	2142.5
0.15	0.7	0.2	0.88	2141
0.2	0.7	0.15	0.89	2134
0.15	0.8	0.15	0.92	2132.5
0.05	0.9	0.15	0.94	2129
0.1	0.7	0.15	0.93	2122.5
0.1	0.9	0.2	0.93	2116.5
0.1	0.6	0.2	0.89	2116
0.2	0.7	0.2	0.9	2109.5
0.2	0.8	0.05	0.94	2105.5
0.05	0.9	0.2	0.9	2104
0.05	0.9	0.1	0.88	2099
0.15	0.6	0.2	0.86	2091.5
0.15	0.8	0.1	0.84	2082
0.2	0.6	0.1	0.93	2077
0.1	0.6	0.15	0.92	2071
0.05	0.8	0.05	0.95	2070.5
0.15	0.8	0.2	0.92	2067.5
0.05	0.8	0.1	0.88	2053
0.15	0.7	0.15	0.8	2049
0.2	0.9	0.2	0.83	2047.5
0.05	0.6	0.15	0.89	2044.5
0.05	0.7	0.15	0.89	2043.5
0.1	0.7	0.1	0.92	2037.5
0.1	0.9	0.15	0.88	2024
0.15	0.8	0.05	0.88	2019
0.15	0.7	0.1	0.91	2007
0.2	0.6	0.2	0.88	2006.5
0.1	0.8	0.2	0.86	1994.5
0.05	0.6	0.05	0.82	1994.5
0.05	0.6	0.1	0.8	1971
0.1	0.9	0.1	0.79	1941

0.1	0.6	0.1	0.86	1925
0.2	0.8	0.15	0.84	1922
0.05	0.8	0.2	0.83	1910
0.05	0.7	0.1	0.83	1904.5
0.15	0.9	0.2	0.75	1876.5
0.15	0.6	0.1	0.92	1876
0.1	0.8	0.1	0.88	1875.5
0.2	0.9	0.05	0.82	1832.5
0.2	0.9	0.1	0.79	1831.5
0.2	0.9	0.15	0.77	1813
0.1	0.7	0.05	0.85	1750.5
0.15	0.6	0.05	0.78	1680.5
0.2	0.7	0.05	0.66	1668
0.1	0.9	0.05	0.71	1663.5
0.1	0.6	0.05	0.72	1658.5
0.05	0.9	0.05	0.64	1588
0.05	0.7	0.05	0.72	1583
0.15	0.7	0.05	0.83	1533.5
0.2	0.6	0.05	0.62	1153

*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?*

Looks like it does. Best agent (by net reward) has 92% success rate (i.e. getting to destination in time)