Project 4 – Train a smartcab

Yaron Blinder

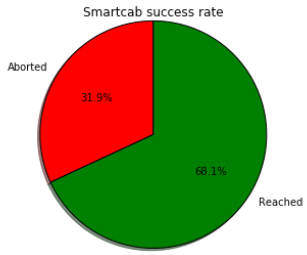
Implement a Basic Driving Agent

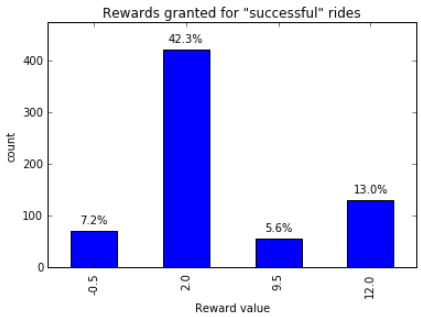
Question 1:

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?

Answer:

Over 1,000 runs, we can observe the following for the initial smartcab:





The randomly moving smartcab successfully reached its destination 31.9% of the time before reaching the hard time limit (-100), receiving one of four values as reward: -0.5, 2, 9.5 and 12. These rewards were achieved 7.2%, 42.3%, 5.6% and 13% of the time, respectively.

Hence, the average or expected reward for this smartcab can be calculated as:

$$Expected\ Reward = Success\ rate * mean\ reward\ for\ success$$

$$= 0.681 * [(0.072 * -0.5) + (0.423 * 2) + (0.056 * 9.5) + (0.13 * 12)]$$

$$= 2.902$$

Inform the Driving Agent

QUESTION 2: *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?*

In order for the smartcab to learn from the reward scheme it must take into consideration all the factors which influence rewards:

- Light – red/green
- Next waypoint – the direction suggested by the planner (forward, right, left)
- Next waypoints of traffic coming from straight ahead, right, and left. (forward, right, left, None)

Justification: Next waypoint should be taken into consideration as it gives the shortest path to the destination. The light and status of any other vehicles in the intersection define which actions are legal. Seeing as the traffic laws govern the reward scheme and are derived from the above detailed information, they should be enough to allow our model to learn appropriate driving policies.

I have chosen to omit the deadline input to avoid a situation where the learner chooses to break traffic laws in order to get to its destination faster.

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?*

The total number of possible states in this environment can be calculated as the combination of all the states for each parameter. There is one edge case where the smartcab is at the destination, and the other states can be combined by multiplication:

$$S = N_{light} * N_{next} * N_{oncoming} * N_{right} * N_{left}$$
$$= 2 * 3 * 4 * 4 * 4 + 1 = 385$$

In order for Q-Learning to learn the optimal strategy, it is required that it visits every action-state pair infinitely often. As this is not exactly feasible, we would want to at least make sure each action-state pair is visited at least a few times. For 100 runs, where each run is between 20-40 actions, it is plausible that our learner will meet this requirement. Moreover, as the penalties are governed by the very small subset of illegal actions it's possible that our learner will arrive at a near-optimal strategy very quickly.

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?*

Upon implementing the Q-Learning algorithm, the agent began to show improvements in performance after the first few trials, as evidenced by a reduction in penalty-incurring actions and overall reward. Early trials seem like random exploration, which is to be expected as the Q table is initialized with random numbers. For this reason, there is a large number of penalties in early trials. Later trials show less randomness with only rare penalties, probably due to the randomness of the step selection defined by epsilon (which has an initial value of 0.1). This randomness can also explain the fact that the agent may sometimes drive in circles, even in later runs.

This is the expected result of implementing reinforcement learning, as the estimated Q function changes due to reward-related feedback and in turn dictates future action policy.

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?*

For Epsilon = 0.1, I ran a parametric sweep over Gamma and Alpha with the values 0.1,0.3,0.5,0.7,0.9. For each combination of parameters, the performance of the agent was measured by calculating the expected reward over 100 runs three times, and averaging that result. The calculated results are shown in the table below:

		Alpha:				
		0.1	**0.3**	**0.5**	**0.7**	**0.9**
Gamma:	**0.1**	11.583	11.543	11.503	11.743	11.352
	0.3	10.967	11.263	11.232	11.367	11.43
	0.5	9.895	10.933	10.875	11.287	11.343
	0.7	8.2433	10.053	9.75	10.845	10.133
	0.9	9.335	8.6617	9.0767	8.26	7.325

The best result is achieved by the combination (Alpha = 0.7, Gamma = 0.1). Reducing Epsilon can somewhat improve these results.

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?*

It is reasonable to state that the agent achieved a near-optimal policy, as the expected reward per ride nears 12 (with a perfect 12 meaning every move was correct and the destination was reached before the deadline). Our optimized parameters yield an expected reward of around 11.75 - very near the optimum, using only 100 learning rides.

An optimal policy for this problem would be one which moves in the direction suggested by the planner every time, as long as that move is legal.