Machine Learning Nanodegree - Project 04

Pedro M Duarte
pmd323@gmail.com
October 29, 2016

I. Implement a Basic Driving Agent

Q: *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?*

A: To answer this question I implemented the `RandomAgent` class. In addition to selecting a random action, this class saves a counter for each transition observed in the system. A transition is defined in this case as the tuple given by

- next waypoint
- intersection state (state of traffic light and oncoming traffic)
- action selected
- reward

With a random action choices the smartcab starts random walking over the grid and eventually makes it to the destination.

It was interesting to keep track of the unique transitions that the agent performed. I ran the simulator 10 times with a setting of `num_dummies=20`. The large number of dummy agents created more complex traffic conditions and increased the possibility of having the smartcab encounter other traffic at intersections.

Below are some notable observations I made after analyzing the simulation results:

- If the action selected violates traffic rules, a reward of -1.0 is given to the agent.
- If the action selected respects traffic rules but does not match the next waypoint, a reward of -0.5 is given to the agent.
- If the action selected respects traffic rules and matches the next waypoint, a reward of +2.0 is given to the agent.
- Whenever the selected action violates traffic rules, the simulator rejects the action and the agent does not move in the grid.
- Even with 20 dummy agents, the most commonly encountered intersection state is that one where the smart cab is by itself at the intersection (no other traffic).
- The situation where there are two dummies at the intersection was encountered a total of 25 times my in simulation, compared with 192 times where the smartcab was with only one dummy at the intersection, and 481 times the smartcab was by itself at the intersection. (Keep in mind that these numbers are for `num_dummies=20`)

II. Inform the Driving Agent: Identify and Update State

Q: *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?*

A: Based on the study of the simulations performed with the `RandomAgent` I was able to identify the most common states encountered by the agent. The number of states to consider for the Q learning algorithm is important. If we pick too many states we will require a lot of observations to be able to learn. On the other hand, if we reduce the situation to only a handful of states, then the agent might not be able to have enough information to distinguish between some kinds of actions, and this will result in the smartcab taking some actions that lead to negative rewards.

For my implementation of the Q learning algorithm I will use the following variables to define the possible states:

- is light green (boolean)
- waypoint (3 possibilities)
- car oncoming (4 possibilities: None, left, right, forward)
- car left (boolean)

Considering the cardinality of the state variables outlined above, we have a total of $2 * 3 * 4 * 2 = 48$ possible states to consider. The most important piece of information that I have decided to leave out is whether or not there is a car coming from the right. Based on the U.S. Right-of-way rules, once the state of the light is known (red or green) the presence or absence of a car coming from the right has no effect on our decision to turn.

The second piece of information that has been deliberately left out of the implementation is the deadline information. The cardinality of the deadline variable can very large (the deadline is set to 5 times the starting distance to the destination), and so including the deadline value in the state representation will dramatically increase the space we need to explore to start reaping the benefits of the Q-matrix convergence.

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?*

A: If we consider all possible variables, we would have a state

representation as follows:

- light state (red or green)
- waypoint (3 possibilities)
- car on left (4 possibilities: None, turning left, turning right, going forward)
- car on right (4 possibilities)
- car oncoming (4 possibilities)
- deadline (on the order of at least 20 possibilities on average)

That makes up a total of 2 * 3 * 4 * 4 * 4 * 20 = 7680 possible states!! Some of the states covered by these possibilities are much less likely to occur than others, and Q-learning will take a lot longer to converge if there are states that are not really seen very often in practice. As was outlined in the previous section, a better approach is to consider a subset of only the most significant states to allow for faster convergence of the Q-learning algorithm.

Another possible approach that might help with large cardinality variables such as the deadline would be to define just a few categories to bin the variable values into. For example one can boil it down to deadline being *urgent* or *not-urgent* depending on a threshold value.

III. Implement a Q-Learning Driving Agent

The Q-Learning driving agent was implemented with the following properties:

A. Q-matrix initialization

All of the entries in the Q-matrix were initialized to a random value between 0 and 1.

B. Learning rate decay

The learning rate was given the dependency

$$\alpha_t = \alpha_0 \Theta / (\Theta + t) \tag{1}$$

where the variable Θ represents an offset parameter that can be used to control the decay of the learning rate, and the variable t is the total number of transitions seen by the agent over its entire lifetime (not just on one trial). This dependency satisfies the criteria for convergence of the Q-matrix, but is designed to decay slowly so that we continue to have a sizable learning rate even after several hundred transitions have been observed by the smartcab.

C. Exploration versus exploitation

For the exploration probability, `epsilon`, I also chose a $1/x$ decay

$$\epsilon_T = \epsilon_0 \Theta_\epsilon / (\Theta_\epsilon + T) \tag{2}$$

but in this case the progress variable T is defined as the total number of successful trips the agent has completed. I incorporated this definition based on the helpful comments from my first project review.

