# Smart Cab Project Report
# Levin Jian, May 2016

# 1. Implement a basic driving agent

## 1.1. Agent accepts inputs

This part is implemented in LearningAgent::selectAction method in agent.py file.

```
def selectAction(self, state):
#         return self.next_waypoint
        return self.getRandomAction()
```

## 1.2. Produces a valid output

This method randomly choose an action out of list [None, 'forward', 'left', 'right']

```
def getRandomAction(self):
        listOfActions=[None, 'forward', 'left', 'right']
        return random.choice(listOfActions)
```

## 1.3. Runs in simulator

The driving agent can run in the simulator without errors. You can run it by launching the agent.py file. Regarding simulator display, two modifications are made to help us better understand the movement of the agents.

1) For each trial, the simulator screen will be displayed twice. The first time shows the state which the learning agent is in, and the action it's about to take; the second time shows the reward the agent get out of its last action.
2) In the simulator screen, the texts about dummy agents are removed, and only the text about the learning agent is retained so that it is easier to identify the learning agent. Also the text now means the action the learning agent is about to take, instead of its next_waypoint. You can see it by launching the agent.py file.

# 2. Identify and update state

## 2.1. Reasonable states identified

This part is implemented in LearningAgent:: getCurrentState method in agent.py file.
The state is a tuple (self.next_waypoint, inputs['light'], inputs['oncoming'], inputs['right'], inputs['left']). This is because all these information can help the learning agent determine next action. The “deadline” information should also be useful, but it’s not included in state because, if it’s included, there will be too many possible states for the learning agent to explore during the training, and yet we are required to do only 100 trials at the utmost.

## 2.2. Agent updates state

The driving agent can update its state when running. You can see it by launching the agent.py file.

# 3. Implement Q-Learning

This part is implemented by class LearningAgent_Basic_Qtable in agent_basic_qtable.py file

## 3.1. Agent updates Q-values

The Q value update are done by two methods, updateQTable_1, updateQTable_2.
updateQTable_1 is executed after the learning agent move, At this point, we will not know the next state(since the traffic light would change, and the other dummy agents would move). So this method merely memorize current state, action and reward. And then after the traffic lights change, and other dummy agents complete their move, we will know the next state , and updateQTable_2 will be executed. The Q value update formula is as below:

q_new = (1- alpha)*q_old + alpha*(self.current_reward + gamma * q_max)

## 3.2. Picks the best action

After training, the learning agent can utilize Q table to pick the best action. This is implemented in LearningAgent_Basic_Qtable:: getQBestAction

```
def getQBestAction(self, state):
        tempList = []
        listOfActions=[None, 'forward', 'left', 'right']
        for action in listOfActions:
            tempList.append(self.qtable[(state, action)])
        m = max(tempList)
        maxindexes =  [i for i, j in enumerate(tempList) if j == m]

        return listOfActions[np.random.choice(maxindexes)]
```

## 3.3. Changes in behavior explained

After applying Q-Learning algorithm, the agent's ability to choose the 'right' action significantly improves:

1) For the basic driving agent, it randomly choose action, regardless of inputs
   `Test result: Average Discounted Reward,0.38 Completion Rate,0.19 Average Penalty,-11.83 Average Deadline,1.92`
2) For the learning agent, it uses Q table to guide its action choice:
   Test result: Average Discounted Reward,27.405 Completion Rate,0.94 Average Penalty,-2.20652173913 Average Deadline,15.55

The reason for the improvement observed is very obvious, the agent is now using Q Table (being tantamount to experience obtained during its training) to guide its action, instead of acting randomly.

# 4. Enhance the driving agent

Grid search is used to find best hyper parameters and enhance the basic Q-learning. This part is implemented by class FineTuneQTable in agent_basic_qtable.py file

## 4.1. Agent learns a feasible policy within 100 trials

After applying grid search, the best parameters has below performance.

Test result: Average Discounted Reward,28.75375 Completion Rate,0.925 Average Penalty,-2.29003558719 Average Deadline,14.6233333333

Out of the 1200 trials, the learning agent is able to reach the destination 92.5% of the time before deadline is reached, and its average discounted reward is 28.75375.

