UDACITY

PROJECT

Train a Smartcab to Drive

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

2 SPECIFICATIONS REQUIRE CHANGES

You have a good agent here and and impressed with your understanding of these reinforcement learning techniques. You have improved both required sections, but just have a couple things left to perfect in each, but none should be too difficult. We look forward to seeing your final submission. And keep up the hard work!

Implement a Basic Driving Agent

Student summarizes observations about the basic driving agent and its behavior. Optionally, if a visualization is included, analysis is conducted on the results provided.

Question 1

- Good job examining the reward policy in this section. One way to force the agent to move is through the use of simulated annealing.

Question 2

- Good job checking out the environment. As there are many tuning knobs we can check out here and play around with!

Question 3

- Impressive observations here, as this smartcab really isn't that smart yet. This lack of success does make sense as the agent is only exploring and not learning, collecting many negative rewards. This behavior is similiar to a random walk.

Inform the Driving Agent

Student justifies a set of features that best model each state of the driving agent in the environment. Unnecessary features not included in the state (if applicable) are similarly justified.

You have modeled the environment very well.

```
state = (waypoint, inputs['light'], inputs['oncoming'], inputs['left'])
```

It is always an important thing to determine a good state before diving into the code, as this will pave the way to an easier implementation. And very nice discussion for the need of all these variables and good idea omitting traffic from the right.

Would recommend discussing some reasons in why the deadline isn't needed. Consider this

- As if we were to include the deadline into our current state, our state space would blow up, we would suffer from the curse of dimensionality and it would take a long time for the q-matrix to converge.
- Also that including the deadline could possibly influence the agent in making illegal moves when the deadline is near.

The driving agent successfully updates its state based on the state definition and input provided.

The driving agent successfully updates its state

Implement a Q-Learning Driving Agent

The driving agent chooses the best available action from the set of Q-values for a given state. Additionally, the driving agent updates a mapping of Q-values for a given state correctly when considering the learning rate and the reward or penalty received.

Still have one issue here. in your `choose_action()` function and code of

```
action = max(self.Q[state],key=self.Q[state].get)
```

your agent should be choosing a random action from a choice of actions that have the highest Q-value. For example, since all actions are initialized with a reward of zero, it's possible that all four actions are considered "optimal". Not having the agent choose a random action from this would imply that it always chooses, perhaps, the first available option. This is incorrect behavior:

```
STATE X:
-- 'forward' : 0.00
-- 'left'    : 0.00
-- 'right'   : -1.023
-- 'None'    : 0.00
```

The agent should choose one of 'forward', 'left', or 'None' with equal probability, since they are all considered optimal with the current learned policy. Using a list would be a good idea.

Suggestion: In your `learn()` function. This is a very peculiar way to update the q-learning algorithm. Would recommend simply doing

```
self.Q[state][action] = (1.0 - self.alpha) * self.Q[state][action] + self.alpha * (reward)
```

Student summarizes observations about the initial/default Q-Learning driving agent and its behavior, and compares them to the observations made about the basic agent. If a visualization is included, analysis is conducted on the results provided.

Good observations here. As this is definitely expected with a decaying epsilon. Therefore we can reduce the chances of random exploration over time, as we can get the best of both exploration vs exploitation. Maybe with a slower decay rate and with more trails, this default Q-Learning driving agent could actually improve.

We can now using these metrics and rating as a good benchmark for the Improve the Q-Learning Driving Agent section.

Improve the Q-Learning Driving Agent

Student summarizes observations about the optimized Q-Learning driving agent and its behavior, and further compares them to the observations made about the initial/default Q-Learning driving agent. If a visualization is included, analysis is conducted on the results provided.

Very cool dynamic decay value is definitely a good idea, as it is crazy how the number of bad action/accidents/violation seem to be directly correlated to your epsilon value in these graphs. Could also look into a faster decaying epsilon value, so maybe it won't have to take so many trials to learn a feasible policy.

Would recommend compare your optimized Q-Learning driving agent to your initial/default Q-Learning driving agent a bit more here. How about the other graphs here, average reward, violations, etc...

The driving agent is able to safely and reliably guide the *Smartcab* to the destination before the deadline.

Congrats on your A+ ratings!!!

Student describes what an optimal policy for the driving agent would be for the given environment. The policy of the improved Q-Learning driving agent is compared to the stated optimal policy, and discussion is made as to whether the final driving agent commits to unusual or unexpected behavior based on the defined states.

Very impressive and good job examining the q-learning table and determining a state where the agent is acting optimally and one where the agent is not. I bet with even more training trials the agent could actually encounter even more of these state, resulting in even more efficient max action selection and higher confidence.

With your Incorrect/unexpected policy of

```
State 2: waypoint=left, light=green, oncoming=forward, left=forward
Q-table values: forward=1.04, right=0.00, None=-4.96, left=-9.56
```

You can see that there are multiple dummy agents involved(oncoming and left traffic); therefore maybe one idea here would be to increase the number of dummy agent on the road so the agent can hopefully learn more of these states in the beginning trials, then become more confident in the later ones.

Student correctly identifies the two characteristics about the project that invalidate the use of future rewards in the Q-Learning implementation.

Relook at your second idea here of "*Additionally, there is no reward for arriving to the destination on time. Therefore, the agent has no incentive to find an optimal route.*" As the agent actually receives a reward of 10 when it reaches the destination. Therefore the destination does have an issue. Does the location of it change throughout the trials runs? Why would this matter? Why would this affect future rewards?

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your project.

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Rate this review

Student FAQ