Implement a Basic Driving Agent

To begin, your only task is to get the **smartcab** to move around in the environment. At this point, you will not be concerned with any sort of optimal driving policy. Note that the driving agent is given the following information at each intersection:

- The next waypoint location relative to its current location and heading.
- The state of the traffic light at the intersection and the presence of oncoming vehicles from other directions.
- The current time left from the allotted deadline.

To complete this task, simply have your driving agent choose a random action from the set of possible actions (`None`, `'forward'`, `'left'`, `'right'`) at each intersection, disregarding the input information above. Set the simulation deadline enforcement, `enforce_deadline` to `False` and observe how it performs.

QUESTION: *Observe what you see with the agent's behavior as it takes random actions. Does the* ***smartcab*** *eventually make it to the destination? Are there any other interesting observations to note?*

Ans: Yes, the car makes it to the destination. But, I think the smart cab is performing terribly, crossing red signs, although taking random actions but not following traffic rules all the time. I have also seen the the car not taking any action although there is green light and there is no oncoming traffic. I have updated the delay to 2 secs so that I can visualize the pygame grid.

I set the enforce_deadline = True and found the success rate is 15%. So, while taking actions randomly the agent is the reaching the destination 15 times out of 100 within the deadline. In a nutshell, the agent is rarely reaching the destination (within the deadline) while taking random actions.

Inform the Driving Agent

Now that your driving agent is capable of moving around in the environment, your next task is to identify a set of states that are appropriate for modeling the **smartcab** and environment. The main source of state variables are the current inputs at the intersection, but not all may require representation. You may choose to explicitly define states, or use some combination of inputs as an implicit state. At each time step, process the inputs and update the agent's current state using the `self.state` variable. Continue with the simulation deadline enforcement `enforce_deadline` being set to `False`, and observe how your driving agent now reports the change in state as the simulation progresses.

QUESTION: *What states have you identified that are appropriate for modeling the* ***smartcab*** *and environment? Why do you believe each of these states to be appropriate for this problem?*

OPTIONAL: *How many states in total exist for the* ***smartcab*** *in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?*

Ans: Following are states I think are important for modeling the smart cab:

Traffic light and other agents (oncoming, left): For our agent to follow the US traffic rules we must know the state of the traffic light (green or red) and if there is any on coming agents or agents coming from left. (Traffic from right direction is not necessary but could be included)

Next_waypoint: The agent need to know the next way point to finally reach the destination through the optimal route based on the current location and in optimal time rather than randomly choosing the next action which would not be optimal in terms computation and it might long time for the Q learning algorithm to converge. Based on the traffic rules - traffic light and oncoming traffic the agent will decide whether it can proceed in the way point direction or not.

Answer to optional question:

Besides traffic light, vehicles from each of the oncoming direction and the time left to reach the destination(deadline). In total there are five states – Traffic light, traffic – oncoming, left, right (it's not required but I can see it's a key in the "input" variable) and deadline (time left to reach the destination). Basically a combination of each value of these five variables would form a unique state. Avoiding deadline I would think in total 2(traffic lights)*4(oncoimg traffic)*4(traffic from left)*4(traffic from right although not necessary)* 4 (next way point) = 512 combinations. I think 512 states are more than enough for Q learning to take the right and optimal decision for the next action keeping in mind the the next way point to reach the destination, the current traffic states and the time left to reach the destination.

I agree with the fact that including deadline would make the complexity of the state space to significantly increase into a size that cannot be feasibly explored within in 100 trials. (we don't have defined inputs for deadlines, its randomly

chosen - deadline = self.env.get_deadline(self)). Including the deadline as a state with randomly chosen values in the Q learn dictionary would definitely lead to memory issues (Specifically space complexity for the Q learn algorithm would be super terrible). Theoretically, it would probably make sense for the learning algorithm to adjust the time to reach the destination based on the deadline, the algorithm might want to fine tune the actions to reach the destination on time based on deadline. For e.g. Higher the deadline, the Q learn algorithm might want to take the right actions to maximize rewards to reach the destination rather than reaching the destination in haste by compromising on rewards (ok to get discounted little bit that rewarded to reach on time if the deadline is less). But again we are not rewarded to reach on time, we are only rewarded if successfully complete our trip and we take right actions as per US traffic rules.

