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']. The rule and the reward of the agent are following.

- The start and goal location changes in each episodes.
- When the car violates the traffic rule, the agent will get the negative reward(-1).
- If the car moves without violating the traffic rule but the agent doesn't follow the waypoint,the agent will lose 0.5 points.
- When the car moves without violating the traffic rule and follows the waypoint, the agent will get 2 reward points
- When the car gets to the destination within the deadline, the agent will get 10 reward points.
- When reaching goal within the deadline, the agent will always get 10 reward points, which means that reaching the destination doesn't always get the maximum reward for each episode.

2 Identify and update state

The agent can sense the information of inputs of traffic light, oncoming car, turning right and turning left which all of other agents have and what's more the agent has the information of next waypoint. Since the start point and destination point will differ from each episode, it's not good to keep track of their points to get high rewards. We should restore the information that keeps the same as the episode changes. Therefore, we should keep track of the traffic information which is traffic light, oncoming car,right and left.

Those are the information which can sense the other agents can sense. The agent should keep the memory of waypoint. These information is necessary not to violate the traffic rules and to get high rewards.

As explained above, the Q table for this questions will be like this.

Table 1: States for Q table

States	Values	Dimentions
Traffic light	Red Blue	2
Waypoint	None Forward Left Right	4
Oncoming	None Forward Left Right	4
Left	None Forward Left Right	4
Right	None Forward Left Right	4

The possible action for the agent is "None","Forward","Right","Left" The dimension of the q table is $2 \times 4 \times 4 \times 4 \times 4 = 512$.

Putting the right into the q table is a must. It seems that turning right while the signal is red is violating the traffic rules on the surface but that's not true. American traffic rules allow us to turn right while the signal is red.Therefore, the state should be kept in the q table. In this game, the rule is followed by the American traffic rule and the reward is also consistent with it(meaning not turning right while the traffic right is red will get penalty to the agent.).

I don't include the deadline into q table because of the curse of dimensionality. Deadline is defined by 5 multiplying by the distance(the distance between the starting point and the destination). The distance are more than 4, that means the deadline will be at least 25 and more. If the deadline is 25 which is the minimum value, the dimension will be $512 \times 25 = 12,800$. This dimensionality is still very high even though the deadline is the minimum value.Therefore, putting the deadline into the q table isn't feasible.

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:state, **a**:action, **s'**:next state, **a'**: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).

Here, I introduce the code snippet of the ϵ-greedy Q learning.

```
# TODO: Select action according to your policy
epsilon=0.1
#epsilon=1.0
rand=random.random()
if rand<epsilon:
    action=random.choice(Environment.valid_actions[0:])
    print "RANDOM ACTION"
else:
    if max(self.q_table[self.state].values())==0:
        action=random.choice(Environment.valid_actions[0:])
        print self.q_table[self.state]

    else:
        action = max(self.q_table[self.state],
            key=self.q_table[self.state].get)

    print action
```

Figure 1: Python code snippet for deciding action policy

In this experiment, I use ϵ-greedy and set ϵ to be 0.1. The agent chooses random action with the probability of 0.1. If the maximum of q table is 0, the agent also chooses random action. If the maximum q-table is more than 0, the agent chooses the maximum value and moves toward it.

```
# Calculate the Q_value
q_value = (1 - alpha) * self.q_table[self.state][action] + \
        alpha * (reward + gamma * max(self.q_table[state_new].values()))
# Update the Q_table
self.q_table[self.state][action] = q_value
```

Figure 2: Python code snippet for updating Q table

The code above is the calculation of fomula(1) and code snippet for updating Q table. I'll discuss the agent's behavior in the next section.

4 Enhance the driving agent

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.

(I) 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 Ratio" and "Penalty Ratio"to see the agent's performance.

"Success Ratio"="No.trials achieve goal before deadline" ÷"No. trials"

"Penalty Ratio"="No. penalty actions" ÷ "No. actions"

For these trials, the number of trials are 100. In this experiment,I set the conditions as follows.

