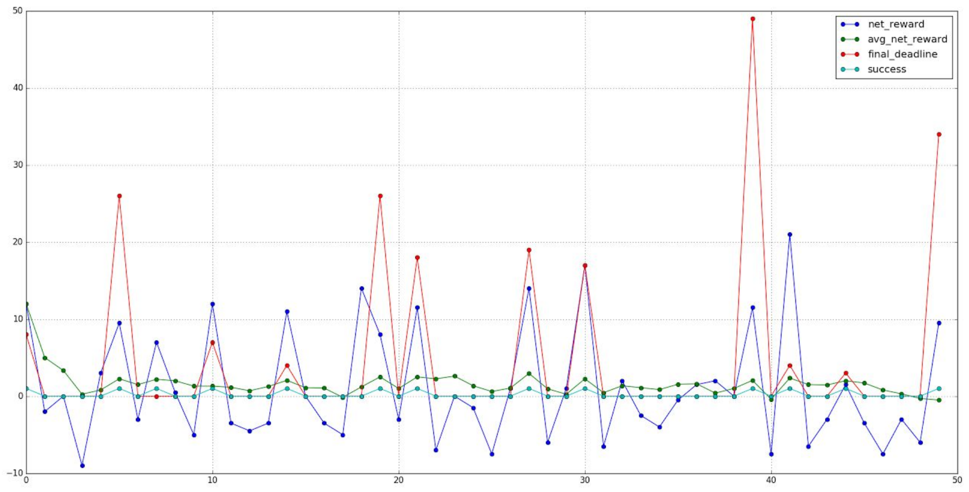
Implement a Basic Driving Agent

QUESTION: Observe what you see with the agent's behavior as it takes random actions. Does thesmartcab eventually make it to the destination? Are there any other interesting observations to note?

Yes, the smartcab makes it to the destination, but it usually takes a lot of time. Due to the random movement, the car is sometimes very fast, but sometimes it moves away from the destination even though it's already close.

Below is a plot of the rewards, final deadline and success of the algorithm for the first 50 trials (enforce_deadline is enabled here to make it more comparable to the other plots below). One can see that the random policy is rarely successful (cyan line at 1) and that the reward fluctuates around 0 (blue line).



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?

I model each state as a combination of several variables: 1) the next waypoint ('left', 'right' or 'forward'), 2) the color of the light ('red' or 'green'), 3) oncoming traffic, 4) left traffic, 5) right traffic (in each case 'left', 'right', 'forward' or None). The values are converted to integers (e.g. 'red' = 0, 'green' = 1), so that the state is described by a tuple of five numbers. This tuple can then be used directly as an index for the numpy array that stores the Q values. I decided to omit the time to deadline from the state representation because the smartcab shouldn't change its strategy if there's less time left (e.g., it shouldn't drive "riskier" or cause accidents just because the deadline is approaching). Instead, the time efficiency of the smartcab is enforced by discounting the rewards.

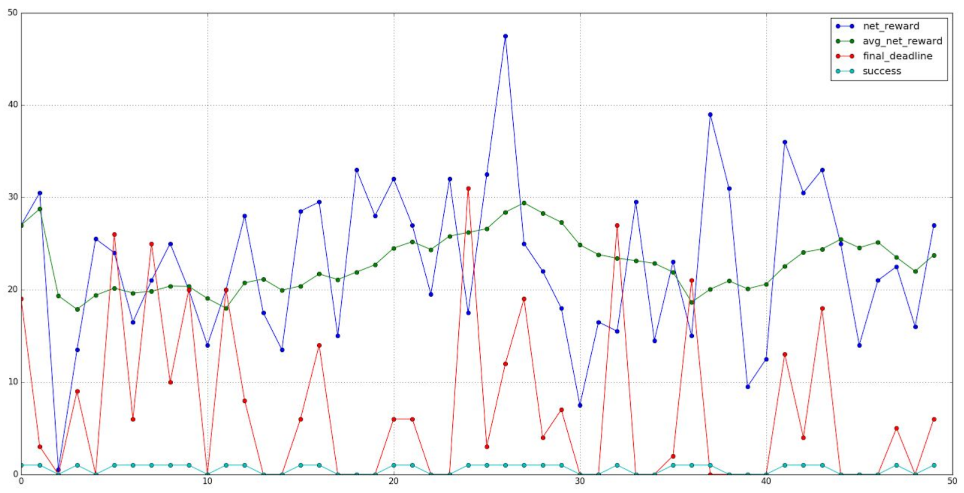
OPTIONAL: How many states in total exist for the smartcab in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?