Implement a Q-Learning Driving Agent

With your driving agent being capable of interpreting the input information and having a mapping of environmental states, your next task is to implement the Q-Learning algorithm for your driving agent to choose the *best* action at each time step, based on the Q-values for the current state and action. Each action taken by the **smartcab** will produce a reward which depends on the state of the environment. The Q-Learning driving agent will need to consider these rewards when updating the Q-values. Once implemented, set the simulation deadline enforcement `enforce_deadline` to `True`. Run the simulation and observe how the **smartcab**moves about the environment in each trial.
The formulas for updating Q-values can be found in **this** video.

QUESTION: *What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?*

Ans: I have noticed that agent is behaving appropriately compared to taking random actions. The success rate is 95%. It's reaching the destination on time and I could notice (when I increased the delay (to visualize)) – it's not crossing any red signs or not taking action against the US traffic rule. Please refer the screen shot below with the success rate. This behavior is occurring because the agent is learning to take the next action (better action to maximize rewards) based on the previous actions based on rewards and the discounts associated with the previous actions.

```
Environment.act(): Primary agent has reached destination!
Simulator.run(): Trial 96
Environment.reset(): Trial set up with start = (8, 6), destination = (2, 4), deadline = 40
RoutePlanner.route_to(): destination = (2, 4)
Environment.act(): Primary agent has reached destination!
Simulator.run(): Trial 97
Environment.reset(): Trial set up with start = (1, 4), destination = (3, 2), deadline = 20
RoutePlanner.route_to(): destination = (3, 2)
Environment.act(): Primary agent has reached destination!
Simulator.run(): Trial 98
Environment.reset(): Trial set up with start = (3, 6), destination = (5, 4), deadline = 20
RoutePlanner.route_to(): destination = (5, 4)
Environment.act(): Primary agent has reached destination!
Simulator.run(): Trial 99
Environment.reset(): Trial set up with start = (8, 2), destination = (2, 6), deadline = 50
RoutePlanner.route_to(): destination = (2, 6)
Environment.act(): Primary agent has reached destination!
Succes percentage is: 95.0 %
```

I have kept (to begin with):
The learning rate = 0.1
The discounted rate = 0.1

Steps:

- ***I initialized the Q learn utility dictionary with value 1 (I chose arbritary positive integer), as mentioned earlier the learning is based on the state of traffic light, (oncoming, left , right) traffic, next_way point(optimal path forward) as inputs and output is the action taken (random action to initialize with). So, I would imagine the keys of the Q learn dictionary would be the tuple of states and result or the value is the random action taken.***

- ***As per my understanding from the Q learning lesson, I should maximize the current state of Q learn utility and then take action that would maximize the learning from the current state of the Q learn utility. I would maximize the reward as well.***

- ***Now I would define the Q learn utility for the next set of states and actions based on the Q learning equation by rewarding for every right action taken based on traffic rules and discounting (gamma parameter) for every wrong move.***

The learning rate is crucial because if the learning rate is '0' the new utility is same as the previous utility have not learned anything new and if its '1' the new utility totally ignores or forgets the previous utility and learnt everything new which is also bad. So, the learning rate should be between 0 and 1. I started with 0.1 to have room to improve later.

I want to start with a low discount rate to begin with. I don't want to discount heavily on wrong actions (not following traffic rules) to begin with

As per the environment.py, at the end of each trial if the reward > = 10 the agent has reached the destination before the deadline Else it has failed

Improve the Q-Learning Driving Agent

Your final task for this project is to enhance your driving agent so that, after sufficient training, the smartcab is able to reach the destination within the allotted time safely and efficiently. Parameters in the Q-Learning algorithm, such as the learning rate (alpha), the discount factor (gamma) and the exploration rate (epsilon) all contribute to the driving agent's ability to learn the best action for each state. To improve on the success of your smartcab:
Set the number of trials, n_trials, in the simulation to 100.
Run the simulation with the deadline enforcement enforce_deadline set to True (you will need to reduce the update delay update_delay and set the display to False).
Observe the driving agent's learning and smartcab's success rate, particularly during the later trials.
Adjust one or several of the above parameters and iterate this process.
This task is complete once you have arrived at what you determine is the best combination of parameters required for your driving agent to learn successfully.

