Reinforcement Learning-traffic light simulator

COMP9417-Machine learning Assignment 2

YUCHEN YAN
Z5146418
This project is done by Yuchen Yan individually.

Introduction:

The traffic light control problem have very meaningful usage. As the urbanization of the world keep growing. The traffic problem becomes more and more common and serious. For example, in those biggest city in the world such as New York, Los angles and also big city in China, such as Beijing, shanghai. This can not only wasting the time of people when crowded in the road for hours. But also will increase the energy wastes. Why we say that, because normally for a car, the most wasting gasoline way of driving is that keep braking and starting. As the traffic increase, then the energy use can be increase so much. In order to improving the road condition. We can rearrange the traffic light control system base on the situation of the particular road. In this project, this traffic light control system were implemented by reinforcement learning system, which can make changes of the traffic light by analysis the current condition of this road. The specific method used for learning in this study is the Q-learning algorithm with epsilon greedy exploration. Let's talk about this problem following.

Reinforcement Learning:

Reinforcement learning is one of the big three machine learning topic. The other two are supervised learning and unsupervised learning. Reinforcement learning cancerned about the agent of a system how to act that can maximum the reward from environment. The problem, due to its generality, is studied in many other disciplines, such as game theory, control theory, operations research, information theory, simulation-based optimization, multi-agent systems, swarm intelligence, statistics and genetic algorithms. Markov decision processes is a environment used by reinforcement learning. It mainly has two components which are agent and environment. The environment will give the agent a state which can describe the current situation about the environment, and also reward about the state. and the agent should give the environment a action to response. Then the cycle just keep looping until the end. Reinforcement learning just want to give a the best why of choosing the next action that can maximum the reward.

Definition of Q-learning:

Q-learning is a method the used to measure the quality a out each state. The 'Q' means quality in some way. In my implementation I used a two dimensional array to store the Q value. One dimensional represents the states and another is the action.
The function that can calculate the Q value in this assignment is following:

$$\delta = \alpha_{s,a}\{\mathbf{r}_{s,a} + \gamma_t \cdot \mathbf{MAX}[\mathbf{Q}_{t-1(s',a')}] - \mathbf{Q}_{t-1(s,a)}\}$$

Let me explain a kittle bit about this equation, this equation is a combination about state s, action a and also reward r. The answer of this equation is the incremental about the Q value, after each time of calculation the Q value will be added by the incremental value, the learning rate is the alpha in me implementation the value is 0.1. r is the reward received for taking action a. Gamma is the discount rate in the interval, which applied to the future rewards. MAX[] is the previously estimated Q-value following the optimum policy starting in state s'. Qt-1 previous estimate of the Q-value of taking action a while in state s. This particular training rule is relevant to stochastic environments such as the traffic environment in the case study outlined in the next section. Decreasing the training rate over time is one of the conditions necessary for convergence of the Q-function in a stochastic environment. If there is a penalty occurs then the MAX will be replaced by MIN. The updated estimate of the Q-value is then stored for later reuse. The Q-values may be stored in an unaltered form in a look-up table, although this requires a significant amount of memory.

Implementation:

The 2-D animation was implemented by the module ***tkinter .***
There are four two-way roads and also four intersections. Here are a little bit explanation.
The overall picture is a canvas which the width and height are 400 * 400.
The green part was implemented by nine squares. It represents the green parts, beside the roads. The roads actually is the white background. And the car is implemented by small squares. On the left top of the image the time-steps is showed on it. The traffic light is small cycles on each intersection.

