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

Hey,
You have done really well in updating your implementation and report. A couple of small changes are needed to meet all the specifications. I'm sure you'll ace your next submission!

[OPTIONAL] Getting Started

Student provides a thorough discussion of the driving agent as it interacts with the environment.

Student correctly addresses the questions posed about the *Train a Smartcab to Drive* code.

Implement a Basic Driving Agent

Driving agent produces a valid action when an action is required. Rewards and penalties are received in the simulation by the driving agent in accordance with the action taken.

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

Great summary here!

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.

" For the same reason left, right and oncoming are essential inputs (maybe not in the context of a single move but in general) since the reward is directly correlated with some of them (depending on the value of traffic light)."

You can elaborate here by describing what these variables are depicting and describing how the rewards are correlated in a little more detail.

The total number of possible states is correctly reported. The student discusses whether the driving agent could learn a feasible policy within a reasonable number of trials for the given state space.

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

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.

Great work updating the implementation. There is still one issue here:

- In the `choose_action()` function, the agent should take a random action when `self.learning` is `False`. With the current implementation, if `self.learning == False`, the program would go to the `else` statement and end up taking the best action possible. Please note the TODOs here:

```
# When not learning, choose a random action
      # When learning, choose a random action with 'epsilon' probability
      # Otherwise, choose an action with the highest Q-value for the current state
```

- A simple fix would be to change this line:

```
if (self.learning and choose_at_random):
  actions = self.valid_actions
```

  The `if` statement can be updated to check for when the agent is *not learning* **or** `choose_at_random`.

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 work adding the similarity between the basic agent and the agent that learns from the policy. You can expand the discussion here though. What similarities do you observe when you look at the visualizations for both?

Improve the Q-Learning Driving Agent

The driving agent performs Q-Learning with alternative parameters or schemes beyond the initial/default implementation.

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.

Good job comparing the improved agent with the previous implementation. You should also add a line noting the improved in the ratings (A+ and A from F and F).

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

Good work getting good ratings for both safety and reliability.

NOTE: If you look at the log files, the agent reaches the destination only 5 times out of the 10 testing trials before the deadline. It might improve with more training trials though.

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.

Good work adding the examples for the optimal policies here. However, you haven't included any examples of sub-optimal policies where the agent did not choose the best action available and receives a negative reward.

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

RESUBMIT

DOWNLOAD PROJECT

Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

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