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

3 SPECIFICATIONS REQUIRE CHANGES

This is an excellent first submission--great job! I can tell you've placed a lot of effort and time into this project: you've taken a methodical and rational approach to optimizing your agent. There are a few changes needed, but they should be easy to fix. Keep up the good work!

Getting Started

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

You do a nice job of summarizing the framework in which rewards are awarded, and accurately explained the different parameters and functions within agent.py, environment.py, simulator.py, and planner.py. Great detail and very nicely explained!

Student correctly addresses the questions posed about the code as addressed in Question 2 of the notebook.

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.

Excellent--your driving agent produces a valid action when an action is required.

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

Your random driving agent performed as expected -- the reported frequencies of bad actions (~40%) and low (~20%) rate of reliability is typical of a random action performance. Well done! Your analysis of the behaviors of your basic driving agent is also thoughtful and detailed.

> As the number of trials increase, the outcome of results do not appear to change significantly for this no learning scenario.
> This Smartcab would not be considered safe (accident with 30% frequency and overall about 40% of bad actions), and would not be considered reliable for its passengers because it's reliability for reaching the destination on time is 10%-30%, which is quite low.*

Nice to see that you ran this with n = 100 and have examined environment.py to review the rewards allocation. This is absolutely the case--since the agent is not set to learn, there is no trend and consistencies we see in the data are coincidental. At this point, increasing the number of training trials will not improve this metric.

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. Students argument in notebook (Q4) must match state in agent.py code.

Nice analysis and justification of the set of features you've chosen to include in your default-QLearner! You are correct that `deadline` is not as important of a feature to include since including this could influence the agent to make illegal moves to meet the deadline. Also, including `deadline` will drastically increase the dimension of the state space.

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.

Your state size calculation is correct!

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

The driving agent changes state while running the program -- nicely implemented.

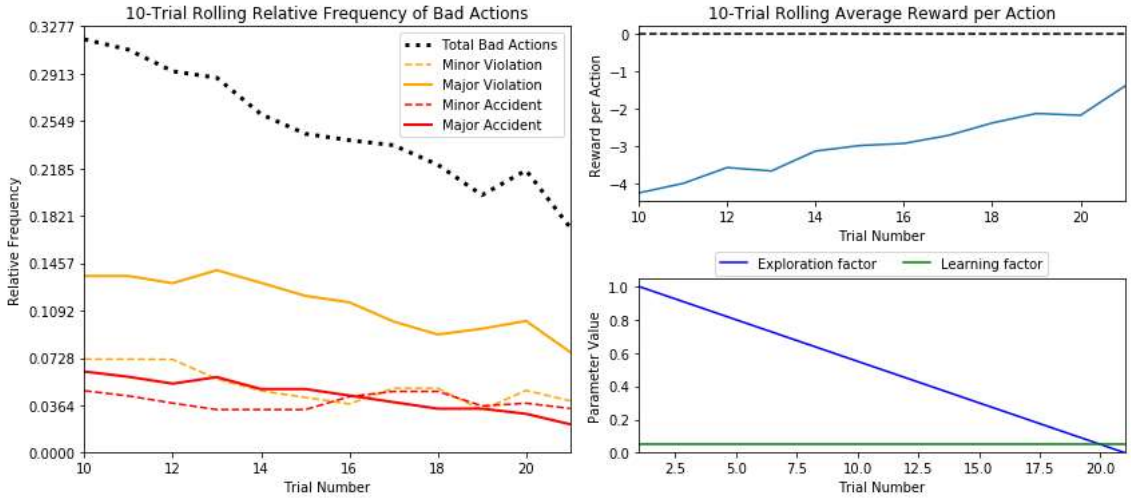
Implement a Q-Learning Driving Agent

The driving agent: (1) Chooses best available action from the set of Q-values for a given state. (2) Implements a 'tie-breaker' between best actions correctly (3)Updates a mapping of Q-values for a given state correctly while considering the learning rate and the reward or penalty received. (4) Implements exploration with epsilon probability (5) implements are required 'learning' flags correctly

Your code is able to iterate over the set of Q-values in a state and select the maximum value in order to implement the best action. Good work!

