Reinforcement Learning for Intelligent Traffic Light Control

COMP9417 Machine Learning Project

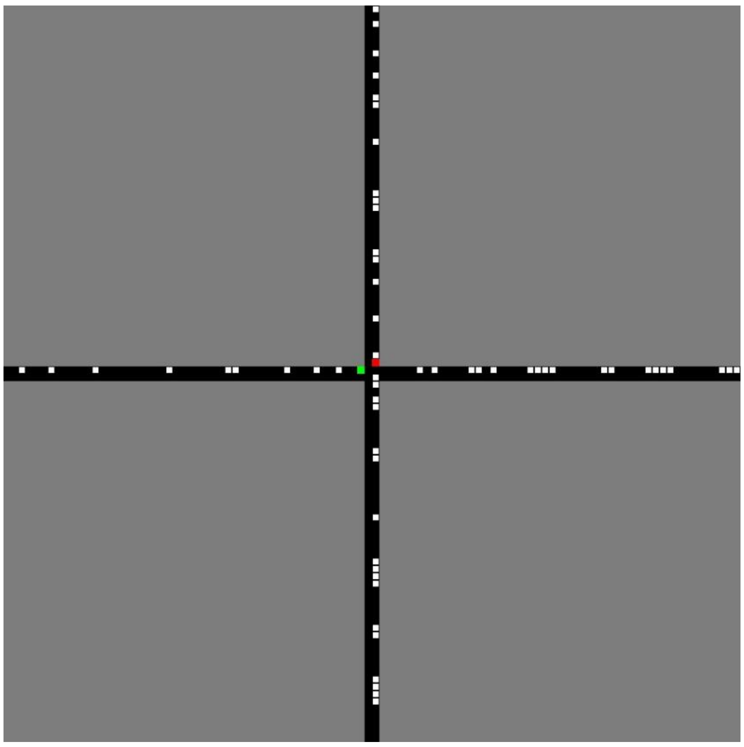
Bingxin Tong z5093617

Yizhe Xing z5104614

Yi Huo z5094643

Introduction

Traffic congestion is a serious and worsening national problem in recent years. Urban areas, which carry nearly 30% of all traffic in urban areas, are particularly affected by increasing urban congestion (Lindley, Jeffrey A 1987). One approach to solve such a tricky issue is to take priority to deal with the traffic signal controllers, which may result in an efficient traffic control. To be specific, the algorithm or the program of traffic signal controllers is to gain a better understanding of adjusting the current and up-to-date traffic lights based on real-time traffic situation. Q-learning is a simple way for agents to learn how to act optimally in controlled Markovian domains (Watkins, 1989). In addition, it could converge the optimum action values so long as all actions are repeatedly sampled in all states and the action-values are represented discretely. Furthermore, the effect values of reward which are earlier and higher than previous could be recorded so that the algorithm could determine the best strategies of following actions. In this report, Q-learning with epsilon greedy exploration is implemented and applied into reinforcement learning related to traffic signal controllers, aiming for learning to switch a set of traffic lights to reduce the delay at lights.



Related Work

In recent years, many intelligent transportation systems, traffic lights controller, has applied Q-learning algorithm which is a strategy implemented for reinforcement learning into real-time and adaptive control of transportation processes. A study conducted by Wiering claimed to use direct reinforcement learning to learn traffic lights controllers for a simulated transportation process issue which consists of a network of 2 x 3 traffic lights controllers (Wiering, M. A. 2000). Some of the experiments including a fix controller which iterates over all traffic nodes decision and a random controller for each traffic node are executed. Based on that, the speed and queueing of vehicles are also modelled for studying. In addition, adding more cars and saturation behaviour are also taken into consideration and analysis.

Many other recent studies have challenged multi-agent to create a classical non-stationary environment (Abdoos, Monireh, Nasser Mozayani, and Ana LC Bazzan 2011). The efficiency decreases with the increase in the number of agents was taken priority to deal with. The method of studying multi-agent is not complete new. Some other study also used multi-agent Q-learning for a predator-prey problem (Wiering, M. A. 2004). Moreover, there is a study choose to use MAQ-L for network routing (Oliveira, Denise 2006). Camponogara, Eduardo, and Werner Kraus also studied true adaptive traffic signal control (Camponogara, Eduardo, and Werner Kraus 2003), while Kuyer, Lior used coordination graphs to improved multi-agent reinforcement learning (Kuyer, Lior 2008).

