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

4 SPECIFICATIONS REQUIRE CHANGES

You have a very good understanding of Q learning.

Please attach your final version of agent.py so reviewer can run your code and verify the performance.

Need to point out, following code should not be in agent.py. It has the logic that the agent should **LEARN** from the process.

```
        # Update as a DummyAgent
        action_okay = True
        if self.next_waypoint == 'right':
            if inputs['light'] == 'red' and inputs['left'] == 'forward':
                action_okay = False

        elif self.next_waypoint == 'straight':
            if inputs['light'] == 'red':
                action_okay = False

        elif self.next_waypoint == 'left':
            if inputs['light'] == 'red' or inputs['oncoming'] == 'forward' or inputs['oncoming'] == 'right':
                action_okay = False

        if action_okay == False:
            action = None
```

实施基础驾驶智能体

✓

学生能够实施智能体接受指定输入所需的接口。

✓

驾驶智能体产生有效输出（one of None、'forward'、'left'、'right'），以响应输入。

✓

驾驶智能体在模拟器中正常运行，未产生任何错误。奖励和惩罚没有必要——我们允许智能体出错。

确认并更新状态

学生确认了模拟驾驶智能体和环境的状态，并给出了合理的理由。

From your report, it is not clear what are included in the agent state.

You mentioned about "Time Left", "Traffic state", "Next_waypoint", and "Locations". Are they the states you are considering for agent states? In your code, you are using system "inputs" for the state.

Could you please clearly explain what are the "factors" you are using for the agent state?

Remember, agent uses the states to make decision for actions and learn from the result (action and reward).

驾驶智能体运行时会根据当前输入更新自己的状态。确切的状态并不重要，也无需与输入有关系，但应该在运行时进行相应更改。

实现 Q 学习

驾驶智能体会正确更新 Q 值的表/映射，从而实施 Q 学习算法。

From the code in your report, I can see you have a good understanding of Q-Learning.
You have successfully implemented the Q-Learning formula and also the "exploration and exploitation".
Great job!!

鉴于某个状态当前的 Q 值集，智能体会选择可用的最佳动作。

From the code in the report, I can see the agent can pick the best action.

学生报告了观察到的行为变化，并提供了合理解释。

Excellent analysis!

增强驾驶智能体

驾驶智能体不断按规定时间到达目的地，并且净奖励保持为正数。

From the code in the report, I can see the agent can pick the correct action and calculate/update Q table.
It will be great if your final version of agent.py is attached so I can run it and verify your agent performance.

已报告了学生在 Q 学习基础实现期间所做的改进。

To help the agent learn, learn smarter, learn faster, we can tune different parameters in the Q-learning functions.

- **Alpha**: the learning rate. It controls how much the agent should learn from new run and how much it shall use its accumulated experience. Usually, learning rate shall drop (learning rate decay) along agent learning process. In this project, we can keep it as constant, but it should not be too high.
- **Gamma**: the discount rate. It controls how much the agent weigh the future reward into the Q value calculation. It can be tuned to help the agent be more "determined" to heading to the destination instead of circling around collecting the small safe rewards.
- **Epsilon**: the random action coefficient. It controls the agent to take some random action instead of following the Q table to find the action.

Do you mean you are getting the best result with epsilon < 0.1, alpha > 0.9, and gamma in [0.1, 0.4]? It seems alpha value is too high, since it means new Q value will use only 10% from Q table (old experience) and 90% from new learning. When the agent is experienced, it should rely more on its experience.

It is recommended to run different combinations of the parameters and present the results in a table format. This way, it can demonstrate the effects of all the parameters, and compare the results of agent performance.

从最终驾驶智能体与学习最佳策略的关系角度讨论了该智能体的性能。

The optimal policy is more related the ultimate goal of this project, what we want to achieve by training the agent.
Please notice that, the agent is trained in a simplified simulation environment. All the conditions, e.g. next_waypoint, traffic lights, step rewards, and deadline, are all designed to help the agent to learn. Any changes of those conditions shall affect the agent performance.

With that in mind, could you please use the performance of your agent and compare it to the optimal policy?
Please discuss in your report,

- is your agent learning the optimal policy well?
- anything that limits the agent to fully learn the optimal policy?
- if the agent can achieve 100% success rate, is it good enough to run in the real world?

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

Student FAQ