UDACITY

PROJECT

Train a Smartcab to Drive

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW 2

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

2 SPECIFICATIONS REQUIRE CHANGES

Hey,

This was an exceptional first submission. The report is really well done and the analysis presented here are very impressive. There is one major flaw in the implementation though, as the Q-learning equation has not been accurately implemented. Once that change is fixed (and a few sections in the report subsequently), you should be golden! We look forward to reviewing your updated work ☺

[OPTIONAL] Getting Started

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

> If the lights is green and the Smartcab isn't moving, then it receives a negative reward.

This is not always true. What happens when there is oncoming traffic? Are the rewards still negative for idling at a green light?

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.

A visualization is included that correctly captures the results of the 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.

Good job with the summary.

This statement is not entirely accurate:

> It makes sense! The agent is randomly taking actions and it could possibly reach the destination.
> Therefore, the reliability is low. We can see that through the bottom plots.

The 10 trial rolling reliability plot actually shows the reliability increase with time. It is close to 50% at the end of the 20 trials which shouldn't make a lot of sense. This is probably coincidental and re-running the visual should give different results (more in line with what you have described)

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.

Great job justifying the features that you have selected and those you have excluded.

Another reason to remove deadline could be the increased state size from its inclusion, since it is a continuous variable

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.

Nice job with the calculation of the number of possible states. It is actually possible to learn a good policy in a reasonable number of trials with a state size this small. Each trial allows for the exploration of many states between source and destination.

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

The state is updated based on the definition 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.

A couple of changes need to be made in the code. Please find the details in the code review section

A visualization is included that correctly captures the results of the initial/default Q-Learning driving agent.

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 summarizing the default agent. The agent seems to be learning, but the number of trials might be too low.

Note: When you update the code and re run the visuals, some of the analysis presented here might need to change.

Improve the Q-Learning Driving Agent

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

A visualization is included that correctly captures the results of the improved Q-Learning driving agent.

Again, the visuals might change when you re run the updated code

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.

It's great that you have tried different values for all hyperparameters and different functions as well. You should mention explicitly which values you used in the end. It isn't clear from the answer whether alpha was chosen to be 0.01 or 0.005 and which of the two decay functions presented were chosen.

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

Congratulations on the high 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.

Great job giving examples for both optimal and sub optimal policies here. You should also describe what an optimal policy is, as the question demands:

> Provide a few examples (using the states you've defined) of what an optimal policy for this problem would look like.

Note: The log files might change on re running the code so the examples here would need to be updated as well. I wouldn't want the next reviewer to fail the rubric just because the older examples were used in the report.

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

Excellent job identifying the two characteristics. To elaborate on them:

- SMARTCAB
  The smartcab is unaware of the destination. In fact, it is also unaware of what the next state is going to be. This is because waypoint is not always followed and it is not possible to predict beforehand what the next state will be. Without the next state, there isn't any way to include future rewards
- ENVIRONMENT
  The starting and destination points are random as well. This makes learning the route by rote impossible and any future rewards learned in the current trial meaningless for the next trial.

RESUBMIT

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS

Best practices for your project resubmission

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

Watch Video (3:01)

RETURN TO PATH

Student FAQ