UDACITY

PROJECT

Train a Smartcab to Drive

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW 3

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

7 SPECIFICATIONS REQUIRE CHANGES

Definitely getting there in your analysis. Just need to use more training trails with increased exploration and your agent should improve. Keep up the hard work.

[OPTIONAL] Getting Started

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

"*When there's green light without incoming traffic it received a penalty, if there's red light it received a reward. Both scenarios, it did nothing. If there's green light with no oncoming traffic, then it will receive a penalty for not moving*"

One more instance that needs to be addressed here. What are the rewards when there is a green light **with** other traffic?

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

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

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.

Driving agent produces a valid action when an action is required

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.

"*rack up plenty of negative rewards and bad actions even as number of trials increase. It is unreliable its safety rating and reliability rating are both F*"

As we do see some random noise here, but this really isn't a trend. And would definitely not want to ride in this not-so-smartcab!

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.

Correct that using the deadline in its full form is unreasonable. Maybe the only way we could actually use the deadline would be to encode it as a binary variable, say +10 time steps and -10 time steps. But this would still double the number of states. Something to think about when working with continuous values.

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.

"*Optionally I learned to use Monte Carlo to estimate the number of trials needed to visit all possible 96 states. In 2000 trials, it's possible to visit 96 states with a 96%+ chance in under 800 steps.*"

Glad that you like the Monte Carlo estimation! Another idea to think about is how the amount and actions of the other dummy agents would affect the agents exploration. Would these states really actually be seen in a randomly uniformly distributed manner? How would the number of other agents on the road affect this? Think about if we only had a few other agents? How often would we see instances with left and oncoming traffic?

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

Your agent updates its state while driving, nice work!

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.

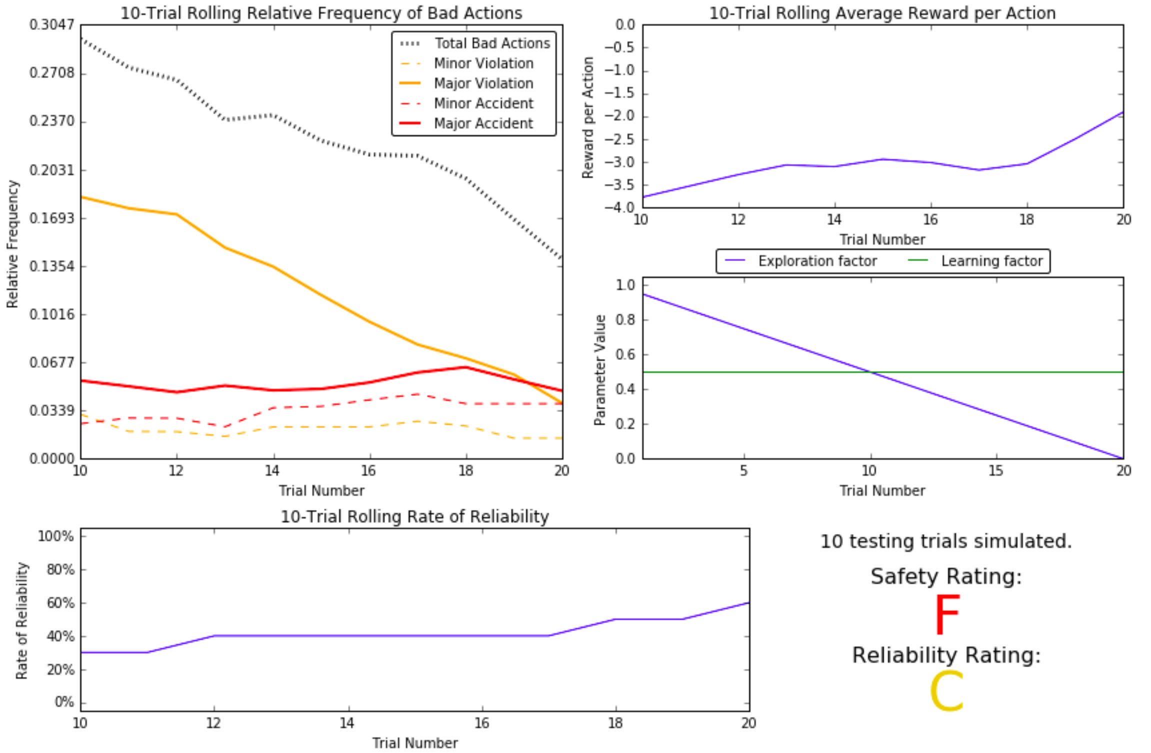
Very nice code adjustments here. Check out the Code Review for some more optimization ideas!