QUESTION: Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?

Answer:

- ***Learning rate = 0.1, discount rate = 0.1, Success = 95%***
- ***Learning rate = 0.4 discount rate = 0.7, Success = 91%***
- ***Learning rate = 0.9 discount rate = 0.7, Success = 92%***

- *Learning rate = 0.5 discount rate = 0.5, Success = 95%*
- *Learning rate = 0.3 discount rate = 0.3, Success = 95%*
- *Learning rate = 0.2 discount rate = 0.2, Success = 95%*

Based above values I would say the best parameters are the ones for which Learning rate and discount rate are between 0.1 and 0.5

*** I think the model has reached an optimal state as it's reaching the destinations in 95% of the trials within the deadline by maximizing the rewards meaning mostly obeying traffic rules with minimum penalties. But I have also observed - say if the deadline set for the trial is high say 40 secs (I am not sure about the unit) compared to say 20 secs, the agent tends to take more turns. It could be that the agent has learnt to move around little bit or tend to take the longer route if the deadline set to reach the destination is more than less. This could be a problem in a real world scenario, say for e.g. I hired a self - driving taxi to reach the airport before 3 hours of my departure, now the taxi knows I have 3 hours for my flight departure it could possibly take the longer / non-optimal route so that it can kill more time in the roads. This is not what I or any passenger would want.*

Please below the output from the code for the last 10 trials. For e.g : I could see in one trial (highlighted in red below) the agent took a left turn on a red light and is negatively rewarded (reward = -1). I would infer the agent has not adequately visited this state in order to take the optimal action or is not heavy discounted for this action – i.e. it's not supposed to take left when there is red light, actually it's pretty severe mistake, I am guessing by increasing the discount rate little we might be able to train the agent to take the optimal action. Also the Q_learn value is '2', the agent might be more exploring than exploiting., now it knows that is it is negatively rewarded, it could possibly not take left when it sees a red light in the future. This issue could be possibly solved if I could implement epsilon based exploration-exploitation policy with a decay rate such the agent explore initially and then strictly exploit Q table in the later half. (As suggested in the feedback)

```
Simulator.run(): Trial 90
```

```
Environment.reset(): Trial set up with start = (2, 3), destination =
(1, 6), deadline = 20
RoutePlanner.route_to(): destination = (1, 6)
LearningAgent.update(): deadline = 20, inputs_light = green,
inputs_oncoming = left, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 19, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = left, reward =
2.0, Q_learn = 2
LearningAgent.update(): deadline = 18, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 17, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 16, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 15, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1

Simulator.run(): Trial 91
Environment.reset(): Trial set up with start = (7, 1), destination =
(5, 6), deadline = 35
RoutePlanner.route_to(): destination = (5, 6)
LearningAgent.update(): deadline = 35, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 34, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 33, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 32, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 31, inputs_light = red, inputs_oncoming =
None, inputs_left = None, action = left, reward = -1.0, Q_learn = 2
LearningAgent.update(): deadline = 30, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 29, inputs_light = green,
```

```
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 28, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = left, reward =
2.0, Q_learn = 2
LearningAgent.update(): deadline = 27, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 26, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 25, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 24, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 23, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 22, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 21, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1

Simulator.run(): Trial 92
Environment.reset(): Trial set up with start = (3, 1), destination =
(1, 5), deadline = 30
RoutePlanner.route_to(): destination = (1, 5)
LearningAgent.update(): deadline = 30, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = left, reward =
2.0, Q_learn = 2
LearningAgent.update(): deadline = 29, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 28, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 27, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 26, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 25, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = left, reward =
2.0, Q_learn = 2
```

```
LearningAgent.update(): deadline = 24, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 23, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 22, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 21, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 20, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 19, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 18, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 17, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1

Simulator.run(): Trial 93
Environment.reset(): Trial set up with start = (8, 6), destination =
(4, 1), deadline = 45
RoutePlanner.route_to(): destination = (4, 1)
LearningAgent.update(): deadline = 45, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 44, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 43, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 42, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 41, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 40, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 39, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
```

```
0.0, Q_learn = 0
LearningAgent.update(): deadline = 38, inputs_light = red,
inputs_oncoming = right, inputs_left = None, action = forward, reward
= -1.0, Q_learn = 1
LearningAgent.update(): deadline = 37, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 36, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 35, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 34, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 33, inputs_light = red,
inputs_oncoming = None, inputs_left = right, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 32, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 31, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 30, inputs_light = red,
inputs_oncoming = right, inputs_left = None, action = left, reward = -
1.0, Q_learn = 2
LearningAgent.update(): deadline = 29, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 28, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 27, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 26, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 25, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1

Simulator.run(): Trial 94
Environment.reset(): Trial set up with start = (1, 2), destination =
(4, 5), deadline = 30
RoutePlanner.route_to(): destination = (4, 5)
LearningAgent.update(): deadline = 30, inputs_light = red,
```

```
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 29, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 28, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 27, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 26, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 25, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 24, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 23, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 22, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 21, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 20, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 19, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 18, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 17, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1

Simulator.run(): Trial 95
Environment.reset(): Trial set up with start = (7, 5), destination =
(1, 1), deadline = 50
RoutePlanner.route_to(): destination = (1, 1)
LearningAgent.update(): deadline = 50, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
```

```
LearningAgent.update(): deadline = 49, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 48, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 47, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 46, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 45, inputs_light = green,
inputs_oncoming = None, inputs_left = right, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 44, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = left, reward =
2.0, Q_learn = 2
LearningAgent.update(): deadline = 43, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 42, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 41, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 40, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 39, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 38, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 37, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 36, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 35, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 34, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 33, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
```

```
0.0, Q_learn = 0
LearningAgent.update(): deadline = 32, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 31, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 30, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 29, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 28, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 27, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 26, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 25, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 24, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 23, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 22, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 21, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 20, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 19, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 18, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 17, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 16, inputs_light = red,
```

```
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 15, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 14, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1
```