α_0	Θ	ϵ_0	Θ_ϵ	success_rate	penalty_rate	objective
0.1	10000.0	0.2	3	1.0	0.046174	0.907
0.1	10000.0	0.2	3	1.0	0.078642	0.842
0.1	10000.0	0.2	3	0.97	0.070211	0.829
0.1	10000.0	0.2	3	0.98	0.052143	0.875
0.1	10000.0	0.2	3	0.98	0.068798	0.842

TABLE I
Results for initial values of Q-learning parameters.

D. Summary of Q-learning parameters

Below is a summary of the Q-learning parameters along with the values that were picked for the first implementation of the Q-learning algorithm:

- Q matrix random initialization interval: [0, 1]
- Learning rate starting value ($\alpha_0 = 0.1$)
- Learning rate decay offset parameter ($\Theta = 10^4$)
- Exploration rate starting value ($\epsilon_0 = 0.2$)
- Exploration rate decay offset parameter ($\Theta_\epsilon = 3$)

Optional:*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?*

A: The main difference after implementing Q-learning is that the agent now reaches the destination for the vast majority of the trials!! On the first few trials the rate of success is low, as the smartcab is learning and the Q matrix is only starting to converge. But after only a handful of trials the smartcab has seen enough transitions and has learned how to get to the destination while obeying traffic rules.

I ran the simulation five times (100 trials each time) with the Q learning parameters outlined in the previous section.

The results of these five runs are shown in Table I, where the performance of the smartcab is quantified using the following parameters:

- **success_rate**: This is the number of successful trips divided by the number of trials.
- **penalty_rate**: This is the number of times a negative reward was received, divided by the total number of transitions seen by the agent.
- **objective**: This is a combination of the success rate and the penalty rate defined as

$$\text{objective} = \text{success rate} - 2 \times \text{penalty rate} \tag{3}$$

The objective is defined as a way to combine success rate and penalty rate into a single metric, while giving more importance to the penalty rate. In this way we will give some more priority to the smartcab's compliance with traffic rules.

IV. 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?*

Top 5:

α_0	Θ	ϵ_0	Θ_ϵ	success_rate	penalty_rate	objective
1.0	10000.0	0.02	3	1.00	0.020393	0.959215
1.0	10000.0	0.02	10	1.00	0.020424	0.959152
1.0	10000000.0	0.02	1	0.99	0.021739	0.946522
1.0	10000.0	0.02	1	0.99	0.023207	0.943586
1.0	10000000.0	0.02	3	0.99	0.025496	0.939008

Worse 5:

α_0	Θ	ϵ_0	Θ_ϵ	success_rate	penalty_rate	objective
0.10	1.0	0.60	10	0.18	0.618148	-1.056296
2.00	1.0	0.02	3	0.01	0.625455	-1.240909
0.10	1.0	0.20	1	0.21	0.727784	-1.245568
0.05	1.0	0.02	1	0.04	0.686051	-1.332101
0.05	1.0	0.60	10	0.13	0.745933	-1.361866

TABLE II
REALIZATION NUMBER 1.

Top 5:

α_0	Θ	ϵ_0	Θ_ϵ	success_rate	penalty_rate	objective
1.0	10000.0	0.02	1	1.0	0.021168	0.957664
1.0	10000000.0	0.02	3	1.0	0.021399	0.957202
1.0	10000000.0	0.02	1	1.0	0.023392	0.953216
1.0	10000.0	0.02	10	1.0	0.024133	0.951735
1.0	10000000.0	0.20	3	1.0	0.029412	0.941176

Worse 5:

α_0	Θ	ϵ_0	Θ_ϵ	success_rate	penalty_rate	objective
0.05	1.0	0.02	3	0.07	0.619638	-1.169277
0.05	1.0	0.60	10	0.19	0.730419	-1.270838
2.00	1.0	0.02	10	0.01	0.662504	-1.315008
2.00	1.0	0.02	3	0.08	0.856231	-1.632462
0.10	1.0	0.02	1	0.03	0.946412	-1.862824

TABLE III
REALIZATION NUMBER 2.

In order to improve the Q-Learning Driving Agent I will implement a grid search over the Q-learning parameter space. To make the search a little simpler, I will keep the Q matrix random initialization interval constant. After all, the range of the initial values only sets an overall scale factor that can also be affected via the learning rate starting value.

The remaining parameters will be varied as follows:

- α_0: [0.05, 0.1, 1, 2]
- Θ: [10, 10^4, 10^7]
- ϵ_0: [0.02, 0.2, 0.6]
- Θ_ϵ: [1, 3, 10]

The above parameter space gives us a total of 81 parameter combinations that can be tried out to improve the smartcab performance.

As was explained in the previous section, the objective function that will be used to evaluate the smartcab performance will be a combination of the success rate and the total number of negative rewards that gives a factor of 2 more importance to the penalty rate.