Success ratio in this trial is 0.20 and the penalty ration is 0.575. The cumulative sum of negative reward is -1185.0, and the cumulative sum of total reward was 3.5. The total number of action from the first episode to the end of the episode is 2814.

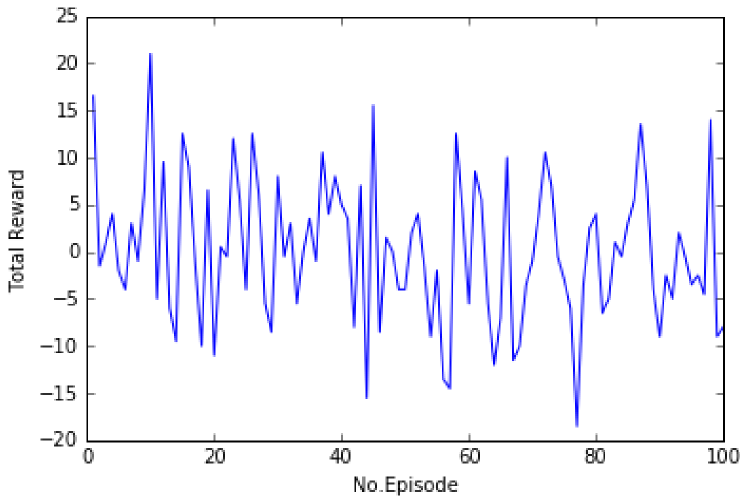

Figure 3: Random Action

As the Fig.3 shows,the total rewards with this trial contains negative values. Without learning from their experience, the agent won't get high reward.

(Ⅱ) Second, the parameters of α and γ are 0.5 and 0.9 respectively. With this condition, the success rate is 0.72 which is higher than the random action. The penalty ratio in this trial is 0.359 which is much smaller than the random action.

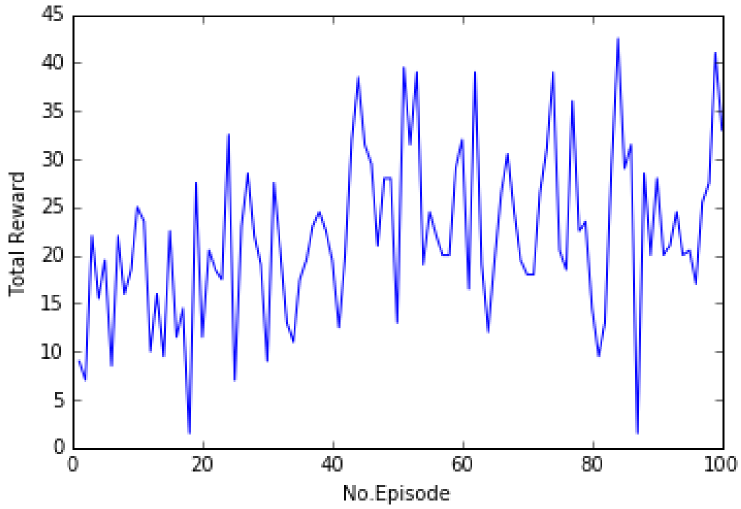

Figure 4: Constant α and γ

The Fig.4 shows,the total rewards for each episode are all positive and the cumulative sum of total reward of all episodes are 2195.5. The cumulative sum of the negative reward is -389.5 and the total number of action is 1889 which is shorter than the first trial, meaning the agent reaches to the goal by shorter roots.

(Ⅲ) Third, the parameters of α and γ are as follows.

$$\alpha = \frac{1.0}{1.0 + time} \tag{2}$$

$$\gamma = \frac{1.0}{1.0 + deadline} \tag{3}$$

With this condition, the success rate is 0.86 which is higher than the trials before.The penalty ration in this trial is 0.325 which is smaller than the trials before.