Simulator.run(): Trial 96

```
Environment.reset(): Trial set up with start = (4, 3), destination =
(7, 5), deadline = 25
RoutePlanner.route_to(): destination = (7, 5)
LearningAgent.update(): deadline = 25, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 24, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 23, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 22, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 21, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 20, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 19, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 18, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 17, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 16, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1
```

Simulator.run(): Trial 97

```
Environment.reset(): Trial set up with start = (4, 2), destination =
(5, 5), deadline = 20
RoutePlanner.route_to(): destination = (5, 5)
```

```
LearningAgent.update(): deadline = 20, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 19, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 18, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 17, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1

Simulator.run(): Trial 98
Environment.reset(): Trial set up with start = (8, 4), destination =
(7, 1), deadline = 20
RoutePlanner.route_to(): destination = (7, 1)
LearningAgent.update(): deadline = 20, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 19, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = right, reward =
2.0, Q_learn = 3
LearningAgent.update(): deadline = 18, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 17, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 16, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 15, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
12.0, Q_learn = 1


Simulator.run(): Trial 99
Environment.reset(): Trial set up with start = (1, 5), destination =
(6, 4), deadline = 30
RoutePlanner.route_to(): destination = (6, 4)
LearningAgent.update(): deadline = 30, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 29, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 28, inputs_light = red,
```

```
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 27, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 26, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 25, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = left, reward =
2.0, Q_learn = 2
LearningAgent.update(): deadline = 24, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 23, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 22, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 21, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 20, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 19, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
LearningAgent.update(): deadline = 18, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 17, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 16, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = forward, reward =
2.0, Q_learn = 1
LearningAgent.update(): deadline = 15, inputs_light = red,
inputs_oncoming = None, inputs_left = None, action = None, reward =
0.0, Q_learn = 0
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 14, inputs_light = green,
inputs_oncoming = None, inputs_left = None, action = left, reward =
12.0, Q_learn = 2
```