TRAINING A SMARTCAB

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?

- The smartcab does eventually make it to the destination, although the trial run does occasionally hit the hard time limit. Ofcourse, there is a large variation in the number of steps the car takes to get to the target and the path is nowhere near optimal.
- The edges of the grid are a little different from the rest of the intersections as the car appears on the other side of the grid after crossing beyond the end (i.e. will appear on the left side if going through the wall on the right). This may cause some problems in training as the optimal path may involve using this to the cars advantage, but the intermediate rewards may not reflect this as it may register as moving away from the target in the short run.

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? 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?

- When considering all possible states in the environment, different factors to consider:
  - 48 different intersections
  - Two types of lights, red and green.
  - Four possible headings of the car, North, South, East, West.
  - Oncoming traffic going left, right, forward or not there.
  - North-South difference to target: 5 possibilities
  - East-West difference to target: 7 possibilities
  - Time Remaining

  This leads to a total of 53760 unique states at any given moment, ignoring time remaining, which would increase the possible states even more. However, not all these states need to be considered when training the smartcab.

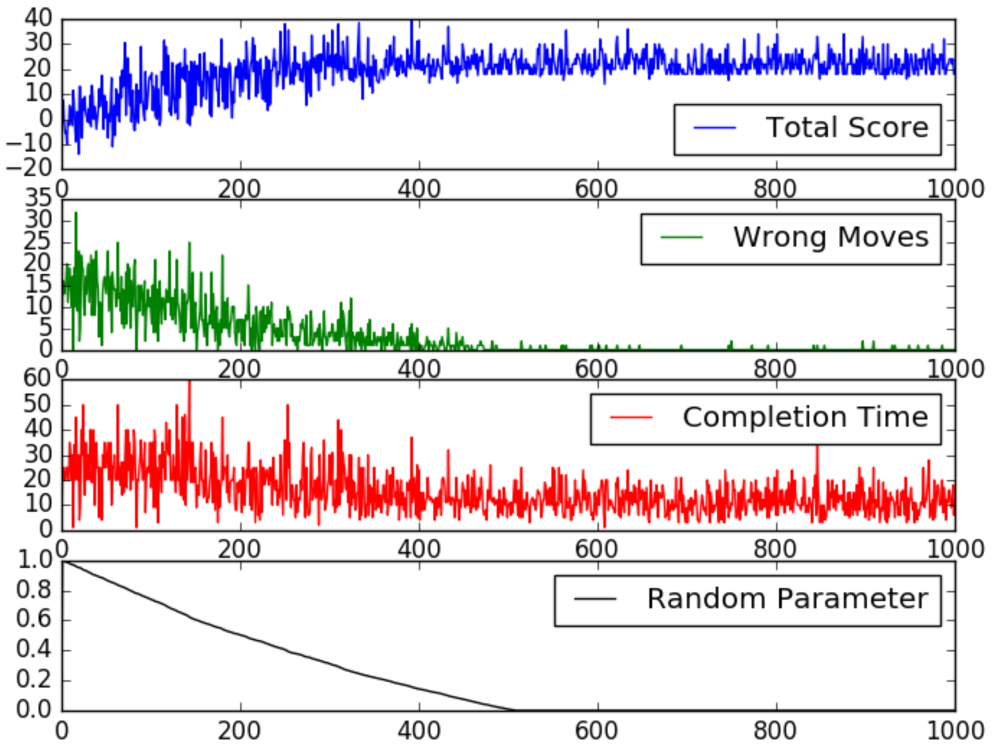
- In our driving environment, the goal of the smartcab is to get to the target destination in the shortest amount of time while obeying all the rules of the road. In order to do this, the car must have certain amount of minimum information. For instance, in order to find out the direction we need some sort of heading, this is provided by the waypoint dictated by the planner. Once we have this heading, we don’t require information on how far the target is or which way we are headed, the appropriate move is to follow this heading and the smartcab should learn to do so. Next, we require information on the traffic lights at the relevant intersection so that the cab can determine whether to proceed or wait. Finally, we need information on other traffic that may be present at the same intersection. For instance, if we are to make a left and there is an oncoming vehicle going straight through, we need to yield. Similar exceptions apply to making a right turn on a red or if the oncoming car was turning right. As a result, so far the relevant factors to consider include:

  - 3 different directions dictated by the planner (forward, left, right).
  - Two types of lights, red and green.
  - Other traffic, coming from the left, right or oncoming. Each one of those could be going left, right or forward, if there at all.