At every time step, the canvas will be updated once. And for the car on the horizontal road from left to right, at every update the square will be moved one step use method move(item,4, 0) , in this case Is 4 pixels. And for the car on the horizontal road from right to left, at every update the square will be moved one step use method move(item,-4, 0), in this case Is 4 pixels. And for the car on the vertical road from up to down, at every update the square will be moved one step use method move(item,0, 4), in this case Is 4 pixels. And for the car on the vertical road from down to up, at every update the square will be moved one step use method move(item,0, -4), in this case Is 4 pixels.
The procedure above will be implemented by one iteration. And at each iteration the canvas will be updated one time use the method canvas.update().
The new cars will be appeared using this equation: t % [randint(10) + 10 -x] == 0. When the equation is true then a car will be entered by each road.
After the whole structure of the canvas is settled. Then we should implement the Q-table. The q-table is actually a two-dimensional array. Where the first dimension is the state and the second dimension is the action of each state. every time the user should choose the action which have the most Q value.
The **state** of this implementation is as follow:
1. sum of all the horizontal roads(closest car position from intersection for road 1 (0-8, 9 if no cars))
2. sum of all the vertical roads(closest car position from intersection for road 1 (0-8, 9 if no cars))
3. light setting (ie 0-green, 1 red for one of the roads)
4. light delay (0-3)

Action:
Dicide to switch or not.
Reward -1.0 if a car is stopped at a red light on either road, zero otherwise.

Optimise discounted sum of future reward.

Use **discount factor**: gamma = .9

Use **learning rate**: alpha = .1

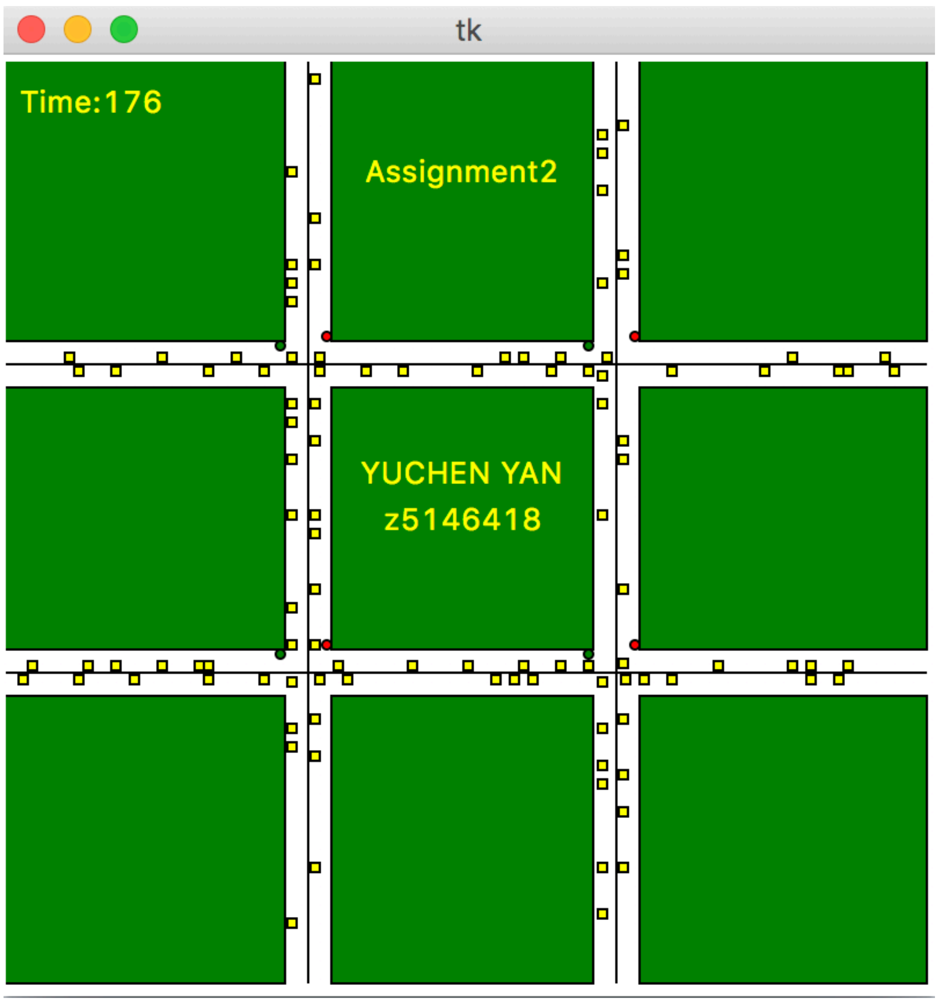
Epsilon-greedy exploration 10%

The Q value will be updated after each iteration by the above parameters. And then continue looping final all the Q value will be converge to a constant value after infinity looping.

