Implement a Basic Driving Agent

QUESTION: 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?

I let the simulator run for 23 runs. 21 out of the 23 runs the cab was not able to reach its destination when taking random actions. 2 out of the 23 runs the cab successfully reached its destination within the deadline of 100 moves. I notice that as the cab gets to the edge of the map it will wrap around to the other side. I also noticed that the cab will obey the traffic lights, even when moving randomly, the cab is not allowed to make illegal turns.

To make the random action I modified the update function to the following:

```
def update(self, t):
    # Gather inputs
    self.next_waypoint = self.planner.next_waypoint()  # from route planner, also displayed by simulator
    inputs = self.env.sense(self)
    deadline = self.env.get_deadline(self)

    # TODO: Update state

    # TODO: Select action according to your policy
    actionList = [None, 'forward', 'left', 'right']
    action = random.choice(actionList)

    # Execute action and get reward
    reward = self.env.act(self, action)

    # TODO: Learn policy based on state, action, reward

    print "LearningAgent.update(): deadline = {}, inputs = {}, action = {}, reward = {}".format(deadline, inputs, action, reward)  # [debug]
```

Inform the Driving Agent

QUESTION: 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 states that are appropriate for modeling the smartcab and environment are light, oncoming, right and left. Each of the states may take on the following values:

light = red or green
oncoming = None, forward, left or right
right = None, forward, left or right
left = None, forward, left or right
next waypoint = None, forward, left or right

Each of these states is appropriate for the problem because the cab needs to be aware of the light being green or red, and it needs to know if other agents are around it as well as which direction they are turning. The color of the light as well as location of nearby agents needs to be accounted for because the cab needs to follow the law and should avoid making illegal turns. Lastly, next waypoint feature is needed so that the cab moves closer to its destination.

I did not include the the deadline and location variables as a state. Deadline was not included because this variable ranges approximately from 60 to 0. This would increase the possible states from 512 values (when only accounting for light, oncoming, right, left, and next waypoint) to 30,720 possible states. This will considerably slow down the learning time of the cab. Similarly, location was not included because many locations would need to be included in the state space. This would further increase the amount of time the cab needs to learn effectively.

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?

There exists 512 possible states. 512 states seems like a reasonable number. I would expect the states that will be used most often will be those with either red or green lights and a forward, left, or right value for next waypoint and oncoming, right, and left states set to None.

Implement a Q-Learning Driving Agent

QUESTION: 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?

My initial values for the Q-learning algorithm were:

alpha = 0.1
gamma = 0.1
epsilon = 0.1
Q(s,a) initialized to all 0's

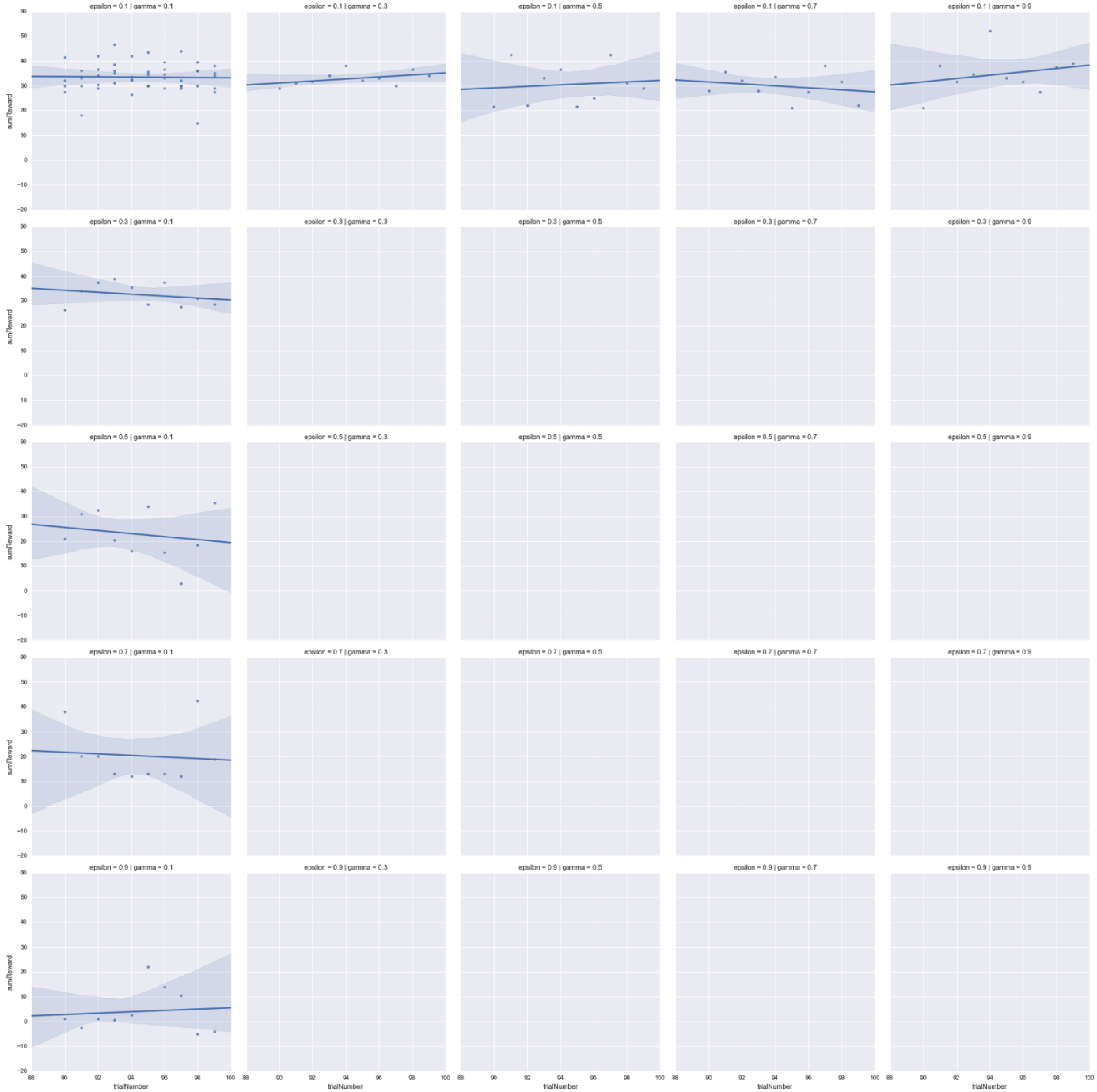
The cab seemed to have learned in several phases. The first phase involved the cab learning what actions caused negative rewards. After several iterations the cab learned to not make these moves and instead prioritized taking a 'None' action, instead of an action that would give it a negative reward. The cab would then enter the next phase of learning which was to sit idle, until the random action policy (which is determined by the epsilon) is activated. If the random action caused a positive reward then the cab increasingly made more actions like this. The final phase was the cab making good decisions, which allowed it to reach its destination relatively quickly, while minimizing negative rewards.

Improve the Q-Learning Driving Agent

QUESTION: 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?

I used alpha (learning rate), gamma (discount factor), and epsilon (random movement rate) to tune the Q-learning algorithm. The learning rate would decay at a rate of 1.0/np.log(i)-0.15, where i is each round. This causes the algorithm to heavily weight new information in early rounds. As more rounds progress, the learning rate is reduced and we are solidifying information that was learned in previous rounds.

I had used various values for the discount factor (gamma) and the random movement rate (epsilon). A gamma of 0.9 and epsilon of 0.1 produces the best performing smart cab. The plot below shows the summed reward for rounds 90 through 100. These are the rounds where the algorithm has had ample time to learn and should be performing at its best. With a high gamma value, the algorithm is striving for higher long term rewards. This correlates with the higher total reward seen with larger values of gamma.



The random movement rate, epsilon, determines for a given action if the algorithm will perform its desired move or if it will take a random action. This is important in early rounds of learning, as a random action may lead to a higher reward. However, if the random action rate is too high then these actions may lead to illegal moves made, even though the algorithm is attempting to make the correct action. The chart below shows as epsilon increases the total reward decreases.

The final driving agent has an epsilon of 0.1 and gamma of 0.9. The bottom chart shows the total rewards for the last 10 rounds.

QUESTION: 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 had the following parameters:
epsilon = 0.1
gamma = 0.9

In the last 10 rounds, the cab minimized its time to reach its destination and incurred penalties when a random decision was made. The random decision has a 10% probability of occurring, and even if it does occur, there is some probability that the decision it made will either occur in a negative, neutral or positive reward. The rules of the road were followed and the cab did not make any illegal turns. An example of this is in the bottom table where the light remains red. When the light is red, the cab does not take the action to turn, as this would result in an illegal move. Also, the cab makes a direct route to the destination and does not make circles or unnecessary moves. There was 1 instance in trial 93 and deadline 16, where the cab made an illegal move. There was oncoming traffic from the right and a red light. The cab seems to be confused. It likely hasn't encountered this state very often and the cab guesses incorrectly on what the optimum move should be.

An optimal policy would be one where the probability of random decisions slowly lowers as the algorithm learns more. This allows the algorithm to learn new strategies in the beginning and at later rounds there is less and less penalty from making random decisions. This can be down by decaying epsilon as learning progresses.