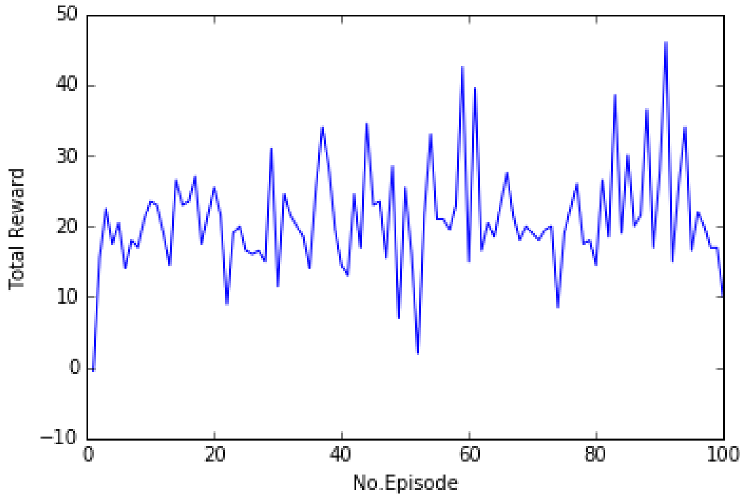

Figure 5: Time dependent model

The Fig.5 shows the total rewards for each episode are all positive and the cumulative sum of total reward of all episodes are 2099.0.The cumulative sum of total penalty in this trial is -437.5 which is higher than the trial before. The number of total actions is 1688 which is the shortest steps from the beginning to the end.The agent takes the shorter way to the goal with containing more negative reward actions.

(IV) Final version

The condition in this implementation is as follows. α is equal to the fomula (2) and γ sets to be 0.9.

With this condition, the success rate is 0.86 which is the same value as the trial 3. However, the penalty ration in this trial is 0.300 which is the smallest of all trials above.

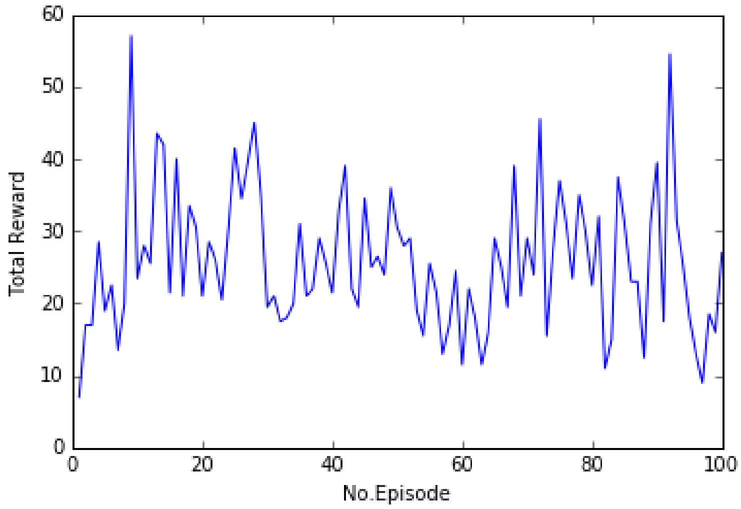

Figure 6: Constant γ and time dependent α

The Fig.6 shows,the total rewards for each episode are all positive and the cumulative sum of total reward of all episodes are 2580.5. The cumulative sum of total reward of all episodes are conspicuously higher than the other trials. The cumulative sum of total penalty in this trial is -283.0 which is the smallest value of all trials and the total number of actions in this trail is 1670 which is also smallest number.

5 Discussion

In the previous section, I introduced 4 models.The random walk model(I) is the case which doesn't implement the q-learning. Second, I made the constant α and γ value models. This model improves the success rate and penalty ratio compared to the first model and. Third model which changes α and γ depending on time. As time goes by, those values are decreasing. In this model, the success rate and penalty ratio has improved compared to the second model. However, the cumulative sum of the total reward has decreased. This means that the second model takes longer way to get to the destination to get positive reward. The third one takes shorter way to get to the destination with getting negative reward. The final model has constant γ value and time-dependent α value. This experiment shows the high success rate and also shows high cumulative sum of total reward. The penalty ratio in this trail is the smallest of all. What's more, the actions from the beginning to the end is the shortest. This means this agent tries to reach the goal with shorter way avoiding the negative rewards.

From these experiment, the γ value should set to high value to get better success rate and cumulative sum of total reward.