SmartCab

Ryosuke Honda

2016/06/23

1 Implement a basic driving agent

Implement the basic driving agent, which processes the following inputs at each time step:

- Next waypoint location, relative to its current location and heading.
- Intersection state(traffic light and presence of cars) and
- Current deadline value(time steps remining)

And produces some random move/action [None,'forward','left','right']

2 Identify and update state

3 Implement Q-learning

One of the most important breakthroughs in reinforcement learning was the development of Q-learning.Its simplest form,one step Q-learning, is defined by

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r(s,a) + \gamma \max_{a'}(Q'(s',a'))) \tag{1}$$

α:Learning rate
γ:Discount rate
s stands for state, **a** stands for action, **s'** stands for the next state, **a'** stands for the all actions

First, set the parameters(α,γ) and environmental reward.Then Initialize the Q table to zero. For each episode, select a random initial state and do the following things until the agent has reached the goal.

- Select one among all possible actions for the current state.
- Using this possible action, consider going to the next state.
- Get maximum Q value for this next state based on all possible actions

- Compute Q value
- Set the next state as the current state

The Gamma and Alpha parameters has a range of 0 to 1. If Gamma is closer to zero, the agent will tend to consider only immediate rewards. If Alpha is closer to zero, the agent will tend to consider only past experience(don't learn).

The reward and penalty for this task is as follows.

4 Enhance the driving agent

Report what changes you make to your basic implementation of Q-learning to achieve the final version of the agent.How well does it perform?

When the agent just started, it has no q-value since the q-table is set to be 0. Therefore, at first I need to set the move randomly(ϵ-greedy). With the probability of ϵ, the agent moves randomly. This prevents the agent from falling into the wrong choice.

First, I implemented random action.The agent just moves random action("None","Forward","Left","Right") and doesn't learn anything from experience.
In the agent.py file, I set ϵ to be 1(The agents moves randomly). I define "Success Rate" to see the agent's performance.

[Success Rate]=[No.trials achieve goal before deadline]/[No. trials] For these trials, the number of trials are 100.

Second, I implemented in the following conditions.

Alpha and Gamma in this trial is constant. The result in this trial is following.

Third, I implemented in the following conditions.