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

Seems as you still have the same issue here in your visual. It seems that you have actually run your agent with `action=None`, since your visual is corresponding to an agent that is not moving(as we see no accidents, since the agent can't get into accidents if it does move). Make sure you run and populate your `log` files with the parameters of

- 'enforce_deadline' - Set this to True to force the driving agent to capture whether it reaches the destination in time.
- 'update_delay' - Set this to a small value (such as 0.01) to reduce the time between steps in each trial.
- 'log_metrics' - Set this to True to log the simluation results as a .csv file and the Q-table as a .txt file in /logs/.
- 'n_test' - Set this to '10' to perform 10 testing trials.
- 'learning' - Set this to 'True' to tell the driving agent to use your Q-Learning implementation.

Should look more like this



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.

You will need to update this analysis once you fix your visual, but you have all the right ideas.

Improve the Q-Learning Driving Agent

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

"*self.epsilon = self.alpha ** self.n_trial previous grader's comment was epsilon is not decaying, but because 0< a < 1 epsilon is decaying. We can see that in the simulation. I am confused. Double asterix means exponetial to the power of. epsilon = alpha ^ number of trials*"

You are correct that your epsilon is decaying here, but your epsilon value is decaying WAY TOO FAST and you are starting it way too low. As we need to allow the agent to explore and learn the best actions at each state and keep epsilon high for a much longer amount of time. Therefore would recommend starting `epsilon=1` and SLOWLY decaying this value to zero. A interesting epsilon decaying rate to check out would be the Gompertz function. And use a negative of it.

You will need to update this section and your parameter values when you agent is performing better.

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

" *Not sure why the safety rating is still so bad though. Earlier a decay function of epislon -= 0.01 works better. I am not exactly satisfied with final rating yet. I think a slower decaying funciton is needed.*"

Yes a slower decay rate is needed. Could also actually use the linear decay rate of `epsilon -= 0.01` or even `epsilon -= 0.001` (while starting epsilon at 1), thus try these out.

You will need to update this section and your comparison to the initial/default Q-Learning driving agent when you agent is performing better.

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

To pass this section you will need to receive at least an A rating for both safety and reliability. Therefore some ideas

- The two big things here is that your agent needs many more training trials, as only 20 is definitely not enough to learn a feasible policy. As I have actually seen up to 7000 training trials used! And the student was able to get A+ ratings with 200 testing trials(this was a bit obsessive, but hopefully you get the point). Therefore to achieve this you will need to keep epsilon high for a much longer amount of time(start this at 1). As this allows for much more exploring and learning! Since the main thing in the training phase is to explore, learn and fill up the Q-Values!

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.

```
('left', 'green', 'right', 'forward') -- forward : 0.57 -- None : 0.00 -- right : 0.00 -- left : 0.00
```

"*This last one is SUBOPTIMAL. Due to the limited number of trials, there has not been an opportunity for the agent to explore Left. None would have been better because the reality is I want to go left, but there's oncoming traffic, so I should wait a bit before heading left. The agent has not had the opportunity to explore this, so None is 0, and Left is also 0. It suboptimally thought going forward is the best strategy*"

Great intuition here! As more training trials with more exploration is exactly what the agent needed here, as you will actually do this to improve your agent.

The reason this is marked as *Requires Changes* is that you will also need to update your Q-Values for this section when your agent is performing better.

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

"*The agent does not have full view of the grid, but only the information at each intersection. And with each trial, the the destination may change.* "

This sums it up great! This is due to egocentric nature of the agent, as well as the random aspects of state that cannot be predicted ahead of time. Were the simulation changed to an allocentric view, where the agent could sense where it was in relation to the destination, etc, then gamma term would be a more important parameter for optimal performance and policy generation.

RESUBMIT

DOWNLOAD PROJECT

3 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

Rate this review

Student FAQ