  Using just these possible factors, we get a total of 384 (3x2x4x4x4) possible states at any given intersection. Another possible factor to consider is the time remaining for each trial run (the 'deadline'). Adding time remaining would increase the state space significantly as the program would interpret each of the possible states mentioned above as unique even if all that has changed is the time remaining. Since the objective is to reach the destination as soon as possible, it does not matter how much time is remaining since the smartcab should make the optimal choice, obeying all traffic rules, regardless of the time remaining. As a result, we shall ignore the time remaining variable.
- The number of states is still quite large and we may require additional trials to reach them. However, the Q-learner will still be able to learn how to respond to states that occur more frequently and therefore perform well under most circumstances. As more states are visited, it should learn the appropriate moves under all possible scenarios.

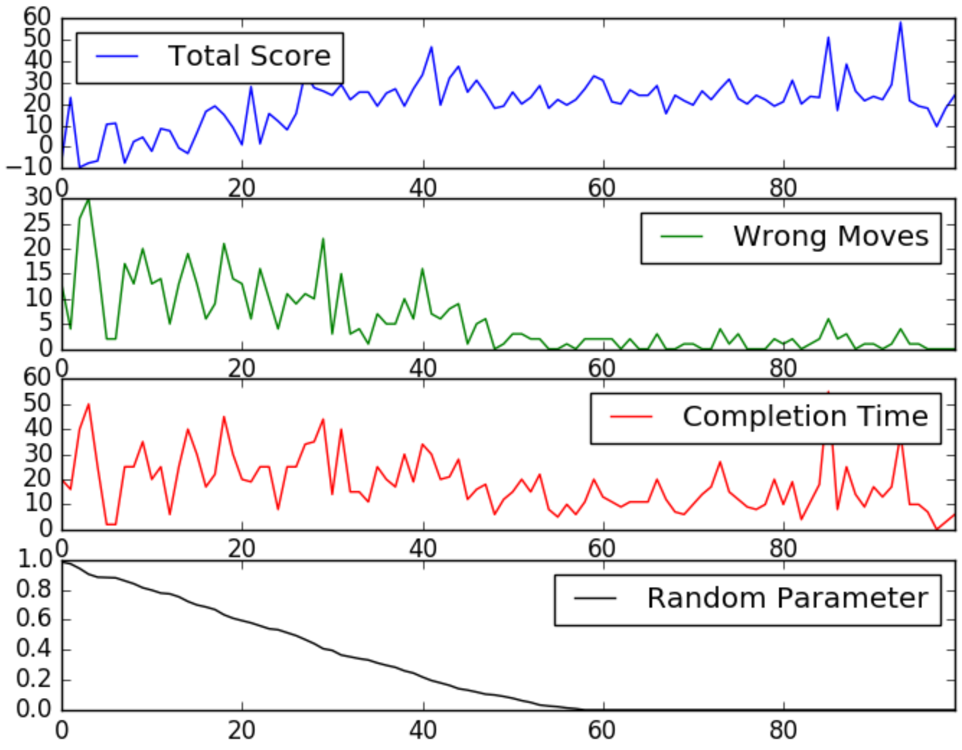
What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

- The smartcab starts off the same and appears to exhibit random behaviour; it has a lot of wrong moves and times out very frequently. However, as epsilon, the random moves parameter, decreases, the smartcab performance improves until it is performing well, reaching the target consistently and not making wrong moves:

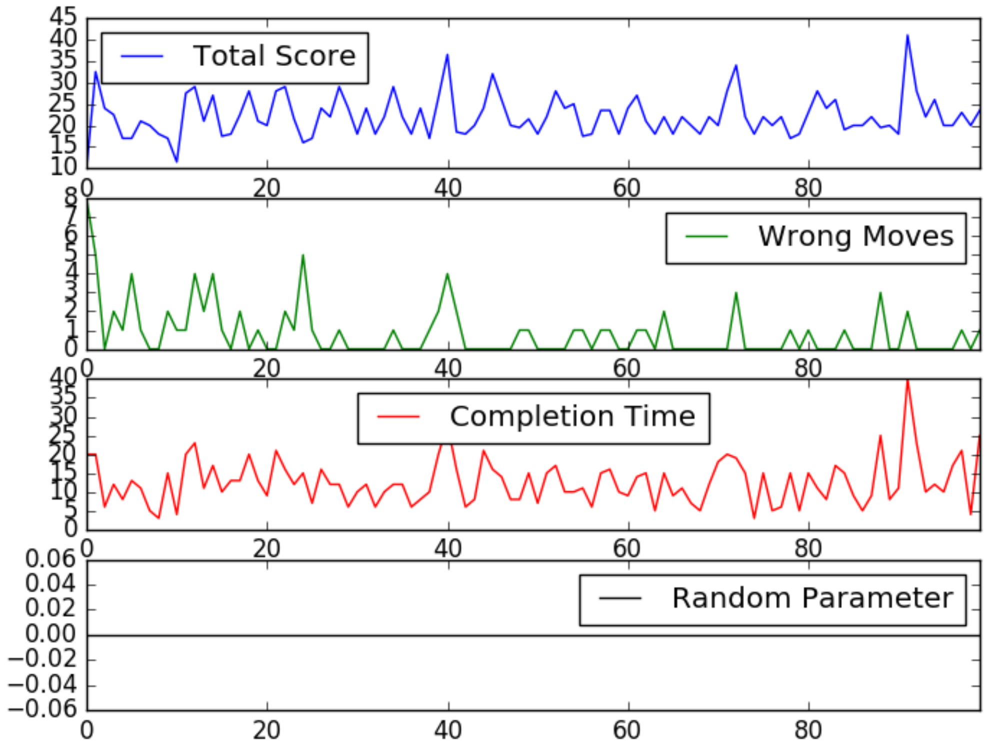


Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

- We now switch over to 100 trials for simplicity. The following uses a discount rate of 0.5, learning rate of 0.5, and a consistently decreasing random parameter to try and illustrate the progression of a greedy limit infinite exploration algorithm:



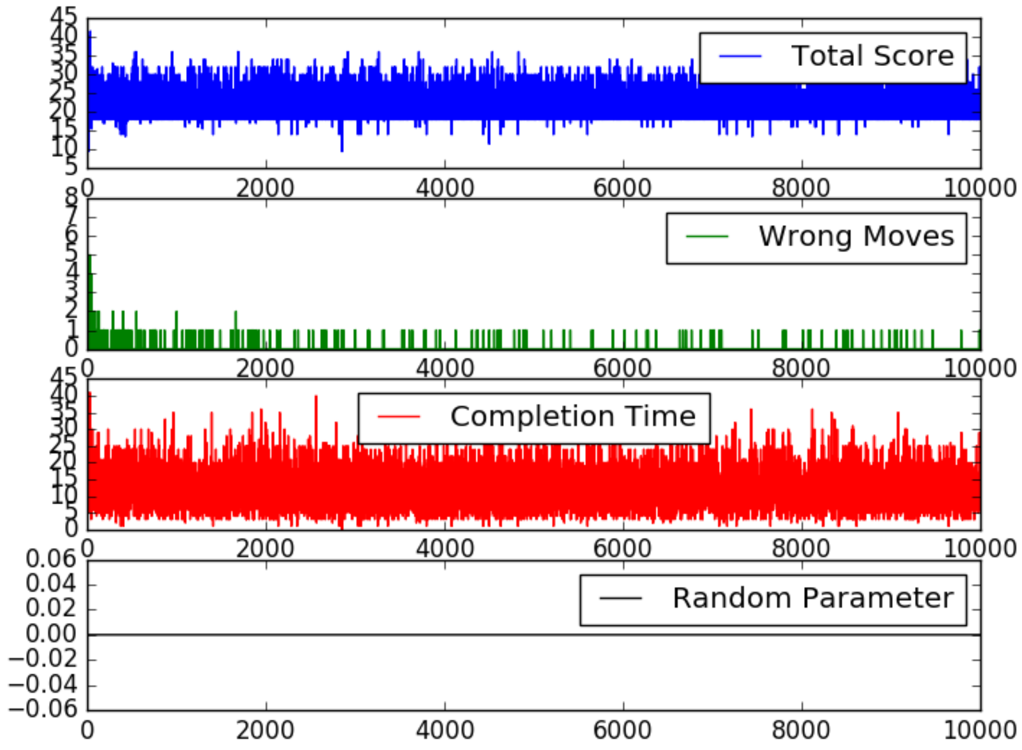
- ➢ In this particular environment, there is always only one correct move and as such, exploration is not needed and actually prevents the smartcab from improving its performance even further. Also, we don't need to look at the potential rewards from future states when considering the current action as the planner's moves are always the correct move. As a result, we can set the random parameter to 0, along with a discount rate of 0 and a learning rate of 1. This essentially results in storing the entire reward of each state and action combination the cab tries and assuming that it stays the same (which is the case in our current game):



- In the last graph, we don't see the same type of improvement as the last but that is because the agent learns the commonly occurring states very fast. Even in the very first trial run, after a few random moves the smartcab starts progressing towards the target. Eventually also learning the rules of the road.

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?

- As the number of trials increases and the agent encounters new states never before seen, we get closer to the optimal policy. The following shows the results for n=10000 with the optimal parameters:



In this case, a total of 224 of the 384 possible states were explored, and the cab only makes the occasional mistake, likely due to encountering states for the first time. As a result we can be quite confident that the policy estimate is close to optimal.