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.

Your default Q-Learner looks like it's learning a policy! We begin to see trends forming here of decreasing frequency of bad actions and increasing rewards, which indicate that the agent is learning. The graphs also show a correctly implemented linear epsilon decay function and the default number of 10 testing trials. However, it looks like your alpha is set to a different number from the default rate of 0.5:



Please run the simulation again with the default alpha.

Nice touch with running different sensitivity studies to examine the effect of dropping the `left` and `right` inputs. As you have also seen, it is clearly possible to optimize the agent with a linear decay function.

> As the number of training trials increased, did the number of bad actions decrease? Did the average reward increase?
> Answer: Yes to both the questions as seen in the display plots.
> How does the safety and reliability rating compare to the initial driving agent?
> Answer: Both the initial (Basic) agent and the default Q-learning agent yielded : Safety = F, Reliability = F

Your observation is correct. We see some improvement shown when compared with the results of the random agent. However, at this point, the agent has not yet explored enough states to build a stable Q-table. I would recommend generating summary statistics for each run as you begin optimizing your agent's performance -- a running count of the trial count, success rate, and net reward/penalties could be useful in zoning in on where your q-learner is getting stuck in local optima or how your errors are distributed.

REQUIRED

- Run default simulation with the default rate of 0.5 for alpha and update your analysis.

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.

Nice job in getting this cab up to A+/A ratings! Really nice to see that you've completed extensive training trials for your agent--this will allow your agent to explore as many states as possible.

> Approximately how many training trials were needed for your agent before beginning testing?
> Answer: 20 training trials

I would just make sure that this answer reflects your final agent as it looks like it had completed 5000+ trials.

> What epsilon-tolerance and alpha (learning rate) did you use? Why did you use them?
> Answer: epsilon tolerence = 0.005, alpha(learning rate) = 0.005 was used (resulted in Safety rating = A+ and ReliabilityRating A+); sensitivity studies were performed for epsilon with exponential decay model using higher alpha = 0.01 (yielded unacceptable safety and reliability ratings of F), and lower alpha = 0.001 (yielded A+ rating for both safety and reliabilty).

Excellent choice! I appreciate that you've explained your thought process here in detail. A higher alpha can lead to slower convergence. However, an alternative to choosing one constant rate is to implement a decay function, which will allow you to begin with a high learning rate and decrease it over the course of training trials. Beginning with a higher learning rate will attribute more weight to new results, and steadily lowering it to a relatively small number will allow your cab to gradually switch to exploitation of the Q-table and policies.

REQUIRED

- Update your answer to the question "**Approximately how many training trials were needed for your agent before beginning testing?**"

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

Congratulations on achieving A+/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. Student presents entries from the learned Q-table that demonstrate the optimal policy and sub- optimal policies. If either are missing, discussion is made as to why.

Nice job on including a very clear, specific examples from your log to shape your optimal policy. You've also selected a few different scenarios, which helps in developing a comprehensive policy. Great job!

There is just one change needed to this answer; please provide a discussion to the question:

Are there any states where the policy is different than what would be expected from an optimal policy?

Examples of sub-optimal policies include those that do not cause violations but reduce the overall efficiency by delaying the agent in reaching the destination. For example, we may see the agent turning right or moving forward at a green light instead of following the waypoint by turning left. Why might the agent not have learned the correct action?

REQUIRED

- Please provide an answer to the question above.

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

Excellent explanations. You are correct that future rewards are negligible since the agent cannot see past the current intersection and that the environment is stochastic. Hence, predictions about future events cannot be made.

RESUBMIT

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ›

Learn the best practices for revising and resubmitting your project.

RETURN TO PATH

Student FAQ