SmartCab project report

Implement a Basic Driving Agent

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?

ANSWER: It seems, the agent could not reach to the given destination within a deadline by taking random actions. But rarely, it could. As observed, it could reach the destination usually at the beginning of simulation.

The reward was between -0.5 and 2.0, it's like

- 0.0 reward for the 'None' action,
- 2.0 reward for valid move
- -0.5 punishment for valid but other direction

The smartcab was jumping from one edge to another, it was very annoying and weird.

The traffic light stays about 3 seconds (updates) at one state. So detouring time over the traffic light (at least 3 moves) was almost same as waiting time at the traffic light.

Inform the Driving Agent

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?

ANSWER:

This is a navigation problem with a deadline. The mission is to reach a given destination with a given time in a given environment (grid space).

So I think, the following states are appropriate (ordered by importance):

- `Next waypoint` - The next waypoint location that is relative to its current location and heading. Since current location and distance are unknown for the agent, the waypoint is

the most important guideline to navigate in the environment. Just by following the 'next_waypoint', the agent could reach the destination successfully, except the traffic light rule.

The agent should learn other optimal moves while following the waypoint. But it may be punished if it goes in other direction. Sometimes going in other direction, specially in a diagonal path to the destination, may minimize the reach time.

- `Light`, `Oncoming cars` are important inputs to obey traffic rule. We have no option to exclude these inputs. Initially I thought to hard code the traffic rule in my source code.
- `Deadline` - The current time left from the allotted deadline. It represents how many moves the agent has left, at maximum.

  The agent has to select a correct action according to deadline. The deadline must be enough value for the task, otherwise it would be a mission impossible. So we may ignore this input and follow the waypoint.

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?

ANSWER:

We need to keep state combinations as few as possible, because many state combinations will take longer time to "learn". Specially 'Deadline' will blow up the size of the state space, because it is an integer value and it could generate huge number of combinations of state.

The `Next waypoint`, `Light` and `Oncoming car` inputs are the most important for this task. So these are mandatory states.

As explained before, we could ignore 'Deadline' input.

So total number of states could be calculated as the following:

```
Total number of states = 3 waypoints x 2 lights x 3 oncoming car directions x 3 left car directions x 3 right car directions = 162
```

In total, there is 162 states. I think, having 162 states (162 x 3 actions) is small size for the problem, and the agent should learn very quickly.

Implement a Q-Learning Driving Agent

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?

At beginning of simulation, the agent makes a lot jumping and round movements, and over time it starts doing long and frequent stops at intersections, but was going correct direction.

Most of time, the agent was going in the direction suggested by the planner. But I was expecting more interesting movements. For example, to minimize waiting time, the agent could select another route on the red light or when oncoming car appears.

Overall the success rate is increased a lot. It was 'learning' very quickly from the past history.

Note:

I used some code snippet from this blog: https://studywolf.wordpress.com/2012/11/25/reinforcement-learning-q-learning-and-exploration/

It was very well explained and easy to understand. And saved a lot time to implement Q-Learning part.

Improve the Q-Learning Driving Agent

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

I did a experiment on the success rate using the discount factor and the learning rate. The following is a random output from the experiment:

```
Success rates over (alpha, gamma)
(0.1, 0.1) :  99
(0.1, 0.2) :  99
(0.1, 0.3) :  98
(0.1, 0.4) :  98
(0.1, 0.5) :  99
(0.1, 0.6) :  99
(0.1, 0.7) :  98
(0.1, 0.8) :  98
(0.1, 0.9) :  97
```

```
...
```

The most high success rate was when alpha is below than 0.3. It may be confirming that the recommended value of alpha for a stochastic problem is 0.1. So I just followed the common practice and set the learning rate to 0.1.

When alpha is 0.1, the success rate was greater than 90%, most of time. But sometimes it drops down to 60%, usually at higher gamma. I couldn't find any reason or mistake. May be during the simulation it learnt some 'wrong lesson' and that lesson leads to a bad result.

QUESTION: Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?

ANSWER

The agent's goal is to learn how to reach a given destination with a given time. I think, the success rate (reached to the destination within a deadline) is the most important indicator for this problem.

Also the agent should obey all traffic rules. In the beginning of the simulation, the agent makes frequent non-valid moves against traffic rule, and it gets better for the later trials. So I measured illegal moves for last 10 trails.

```
Performance for last 10 trials
------------------
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
{'illegal': 0, 'success': True}
```

It was almost 100% correct moves, but very occasionally it makes illegal moves, for the last 10 trials.

For several simulation, the agent's success rate shows more than 90% when alpha is 0.1, for last 10 trails it was around 99%.

I think, the optimal configuration for this problem is set alpha to 0.1 and gamma to 0.5. On this configuration, the agent was performing well.