- The optimal policy for this problem essentially follows the planner's direction while obeying the rules of the road and yielding when appropriate. This is essentially what the reward structure attempts to show. From our 10,000 trial experiment we have the following policy for making a right turn at a red light when the car to our left is going forward:

Trial	waypoint	light	oncoming	right	left	act_None	act_forward	act_left	act_right
39	right	red	right	None	forward	0	-1	-1	0
95	right	red	None	None	forward	0	-1	-1	-1
99	right	red	None	right	forward	0	-1	0	0
104	right	red	None	forward	forward	0	-1	0	-1
135	right	red	left	None	forward	0	-1	-1	-1
167	right	red	forward	None	forward	0	-1	-1	0
273	right	red	None	left	forward	0	-1	-1	-1

In this table, we can see that making a right turn at a red light has a negative reward and since the program chooses the maximum value it would choose to not act (if there is a tie then it randomly chooses). This is the correct behaviour and shows that we are converging towards the optimal policy. Some of the 'act_right' values are 0 and indicate that we haven't approached a scenario where the cab made a right and received a negative reward yet.

- As a follow up, in all other cases we can see that making a right turn has a positive value:

Trial	waypoint	light	oncoming	right	left	act_None	act_forward	act_left	act_right
27	right	red	left	None	right	0	-1	0	2
31	right	red	forward	forward	None	0	0	0	2
33	right	red	None	right	left	0	0	0	2
53	right	red	None	forward	left	0	0	0	2
58	right	red	None	None	None	0	-1	-1	2
61	right	red	right	right	None	0	-1	0	2
67	right	red	right	forward	None	0	0	0	2
73	right	red	None	None	left	0	0	-1	2
77	right	red	forward	None	left	0	-1	-1	2
93	right	red	None	right	None	0	0	-1	2
97	right	red	None	left	left	0	0	0	0
98	right	red	forward	left	left	0	0	0	0
101	right	red	left	None	None	0	-1	-1	2
107	right	red	None	forward	right	0	0	0	2
122	right	red	forward	left	None	0	0	0	2
127	right	red	right	forward	left	0	0	0	0
134	right	red	None	None	right	0	-1	-1	2
139	right	red	right	left	None	0	-1	-1	0
159	right	red	forward	None	right	0	0	-1	2
166	right	red	forward	forward	left	0	0	-1	0
168	right	red	None	forward	None	0	-1	0	2
170	right	red	left	forward	None	0	0	-1	2
178	right	red	None	left	right	0	-1	-1	2
180	right	red	right	None	left	0	0	0	0
201	right	red	left	left	None	0	0	-1	2
205	right	red	left	right	None	0	-1	0	2
214	right	red	left	left	left	0	0	0	0
231	right	red	right	None	None	0	0	-1	2
235	right	red	left	None	left	0	0	0	2
239	right	red	None	left	None	0	0	0	2
241	right	red	forward	right	None	0	0	0	2
259	right	red	right	None	right	0	-1	-1	2
278	right	red	forward	None	None	0	-1	0	2

- For left turns at green lights, we have:

Trial	waypoint	light	oncoming	right	left	act_None	act_forward	act_left	act_right
6	left	green	forward	left	right	0	-0.5	-1	-0.5
23	left	green	right	None	forward	0	-0.5	-1	0
30	left	green	right	None	right	0	-0.5	-1	-0.5
32	left	green	forward	forward	None	0	0	-1	-0.5
40	left	green	forward	None	None	0	-0.5	-1	-0.5
47	left	green	forward	None	right	0	-0.5	-1	-0.5
51	left	green	forward	left	forward	0	-0.5	0	0
63	left	green	right	right	None	0	-0.5	-1	-0.5
116	left	green	forward	None	forward	0	-0.5	-1	-0.5
157	left	green	right	forward	None	0	0	-1	-0.5
160	left	green	right	right	forward	0	-0.5	0	0
191	left	green	forward	forward	left	0	0	-1	0
192	left	green	forward	left	None	0	-0.5	-1	-0.5
194	left	green	forward	right	left	0	0	0	0
208	left	green	right	left	left	0	0	-1	0
210	left	green	right	None	None	0	-0.5	-1	-0.5
212	left	green	forward	None	left	0	0	-1	-0.5
254	left	green	right	None	left	0	-0.5	-1	0
268	left	green	right	left	None	0	-0.5	-1	-0.5

As we can see, acting left is always (with couple of exceptions due to incomplete exploration) negative in the cases where the oncoming traffic is going forward or making a right turn.

- In summary, as the number of trials goes to infinity, we can be confident that the policy will converge to the optimal policy.