Implementation

The aim of this project is to learn to switch a set of traffic lights to reduce the delay at lights. The traffic control simulation algorithm used in this project is self-coded using Python 3.5 and the simulation implementation is achieved by OpenCV 3.4. There are two source codes, one is for simulation algorithm, and the other is for test and evaluation work. There are two models in the simulation code, one is for fix model, which can control the traffic lights using a fixed change time of 10 time-steps, another is for Q-learning model, which can control the traffic lights using a reinforcement learning algorithm (Q-learning) to improve the performance of fix model.

Simulation Environment:

Both fix model and Q-learning model run in the same environment. There are only two intersecting roads in this environment and both are controlled by a set of traffic lights. Cars can only move towards one direction on the roads and cannot turn a corner. Cars can flow when the respective light signal is green, and stopped when the light signal is red. Because cars cannot occupy the same space, other cars should queue behind stopped cars.

We assume that the size of simulation region is 100 units * 100 units, where 1 unit equal to 10 pixels in the result video of simulation. Therefore, the road length is 100 units and one unit of the road holds one car. The speed of a car is one unit per time-step. The whole simulation takes 1000 time-steps in total. In addition, the cars should be removed from the road when they reach the boundary of the simulated region.

Moreover, there is a limitation for the traffic controller, which is the change time. A controller can change the lights, but only after waiting at least three time-steps since the last change.

Furthermore, we assume that cars enter the road some way from the intersection at random intervals, the function to generate new car is given by:

$$f(t) = True\ if\ t\ mod\ [randint(1,10)] == 0$$

Where t is the time-step, and a new car is generated when $f(t)\ is\ True.$

Simulation Algorithm:

For simulation code, it should be divided into two main parts. One is for fix model, another is for Q-learning model.

Fix Model:

For fix model, what we need to do is to control the lights using a fixed change time of 10 time-steps. The most important thing is to organize the whole simulation process logically. The specific assumptions of this simulation have been mentioned in the section of 'Simulation Environment'. In addition, we make one more assumption to avoid car crash in our simulation: if a car sees the traffic light become green and attempts to move forward, but it sees another car move across the intersection in front of it, the car will choose to stop and wait for a time-step.

The fix model algorithm goes as follows:

```
1. While time-step < 1000:
        (i) Change lights after 10 time-steps:
                If time now is the time to change light, then switch.
                Else remain the lights.
        (ii) Move cars:
                If a car is at the intersection:
                        If the car sees a green light:
                                If no car in front of it, then move forward one unit.
                        If the car sees a red light, then stop.
                Else:
                        If no car in front of it, then move forward one unit.
        (iii) Generate new cars using random function
```

Q-learning Model:

Q-learning is a simple way for agents to learn how to act optimally in controlled Markovian domains (Watkins, 1989). In addition, it could converge the optimum action values so long as all actions are repeatedly sampled in all states and the action-values are represented discretely. Furthermore, the effect values of reward which are earlier and higher than previous could be recorded so that the algorithm could determine the best strategies of following actions.

The dynamic programming formula of Q-learning should be:

$$\mathrm{Q}(s_t, a_t) = (1 - \alpha) \times \mathrm{Q}(s_t, a_t) + \alpha \times \left(r_t + \gamma \times \max\mathrm{Q}(s_{t+1}, a)\right)$$

Where s_t is the current state and s_{t+1} is the next state after select an action a_t. $\mathrm{Q}(s_t, a_t)$ is the value of taking action a_t in the current state s_t, $\max\mathrm{Q}(s_{t+1}, a)$ is the maximum of the values of taking all possible actions a in the future state s_{t+1}. α is learning rate, γ is discount factor and r_t is the reward observed for the current state s_t.

In this project, we use Q-learning with epsilon greedy exploration to learn how to change traffic lights in order to improve the performance of fix model. First, we need to initialize all the parameters needed in this algorithm, such as: learning rate α, discount factor γ, epsilon for greedy exploration and episode, which is the iteration number. Then, we need to initialize Q-table with all values set to zero, where each item is represented by a state with an action. In addition, we need to initialise R-table, according to the rules that reward -1.0 if a car is stopped at a red light on either road, zero otherwise. Next, in each episode, first choose an initial state and then select an action according to epsilon greedy exploration. It is to encourage exploration, with a random number between 0 and 1 smaller than parameter epsilon to randomly choose an action, otherwise select the action with the highest corresponding Q value. If there are multiple actions with the highest Q value, then randomly choose one of them. After select an action, perform this action, observe the current reward and get the next state. Then, we should update Q-table by the formula above. Finally, update the current state as future state and repeat.

