Implement a basic driving agent

The agent just walked around in gird_city, and sometimes arrived target location, some of them didn't make it(because even if turn off enforce_deadline the game still going to end when deadline go down to -100. If the game run forever, I believe every agent will arrive target location eventually).

Identify and update state

According to right-of-way rules, we should pay attention to at least these states below:

Agent can't turn left when: ### 16 states in total
(light:green, oncoming:forward, left:whatever, right:whatever)
On a green light, you can turn left only if there is no oncoming traffic at the intersection coming straight.

Agent can't turn right when: ### 28 states in total
(light:red, oncoming:left, left:forward, right:whatever)
(light:red, oncoming:left, left:whatever, right:whatever)
(light:red, oncoming:whatever, left:forward, right:whatever)
On a red light, you can turn right if there is no oncoming traffic turning left or traffic from the left going straight.

If agent do not obey the traffic rules, it will get a large penalty, and learning to prevent make mistake again.

When agent choose an action that different from next_waypoint which given by planner, it will get a small penalty, and learning to not choose an action different form next_waypoint.

In some special cases, for example:
(light:green, oncoming: forward, left:None, right: None)
and next_waypoint ask agent go left which will break traffic rules. When agent face such a situation, it will learn to choose stay in current location(action:None), because whatever moving action(forward, left, right) it choose, it will get penalty, but if agent choose to not move, it will get 0 which still better than penalty.

When agent face some states that no belong to any of above, it will get reward which will encourage agent move to target location, and if agent arrive the target location it will get a great reward, that will be an great encouragement.

It is important to point out that I decide to not include deadine in state, because if deadline include in state, the state space will get too sparse and Q values won't converge. I also had tried to break deadline into three values(0%~33%,33%~66%,66%~100%), then included

deadline in state, and ran some experiments, but it didn't reject null hypothesis(there is no difference between include deadline in state or not), so I decided to not include deadline.

Implement Q-Learning

Agent still walk randomly at the begining of training, as time past, agent will arrive target location with higher probability by choose a better way. As training times raise, agent will learn to stop rather pick a further way. At the ending of training, for example ,the 90th trial, agent will pick a way to target location which almost perfect. But seldom agent may choose a worse action, because it never meet such situation before.

Agent's behavior have such great changes all rely on Q-learning.

$$\widehat{Q} \leftarrow_{\alpha} r + \xi max_{a'} Q(s', a')$$

The formula explain why Q-learning works, it can learn form reward/penalty, make agent behave in a reasonable way.

Here are somethings additional:

- there is 0.1 chance that agent pick actions randomly at the first trial, and it will decay by multiplying 0.9, so in the later trials agent won't choose actions randomly.
- Actually agent will learn well even if I ask it choose best action base on policy all the time(set self.randomaction=0), but agent will make a bit more mistakes in the later trials than ask agent pick random actions in the early trials, because via early exploration(random actions) agent can handle unexpected states a little better, that's why I decide to let agent have 0.1 chance pick random actions.
- learning rate will decay by multiplying 0.99 at each trial, because we don't need learning rate stay high in the later trials.

Enhance the driving agent

I have tuned learning rate, random action chance, and discount factor. Learning rate too high or too low may not help agent learn approximately, though some control experiments I figure out 0.7 maybe is the best. There is no way to set a high random action chance, because by pick lots of random actions, agent act worser, exploration too much but less exploitation just don't make things work. It's funny about tune discount factor, when agent learning without discount, it behave wried, do stupid stuff like going in circles in order to accumulate more reward before reaching the destination; when agent learn without history(discount=0), it will try to get to the destination ASAP even if disobeying traffic rules.

After tuning parameters, my agent can figure out a policy what can lead it to target almost perfectly in about 20 trials, it learns very fast. Agent still make mistake seldom even after 50 trials training because state space is too big, there always exist some states which agent may never meet since training start, but it appears very few so I think it is acceptable.

Yes, I think my agent get close to finding an optimal policy. If lucky enough, my agent can reach the destination in the minimum possible time and not incur any penalties, and it turns out

appear very frequent. But there is not guarantee that agent can make the best all the time, because state space is too big to cover by limit training.