In tables II to IV, we show the top 5 and worse 5 parameter combinations for three runs of the grid search. The following conclusions can be made:

- Optimizing the Q-learning parameters we can achieve a nearly perfect (¿99%) success rate!
- The objective function is optimized for small initial values of the exploration rate ($\epsilon_0 = 0.02$ seems to be the best). This has to do with the fact that there is already some randomness built into the Q-matrix from the start due to our choice of initialization.
- The value of Θ for the optimal runs indicates that it is best for the learning agent to keep a sizable learning rate for a large number of transitions.
- It seems like the sweet spot for the value of α_0 is 1.0. This makes sense if we keep in mind our random initialization of the Q-matrix, with random values in the range [0, 1]. The random initialization range will be tightly coupled with the scale of the rewards and the value of α_0.
- For our initial Q-learning parameters the success rate is 98% on average, and the penalty rate is approximately 6% on the average. After the grid search we see that by optimal choice of the Q-learning parameters this can be boosted up to nearly 100% success rate and a penalty rate as low as 2%.

Top 5:

α_0	Θ	ϵ_0	Θ_ϵ	success_rate	penalty_rate	objective
1.0	10000.0	0.02	1	1.00	0.020930	0.958140
1.0	10000.0	0.20	3	1.00	0.021676	0.956647
1.0	10000000.0	0.02	3	1.00	0.028810	0.942381
1.0	10000000.0	0.02	1	0.99	0.025037	0.939926
1.0	10000.0	0.02	10	0.99	0.027734	0.934532

Worse 5:

α_0	Θ	ϵ_0	Θ_ϵ	success_rate	penalty_rate	objective
0.1	1.0	0.60	3	0.090000	0.586850	-1.083700
2.0	1.0	0.02	10	0.070000	0.604925	-1.139851
2.0	1.0	0.60	10	0.111111	0.695170	-1.279229
2.0	1.0	0.60	1	0.070000	0.742257	-1.414514
2.0	1.0	0.60	3	0.100000	0.858675	-1.617350

TABLE IV
REALIZATION NUMBER 3.

Q: *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?*

A: From the optimization of the Q-learning parameters shown above I do believe that the smartcab is doing pretty well. In some cases we even see 100% success rate, which means that even on the first simulation trial the smartcab is already learning enough to make it to the destination within the allotted time.

An optimal policy for this problem involves both the success rate and the penalty rate. As was mentioned above, I gave twice the importance to the penalty rate than I did the success rate, when considering the objective function for optimization of the Q-learning parameters. In a world where it is very costly to commit a traffic violation, the weight given to the penalty rate could be set to an even higher value.

I do believe that the smartcab could do better in terms of penalty rate, at the expense of missing some trip deadlines. The best result shown so far is 2% penalty rate. I am a driver and I would not want to commit a traffic violation at that rate. I think a more reasonable penalty rate that would make the smartcab comparable to a human driver would be around

green?	car_left	car_oncoming	waypoint	None	forward	left	right
False	False	forward	forward	**0.07**	-0.86	-0.92	-0.27
			left	0.15	**0.35**	0.23	-0.22
		left	forward	**0.42**	0.36	0.02	0.18
		right		**0.96**	0.49	0.67	0.60
		None		**0.00**	-0.90	-1.00	-0.42
			left	**0.00**	-0.99	-0.91	-0.42
			right	0.05	-0.92	-0.93	**2.22**
	True		forward	**0.17**	0.15	-0.91	-0.28
			right	0.09	-0.86	0.36	**0.38**
True	False	forward	forward	**0.61**	0.43	0.36	-0.40
		left		0.22	**2.18**	-0.23	-0.24
			right	0.37	0.34	-0.38	**0.82**
		right	forward	0.15	**1.90**	0.64	0.08
		None		0.22	**2.18**	-0.24	0.31
			left	0.27	0.71	**2.00**	0.79
			right	0.08	-0.41	0.31	**2.22**
	True		forward	0.00	**0.01**	-0.27	-0.47
			right	0.01	0.16	0.75	**1.88**

TABLE V
FINAL Q MATRIX

one traffic violation for every 10000 intersections visited, or a 0.01% penalty rate.

To be more specific about what was learned by the agent we can go ahead and inspect the final values of the Q matrix for a 100-trial run of the smartcab simulation. The results are shown in Table. V. The table rows are color-coded such that if the correct action for the row state is favored it is green, if an incorrect action is favored it is colored red, and if there is not enough information in the state to make the right decision it is colored orange.

The first and most important thing to note is that our Q-matrix only has 17 rows in this case. The number of possible states in the state space that we are considering is 48. So there are a total of 31 states in our state space that we did not even see over the course of 100 trials!! This has to do with the fact that we do not see states that involve other traffic very often, because our simulation environment has only three dummy agents sharing the world with our smartcab.

Looking at the colors in Table V we see that there are some states where our smartcab has not learned enough, and still continues to make mistakes. The states where mistakes are made all involve other traffic, and the reason the agent has not been able to learn much about those is again due to the fact that there are only three other agents in the environment.

Overall I am happy with the performance of the smartcab, however based on the examination of the Q-matrix I think that perhaps it could be improved if one would have a learning rate that depended on the specific state. In that way if one stumbles later on in the simulation onto a state that is not encountered very often, one could give a sizable weight to the discounted reward observed in the transition that would follow from the unfrequently observed state.