The Q-Learning algorithm goes as follows:

```
1. Initialize needed parameters, such as: $\gamma$, $\alpha$ , epsilon and episode number.
2. Initialize Q-table to all zero. Initialize R-table as rules: reward -1.0 if a car is stopped at a red light on either road, zero otherwise.
3. For each episode:
    (a) Select a random initial state.
    (b) While time-step < 1000:
        (i) Select action:
            If light delay < 3 time-step, then do not switch lights.
            If it is time to switch light in fix model, then switch lights.
            If random < epsilon, then random select an action.
            If random > epsilon, then select the action that has the maximum value in Q-table.
        (ii) Perform action and Measure reward
        (iii) Get next state
        (iv) Update Q-table
        (v) Set the next state as the current state
```

The state using in the Q-learning algorithm is:

- closest car position from intersection for road 1 (0-8, 9 if no cars)
- closest car position from intersection for road 2 (0-8, 9 if no cars)
- light setting (0 green, 1 red for one of the roads)
- light delay (0-3)

Simulation Implementation:

In order to display the simulation on the screen, we decided to use OpenCV to write images for each time step and then write all of the images into a video. There are two videos for simulation, one is for fix model and the other is for Q-learning model. The videos have been uploaded on YouTube, please go to https://youtu.be/KSVe5CjPbsk for fix model, and https://youtu.be/ezEEBDhFZmY for Q-learning model.

Test and Evaluation Work:

Another code is for test and evaluation, the specific usage is described in README file.

Results

The performance measure of this project is the sum of the number of time-steps the number of cars is queued in both roads in 1000 time-steps.

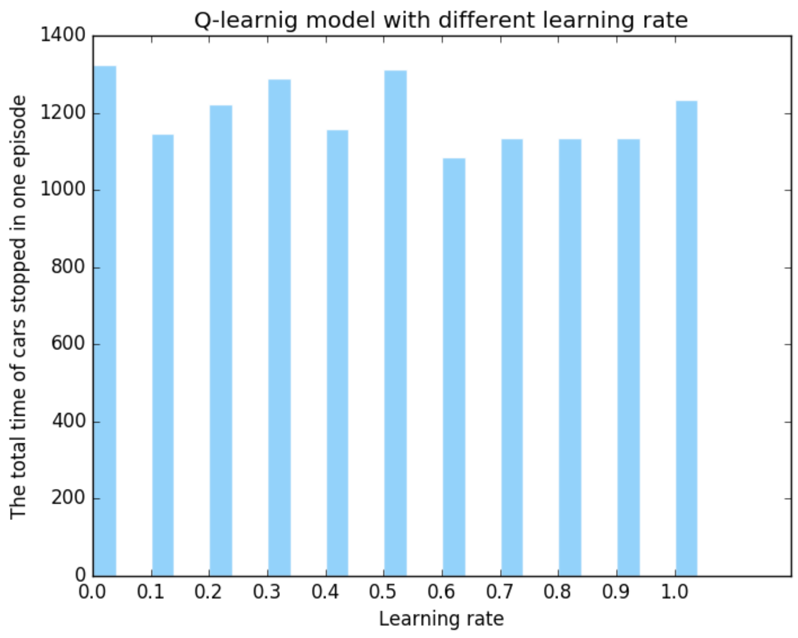
The most important thing is to find out the optimal parameters for Q-learning model first, and then use the best set of parameters to compare with the fix model.

In addition, in order to compare fix model and Q-learning model correctly, we generate cars in both models with same random seed, which can make the total number of cars generated in a whole 1000 time-steps the same. Therefore, we can compare the number of time-steps the cars stopped in a whole 1000 time-steps between two models to conclude which one has a better performance.