Extension features:

```
1. Include 2-way traffic
2. Vary RL parameters
The state of this situation was increased in order to make the two
way roads.
3. Include extra lanes
4. Increase intersections to 4
5. Try different state space descriptions
```

Output explanation:



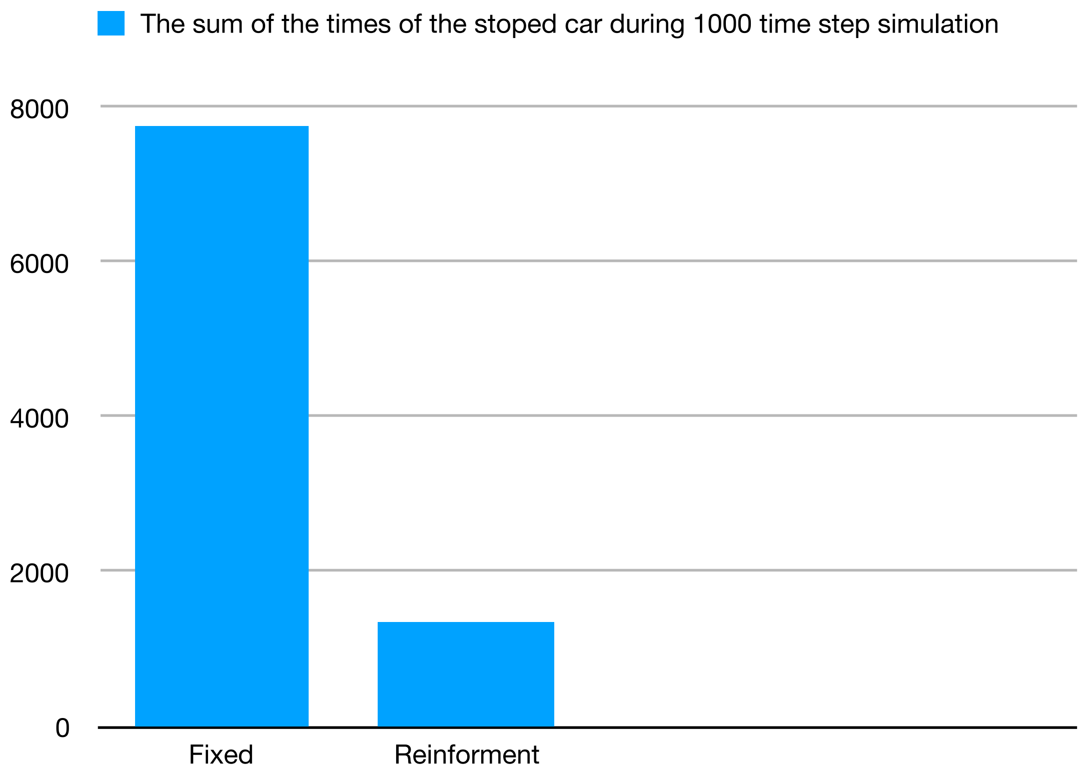
This is the output of my project. As you can see, there are four two-way roads and also four intersections. The lanes are represented by white lines. And the green part is the green belt.

Performance measure:
The performance measure for all of my experiments is the sum of the number of queued (stopped) cars in all lanes within each timestep. I then further evaluate this performance measure as an exponential moving average with a smoothing factor = 1 – –1 where is the scaling constant and is equal to 300 iterations. The exponential moving average is calculated using the following equation:

$$s_t = \alpha x_t + (1 - \alpha) s_{t-1}$$

where is the new exponential moving average at timestep , is the number of stopped cars during this timestep, and –1 is the exponential moving average at the previous timestep – 1.