## 4.2. Improvements reported

Below parameter combinations are tried:

```
alphas = [0.1,  0.5]
gammas = [0.5,1]
epsilons = [None, 0.5,0.8]
```

```
And the result is as below:
```

| No | score | alpha | gamma | epsilon | finalstringrepresentation |
|---|---|---|---|---|---|
| 0 | 23.74583 | 0.1 | 0.5 | None | Test result: Average Discounted Reward,23.7458333333 Completion Rate,0.990833333333 Average Penalty,-0.840388007055 Average Deadline,15.8408333333 |
| 1 | 22.42083 | 0.1 | 0.5 | 0.5 | Test result: Average Discounted Reward,22.4208333333 Completion Rate,0.994166666667 Average Penalty,-0.859872611465 Average Deadline,16.6433333333 |
| 2 | 23.66583 | 0.1 | 0.5 | 0.8 | Test result: Average Discounted Reward,23.6658333333 Completion Rate,0.990833333333 Average Penalty,-0.820512820513 Average Deadline,16.5366666667 |
| 3 | 23.58333 | 0.1 | 1 | None | Test result: Average Discounted Reward,23.5833333333 Completion Rate,0.985833333333 Average Penalty,-0.902210884354 Average Deadline,16.8733333333 |
| 4 | 23.13583 | 0.1 | 1 | 0.5 | Test result: Average Discounted Reward,23.1358333333 Completion Rate,0.986666666667 Average Penalty,-1.01660839161 Average Deadline,16.195 |
| 5 | 23.74208 | 0.1 | 1 | 0.8 | Test result: Average Discounted Reward,23.7420833333 Completion Rate,0.988333333333 Average Penalty,-0.908717105263 Average Deadline,16.6658333333 |
| 6 | 23.1025 | 0.5 | 0.5 | None | Test result: Average Discounted Reward,23.1025 Completion Rate,0.99 Average Penalty,-0.967332123412 Average Deadline,16.6958333333 |
| 7 | 28.64042 | 0.5 | 0.5 | 0.5 | Test result: Average Discounted Reward,28.6404166667 Completion Rate,0.918333333333 Average Penalty,-2.18529929577 Average Deadline,14.5583333333 |
| 8 | 28.75375 | 0.5 | 0.5 | 0.8 | Test result: Average Discounted Reward,28.75375 Completion Rate,0.925 Average Penalty,-2.29003558719 Average |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Deadline,14.6233333333 |
| 9 | 14.63542 | 0.5 | 1 | None | Test result: Average Discounted Reward,14.6354166667 Completion Rate,0.220833333333 Average Penalty,-3.69217830109 Average Deadline,2.09416666667 |
| 10 | 9.974583 | 0.5 | 1 | 0.5 | Test result: Average Discounted Reward,9.97458333333 Completion Rate,0.635 Average Penalty,-11.4744897959 Average Deadline,8.48583333333 |
| 11 | -3.15208 | 0.5 | 1 | 0.8 | Test result: Average Discounted Reward,-3.15208333333 Completion Rate,0.176666666667 Average Penalty,-12.3909395973 Average Deadline,1.96833333333 |

```
Best parameters(the one with highest score)
```

| | | | | | |
|---|---|---|---|---|---|
| 8 | 28.75375 | 0.5 | 0.5 | 0.8 | Test result: Average Discounted Reward,28.75375 Completion Rate,0.925 Average Penalty,-2.29003558719 Average Deadline,14.6233333333 |

```
Basic Q learning
```

| | | | | | |
|---|---|---|---|---|---|
| 0 | 23.74583 | 0.1 | 0.5 | None | Test result: Average Discounted Reward,23.7458333333 Completion Rate,0.990833333333 Average Penalty,-0.840388007055 Average Deadline,15.8408333333 |

## 4.3. Final agent performance discussed

An optimal policy should always move towards the direction of next_waypoint unless a move is not allowed due to traffic light or oncoming dummy agents, in that case, it should perform “None” action.

The final driving agent has below parameters and test result:

| | | | | | |
|---|---|---|---|---|---|
| 8 | 28.75375 | 0.5 | 0.5 | 0.8 | Test result: Average Discounted Reward,28.75375 Completion Rate,0.925 Average Penalty,-2.29003558719 Average Deadline,14.6233333333 |

Visualize the final agent in the simulator display, we can see that the final agent roughly act

by the optimal policy.
One interesting point to note is that although the final agent(NO 8) has higher Average Discounted Reward than the basic q table agent(NO 0), it has lower completion rate. This is because Q learning is all about maximizing discounted reward. To push the final agent towards having higher completion rate, we will need to adjust relevant reward mechanism like penalizing not being able to reach destination.