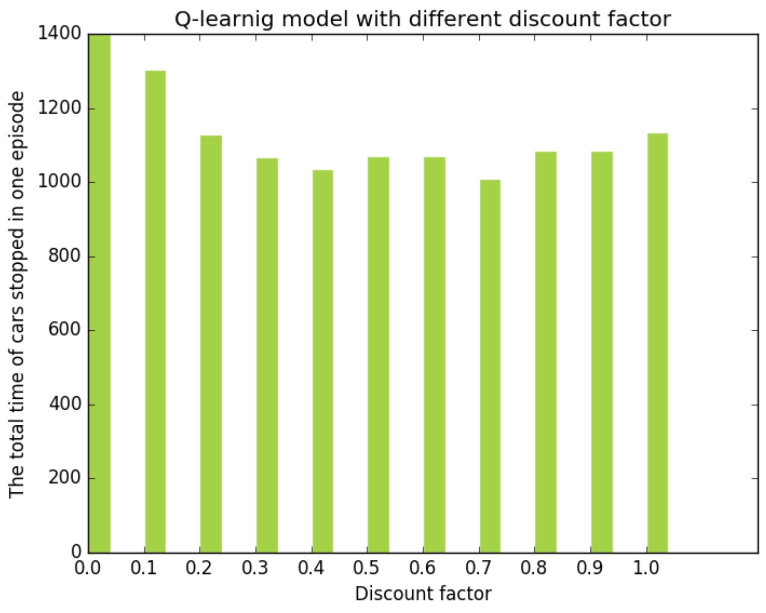
All the result graphs below are generated with seed 33 and episode 50.

Q-learning model with different learning rate:

The learning rate alpha determines to what extent newly acquired information overrides old information. A factor of 0 makes the agent learn nothing, while a factor of 1 makes the agent consider only the most recent information. According to the graph below, it is obvious that a learning rate of 0.6 gives the best result.

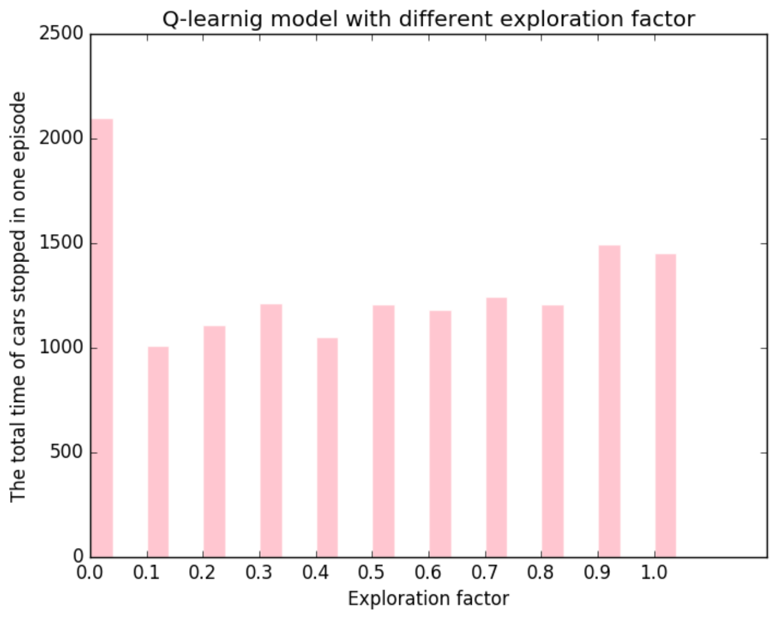


Q-learning model with different discount factor:



The discount factor gamma parameter has a range of 0 to 1. If Gamma is closer to zero, the agent will tend to consider only immediate rewards. If Gamma is closer to one, the agent will consider future rewards with greater weight, willing to delay the reward. According to the graph above, the discount factor which leads to the best performance is 0.7.

Q-learning model with different exploration factor:

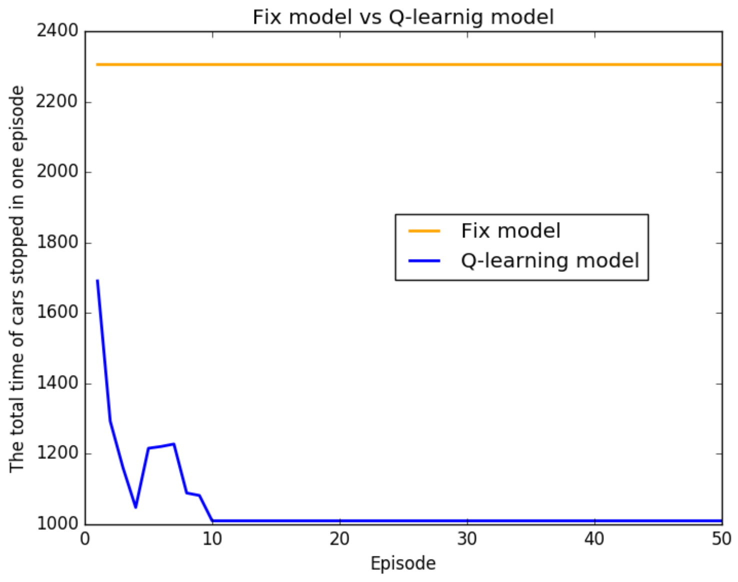


Epsilon greedy policy is a way of selecting random actions with uniform distribution from a set of available actions. If the exploration factor is 0, then there is no exploration conducted, and agent will repeat a path which is previously taken even if the path is not part of the optimal route. As shown by the graph above, an exploration factor of 0 cannot lead to optimal result. It is obvious that 0.1 is the best value for epsilon.

Fix model vs Q-learning model:

The figure below shows a comparison between these two models in a set of optimal parameters which are chosen from the three steps above. The seed is 33, γ is 0.7, α is 0.6, epsilon is 0.1 and episode number is 50.

As is shown in the figure below, we can conclude that Q-learning model become gradual convergence in 10 times of iteration and get a stable value which is around 1000. The result of Q-learning is almost a half of the result of fix model. It is clear that the Q-learning model is performed better than fix model.



Conclusion

This project aims to improve the performance of a fix model which switches the traffic lights in a fixed time by using reinforcement learning algorithm. The performance is measured by is the sum of the number of time-steps the number of cars is queued in both roads in 1000 time-steps. By doing many experiments, we could find out a set of optimal parameters for Q-learning model. After comparing the fix model which switches the traffic light every 10 time-steps with the Q-learning model which is using the optimal parameters, we can conclude that Q-learning model shows a potential to solve traffic problems and decrease the delay of waiting for red light. However, there are some limitations in this model. For example, the performance is related to the parameters choose and we didn't consider about other situations in real world. In a word, this Q-learning model only works for a specific environment like the one we describe before and there is still a long way to go.

Future Work

The strategies of reinforcement learning could be implemented into many areas related to the simulation of traffic transportation. However, the design of the traffic lights controller we made is based on some certain constrains, for example, the speed of cars, the distance between two cars, etc. In addition, the effect of pedestrians or the weather condition may result in different decisions that the traffic lights controllers could made. In the future, the different speed for each car and the diverse distance between different cars should take priority to consider into the design of the programming. Furthermore, the traffic congestion and traffic accident should be modelled for the real traffic transportation. Since the traffic accident occur by coincident, the traffic controller should also learn the strategies about how to deal with such a situation in real life so that it could make the correct and appropriate decision.

In conclusion, testing the systems on more realistic traffic simulations are necessary. The systems should be improved in terms of two ways traffic road, different reinforcement learning, distance between different cars, extra lines or extra intersections.

Reference

[1] Abdoos, M., Mozayani, N. and Bazzan, A.L., 2011, October. Traffic light control in non-stationary environments based on multi agent Q-learning. In Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on (pp. 1580-1585). IEEE.

[2] Wiering, M.A., 2000. Multi-agent reinforcement learning for traffic light control. In Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000) (pp. 1151-1158).

[3] Wiering, M.A., Veenen, J.V., Vreeken, J. and Koopman, A., 2004. Intelligent traffic light control.

[4] Arel, I., Liu, C., Urbanik, T. and Kohls, A.G., 2010. Reinforcement learning-based multi-agent system for network traffic signal control. IET Intelligent Transport Systems, 4(2), pp.128-135.

[5] de Oliveira, D., Bazzan, A.L., da Silva, B.C., Basso, E.W., Nunes, L., Rossetti, R., de Oliveira, E., da Silva, R. and Lamb, L., 2006, December. Reinforcement Learning based Control of Traffic Lights in Non-stationary Environments: A Case Study in a Microscopic Simulator. In EUMAS.

[6] Camponogara, E. and Kraus, W., 2003, December. Distributed learning agents in urban traffic control. In Portuguese Conference on Artificial Intelligence (pp. 324-335). Springer, Berlin, Heidelberg.

[7] Lindley, J.A., 1987. Urban freeway congestion: quantification of the problem and effectiveness of potential solutions. ITE journal, 57(1), pp.27-32.

[8] Watkins, C.J. and Dayan, P., 1992. Q-learning. Machine learning, 8(3-4), pp.279-292.