Report

April 24, 2016

1. In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?

Answers: By applying random actions to the cab, it will generate random actions on the map, but it will still obey the traffic rules since the act function in enviroment.py will not perform that random action if the agent breaks the rule. The agent will finally ready the target location by a small chance since the map in finite when set enforce_deadline to false. The rewards are negative.

2. Justify why you picked these set of states, and how they model the agent and its environment

Answers: For the baseline, the states feature I choose include traffic light, incoming traffic, left traffic, right traffic and the action the agent intend to perform. I choose all of them to fully capture what's going on there. Deadline is not included here because I think it's more important for cab driver to obey the traffic rules. The agent already includes the planner in the state to give the next waypoint, if the agent can follow the planner it will not miss the deadline.

3. Implement Q-Learning. What changes do you notice in the agent's behavior?

Answers: It begins to reach the destination on time and get positive final reward. But with a high chance the agent will stuck in a rut to just make right turns in a loop all the time and miss the deadline. Maybe it's because I set the learning rate(0.5) and gamma(0.7) too high so that the agent by a chance make a lot of right turn at the beginning and quickly learns it can make more right turns to collect a reward of 2. Even in the situation it should make a left turn, it sacrifices to get a negative reward -0.5 (since action != agent.get_next_waypoint()) in order to make three more right turns because the q table shows more q values.

So I reduce gamma to 0.5 and alpha to 0.2, then the failure rate reduces to around 4%.

4. Enhance the driving agent

4. 1 Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform?

Answers: To get the best performance, one way is to do a grid search of alpha and gamma, in this way we can find a good parameter combination but only for this model, which means these parameters can not be generalized, so I just skip this.

The other way is to use an exploration function. I implement it in function *next_action_index_with_exploration*. The basic idea is cache the time of state->action the agent have tried. If it doesn't meet a certain threshold(*self.NE*) then we use a certain positive q_value for that state->action (*self.reward_plus*). After we meet enough numbers of that state->action status, we begin to use the values we've learned in q_table. I set *self.NE* to be 3.

The third way is to reduce the size of all the states. By reduce the state size, the q table can be updated more faster because "same" states will be combined, we don't need to run a lot of trial in order to meet each state and update the q values for them all individually. I implement a state filter in function *inputs_filter* following the rules below:

- Ignore right traffic
- If light is green, ignore left traffic
- If light is red, ignore oncoming traffic

I define the failure rate as the percentage that the agent fails among all the trails, in order to better capture the rate, I define *run_multi* function to get 100 fresh starts, and keep the 100 trials. Since there're lots of randomness involved, by just increasing the number of trails won't prove anything since the agent rarely fails after it get a almost converged q table. Here's the result:

	No Filter(Baseline)	With Filter	With Exploration	With Filter & Exploration
AVE Fail rate	4.4%	3.64%	4.48%	**3.45%**
AVE Q table size	34	**13**	35	14

We can see that with filter and exploration we can get the lowest failure rate. With filtering only we can significantly reduce the q table size from 34 to 13, as well as the failure rate from 4.4% to 3.64%.

4. 2 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?

For the baseline without filtering and exploration, I got one policy like this, first 5 columns list the state's features, last 4 columns list the 4 actions Q values:

light	oncoming	right	left	next way point	A_Stay	A_Foward	A_Left	A_Right
Green	None	None	None	Left	0	0	0	**0.477**

- When the light is green, no oncoming/left/right traffic and the agent tends to go left, but according to the Q table, it will choose to turn right(0.477), which is no good.

With filtering and exploration, the agent can get much better policies:

light	oncoming	right	left	next way point	A_Stay	A_Foward	A_Left	A_Right
Green	None	None	None	Left	0	-0.252	**7.180**	-0.0069
Green	Forward	None	None	Left	**0.453**	0	0	0

- When the light is green, no oncoming traffic and the agent tends to go left, then just go(7.18)
- If there's oncoming traffic going forward, then stay(0.45) instead of going left(0)

Because of the randomness sometimes filter+explore model tends to make more right turns to collect rewards sometimes not. Which means the current Gamma(0.5) is still too high. So I change it to 0.1.

	gamma	light	oncoming	right	left	next way point	A_Stay	A_Foward	A_Left	A_Right
1	0.5	Red	None	None	None	Left	**0.162**	0	0	0
2	0.5	Red	None	None	None	Left	0	-0.512	-0.504	**1.912**
3	0.1	Red	None	None	None	Left	**0.118**	-0.491	-0.489	-0.224
4	0.1	Red	None	None	None	Left	**0.201**	-0.488	-0.489	-0.196

I also run 100 fresh starts, each with 100 trials, the failure rate changes to:

	No Filter(Baseline)	With Filter	With Exploration	With Filter & Exploration
AVE Fail rate	1.82%	**1.51%**	3.31%	2.52%

It turns out when gamma is small, the filtering plays a huge role because smaller gamma means each state get less "impact" from next state each time when updating the q table. Without state filtering, the large state space will needs more trials to update, so the q table will be far from converge. I increase the number of trials from 100 to 1000 to better capture the failure rate:

	No Filter(Baseline)	With Filter	With Exploration	With Filter & Exploration
AVE Fail rate	0.836%	1.306%	1.039%	**0.5382%**

It shows with filtering and exploration together, the agent can get best performance.