Below is the output from the last 10 trials of the optimal learning strategy:

Key	Value
	Red Light
	Green Light
N	None
r	Right
f	Forward
l	Left

Trial	Deadline	Light	Oncoming	Right	Left	isRandom	Action	NextWaypoint	Reward
90	30		N	N	N	0	r	r	2
90	29		N	N	N	0	r	r	2
90	28		N	N	N	0	N	f	0
90	27		N	N	N	0	f	f	2
90	26		N	N	N	0	l	l	12
91	30		N	N	N	0	N	l	0
91	29		N	N	N	0	N	l	0
91	28		N	N	N	0	N	l	0

91	27		N	N	N	0	N	l	0
91	26		N	N	N	0	N	l	0
91	25		N	N	N	0	l	l	2
91	24		N	N	N	0	N	f	0
91	23		N	N	N	0	N	f	0
91	22		N	N	N	0	N	f	0
91	21		N	N	N	0	f	f	2
91	20		N	N	N	0	f	f	2
91	19		N	N	N	0	f	f	2
91	18		N	N	N	0	N	l	0
91	17		N	N	N	0	N	l	0
91	16		N	N	N	1	N	l	0
91	15		N	N	N	0	l	l	2
91	14		N	N	N	0	N	f	0
91	13		N	N	N	0	N	f	0
91	12		N	N	N	0	N	f	0
91	11		N	N	N	0	N	f	0
91	10		N	N	N	0	f	f	12
92	30		N	N	N	0	f	f	2
92	29		N	N	N	0	f	f	2
92	28		N	N	N	1	l	f	-0.5
92	27		N	N	N	0	r	r	2

92	26		N	N	N	0	N	f	0
92	25		N	N	N	0	N	f	0
92	24		N	N	N	0	f	f	2
92	23		N	N	N	0	r	r	2
92	22		N	N	N	0	N	f	0
92	21		N	N	N	1	l	f	-1
92	20		N	N	N	0	f	f	2
92	19		N	N	N	0	N	f	0
92	18		N	N	N	0	N	f	0
92	17		N	N	N	0	N	f	0
92	16		N	N	N	0	N	f	0
92	15		N	N	N	0	f	f	12
93	20		N	N	N	0	f	f	2
93	19		N	F	N	0	f	f	-1
93	18		N	N	N	0	N	f	0
93	17		N	N	N	0	N	f	0
93	16		N	N	N	1	N	f	0
93	15		N	N	N	0	f	f	2
93	14		N	N	N	0	N	l	0
93	13		N	N	N	0	N	l	0
93	12		N	N	N	0	N	l	0
93	11		N	N	N	0	l	l	2

93	10		N	N	N	0	N	f	0
93	9		N	N	N	0	N	f	0
93	8		N	N	N	0	f	f	12
94	30		N	N	N	0	l	l	2
94	29		N	N	N	1	r	f	-0.5
94	28		N	N	N	0	N	l	0
94	27		N	N	N	1	l	l	-1
94	26		N	N	N	0	N	l	0
94	25		N	N	N	0	l	l	2
94	24		N	N	N	0	f	f	2
94	23		N	N	N	0	r	r	2
94	22		N	N	N	0	f	f	12
95	25		N	N	N	0	r	r	2
95	24		N	N	N	0	f	f	2
95	23		N	N	N	0	N	f	0
95	22		N	N	N	0	N	f	0
95	21		N	N	N	0	f	f	2
95	20		N	N	N	0	f	f	2
95	19		N	N	N	0	l	l	12
96	20		N	N	N	0	r	r	2
96	19		N	N	N	0	r	r	2
96	18		N	N	N	0	N	f	0

96	17		N	N	N	0	N	f	0
96	16		N	N	N	0	N	f	0
96	15		N	N	N	0	f	f	2
96	14		N	N	N	0	N	l	0
96	13		N	N	N	0	N	l	0
96	12		N	N	N	0	N	l	0
96	11		N	N	N	0	l	l	12
97	20		N	N	N	0	f	f	2
97	19		N	N	N	0	f	f	2
97	18		N	N	N	0	r	r	2
97	17		N	N	N	0	f	f	12
98	20		N	N	N	1	r	l	-0.5
98	19		N	N	N	0	N	f	0
98	18		N	N	N	0	N	f	0
98	17		N	N	N	0	N	f	0
98	16		N	N	N	0	f	f	2
98	15		N	N	N	0	N	f	0
98	14		N	N	N	0	f	f	2
98	13		N	N	N	0	f	f	12
99	20		N	N	N	0	l	l	2
99	19		N	N	N	0	N	f	0
99	18		N	N	N	0	N	f	0

99	17		N	N	N	0	f	f	2
99	16		N	N	N	0	N	f	0
99	15		N	N	N	1	r	f	-0.5
99	14		N	N	N	0	l	l	12