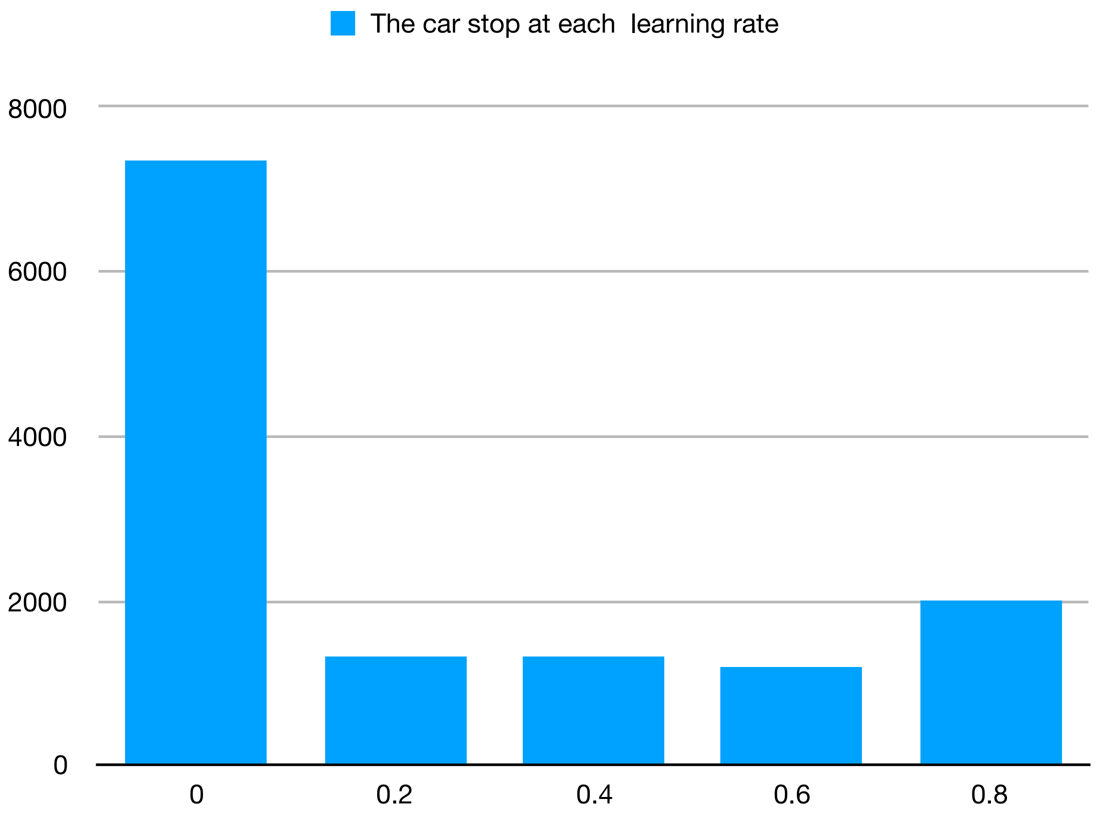
Let we show the result in a bar picture:
After 1000 times iteration:



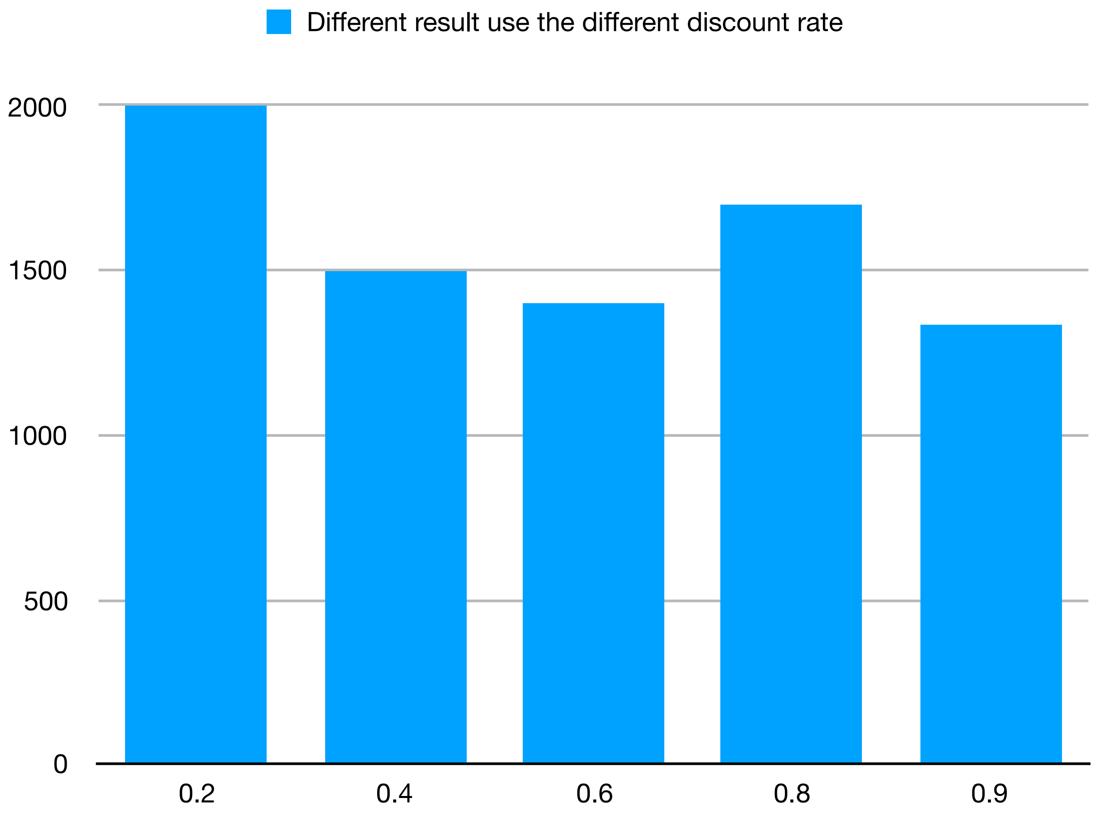
As you can see the performance is increased nearly five time. Then the result of the traffic light system will be increased. And then as a result the energy will be saved so many times.

Then we can change the parameter used in this project.

First I want to change the learning rate value to 0, 0.2, 0.4, 0.6, 0.8, 1.0



Then we can also change the gamma value (discount factor) to see the result:



The result are showed on the above bar chart.

Future work:

This model is just a 2d model and lack of the real situation animation. In the future the road situation can be added so many other features, which can make the situation more like the reality. This example showed is just way to simple. And in the reality the situation is just more and more complex. The final stage of this ongoing research effort involves integrating the multiagent traffic control system with dynamic route guidance, also based on reinforcement learning. This is seen as a two-way interaction. Collective perceptions of the Q-learning agents concerning the distribution of congestion across the network could be used as a basis for advising drivers of lesscongested routes using variable-message signs, local-area radio broadcasts, or other means.

References:

[1] Abdulhai, B., Pringle, R., Karakoulas, G.J. 2003, ‘Reinforcement learning for true adaptive traffic signal control’, Journal of Transportation Engineering, vol. 129, no. 3, pp. 278-285.

[2] Mannion, P., Duggan, J., Howley, E. 2015a, ‘An experimental review of reinforcement learning algorithms for adaptive traffic control’, Autonomic Road Transport Support Systems, Autonomic Systems, Birkhauser/Springer.

[3] Rummery, G.A., Niranjan, M. 1994, ‘Online Q-learning using connectionist systems’, Cambridge University Engineering Department.

[4]Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., Bakker, B. 2005, ‘Reinforcement learning of traffic light controllers adapting to traffic congestion’, BNAIC, pp. 216-223.