Combining the values of the state variables detailed above, there are 3*2*4*4*4 = 384 possible states. As the deadline for this setting is 20 time steps, it should take the algorithm a reasonable number of trials until it observes all states (and learns the optimal behavior).

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?

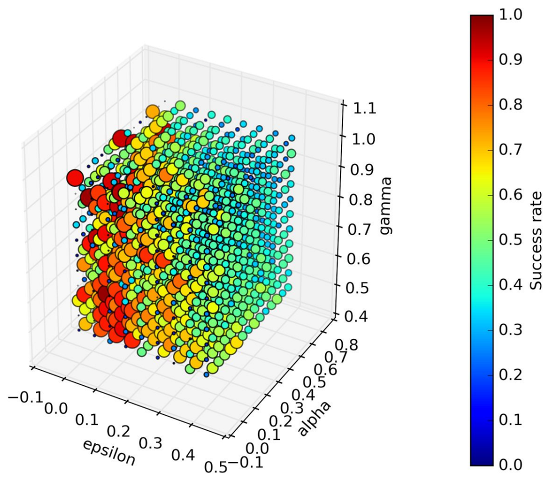
The behavior of the smartcab is way better: It moves more or less straight to its destination without taking too many unnecessary turns, it waits at red lights, and yields to other cars. Apparently, it has learned this policy from observing similar situations over and over again, and trying to maximize reward. However, the cab has not yet reached an ideal policy - sometimes, it takes visible detours, or it gets stuck in a situation and visits the same crossings multiple times.

The plot below shows the same values as the plot above, this time for (unoptimized) Q learning with epsilon = 0.2, alpha = 0.3, gamma = 0.85. The car is successful most of the time (though not always), and the reward is clearly positive (in contrast to the ~0 reward with a random policy). Interestingly, the algorithm did not take many trials to learn this policy (which indicates that it observes a large number of different states even in a single trial).

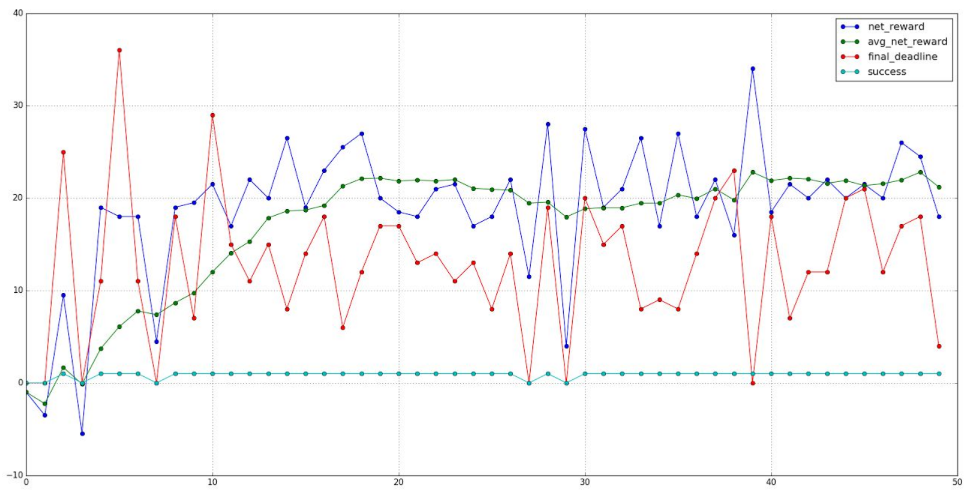


QUESTION: Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

To optimize the hyperparameters, I performed a grid search over the Q learning parameters (epsilon from 0 to 0.4, alpha from 0 to 0.7, gamma from 0.5 to 1; each in steps of 0.05; 100 trials per parameter combination). Here are the results, plotted in the 3-dimensional parameter space (size and color indicate success rate):



The highest success rate was observed at epsilon = 0.05, alpha = 0.10, gamma = 0.60. With this setting, the algorithm completed 98 % of the trials successfully and achieved a mean reward of 21.82 per trial. The plot below shows the metrics with this parameter setting for the first 50 trials. In comparison to the unoptimized driving agent, it takes more trials to achieve a high reward, but is then successful most of the time.



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?

To assess the driving behavior, I observed the optimized agent after ~50 trials. Its policy is very good, but not completely optimal. With minor exceptions, the smartcab takes the direct route to the destination. However, sometimes it gets stuck in a situation where it “circles” the same crossings multiple times (compare the unoptimized agent above). A higher exploration rate (epsilon) might mitigate this problem, but might also lead to more small detours (i.e. taking the wrong direction at a single crossing). Penalties for breaking traffic rules occur very rarely.

An optimal policy should prioritize not causing any accidents, and secondly taking a minimum amount of time to the destination.