Learning Efficient Traffic Light Switching

Karl Krauth z3416790,
David McKinnon, z3421068
COMP9417 Assignment 2

Introduction

Switching traffic lights at an intersection can be a large factor in the amount of traffic build-up at the lights. Keeping certain lights red and green for the right time intervals can reduce the amount of traffic, and create a more efficient road system.
The goal of this project was to implement a reinforcement learner for traffic light switching on the intersection of two one-way roads, to choose when to change lights to green or red for the greatest throughput of cats in both directions.
The project was also extended to implement that same learner on the intersection of two-way roads, and also to test both under different distributions of cars. See [3] for the exact assignment specification.

Assumptions of the system

This simulation does not take into account braking and acceleration, and assumes that all cars move at the same speed uniformly. It is also assumed that both roads in the simulation have a similar usage (except for a couple of experiments).

Implementation

A road simulation system was created in Python, which consisted of the following components:

- Two-way roads: they spawn cars at random intervals at each end and remove them once the cars have reached the other end. The roads also stop cars from progressing when the light is red.
- An intersection: ensures traffic lights are switched such that no crashes can occur. Since the intersection is two cars wide we ensure that both lights are red for two timesteps before another light turns green. We also have a minimum switching time of 3 seconds.
- A distribution module: contains various random functions used for car generation (See Table 1 below for the distributions).
- A learning module: contains the Q-learning algorithm to determine whether or not to switch lights. It is described in detail below.
- A view module: Provides a visualization of the current state of the road.

A snapshot of the simulation can be seen in Appendix A, Figure 1.

Learning Algorithm:

We implemented Q-learning which can be found within the learner.py file.
Our states consist of:

- The distance of the closest car for each road that has not passed the intersection yet, if the closest car is further than 8 spots from the intersection then we simply set the distance to 9.

- The state of each traffic light (red or green).
- The time before the traffic light can get switched again.

Possible actions simply include whether we want to switch the traffic lights or not.
We store our estimated Q values in a dictionary of arrays where the keys are the possible states and the array indexes are each possible action for each state.
We also include a reward function as defined in the spec: -1 if a car is waiting at any traffic light, or 0 otherwise.
The learner can improve its estimates for Q values based on an old state, a new state and the action that was taken. The learner can also recommend an action to take based on its current Q estimates. It simply recommends the action that would lead to the state with the highest Q value.

Experiments and Results
The performance measure is the sum of the number of time-steps the number of
cars is queued in a period of time. Our performance measure is slightly different from the one given in the assignment spec, rather than counting the number of continuous cars at a stopped sign we count the number of cars that can not move forward in a time step.
Therefore, a lower number means a better system.
The experiment was to run the simulation for a time, and record the performance measure every 1000 time steps, then take the average over that time. This experiment was performed for the baseline and the reinforcement learner, using different probability distributions to model car creation.
The car creation was controlled by sampling from the following distributions:

- Poisson distribution.
- Gaussian distribution with mean 10 and variance 1.
- Uniform distribution with probability 0.1.
- And the standard probability distribution given in the assignment spec (this was denoted ‘Random’ in the tables below).

The table below shows the average performance over a given number of measurements for each experiment.
Table 1: The average performance for the learner and baseline, on default parameters, over a series of distributions on a signle direction road.

Performance	Random	Poisson	Normal	Uniform
Baseline number of measurements	3483	2820	3199	3329
Baseline performance	845	1495	722	670
Learned number of measurements	2917	2430	2695	2754

Leaned performance	151	440	127	112

It was noted that a Poisson distribution of cars gave a higher density of cars overall, and so another simulation was run: one road was Poisson-distributed, and the other Random, to simulate a minor road intersecting with a major road.
The results of this were that across more than 9000 measurements, the baseline had an average of 1184, and the learner had an average of 253.

These experiments were then repeated on the same intersection with two-way roads. The results are in Table 2.

Table 2: The average performance for the learner and baseline, on default parameters, over a series of distributions, on two-way roads

Performance	Random	Poisson	Normal	Uniform
Baseline number of measurements	2136	1794	2012	2093
Baseline performance	2060	4301	2047	1903
Learned number of measurements	2233	1851	2075	2097
Learned performance	1235	8986	1261	1120

Under the major-minor road scenario, with one road Poisson-distributed (mean 2), the other random, the average performance was 3245 after 2408 measurements for the learner, and 3180 after 2443 measurements for the baseline. The difference in performance here is not statistically significant.

Varying Parameters:
All the above experiments were performed using a discount parameter of 0.9 and a learning rate inversely proportional to the number of times a specific state with a given action had been visited. we experimented with various values for our discount parameter which lead to comparable or worse performance. We also experimented with various learning rates which led to worse performance.

Discussion
As can be seen from the results in Table 1, using a reinforcement learner to control the switching of traffic lights on an intersection of two one-way streets drastically improves performance and reduces the number of cars that are queued over a time period. This leads to less road congestion and more efficient road systems.

Similar results occurred for a two-way road system, but the ratio between the two performances is smaller, with the learner only performing about twice as well as the baseline. This is still a vast improvement though.
However, one thing worth noting is the Poisson-distributed cars on a two-way road system - the learner performed *half* as well as the baseline. This is probably due to the enormous density of cars on *both* roads, and the learner is likely giving priority to one road at the expense of the other, whereas the baseline is fair. Both still have large amounts of traffic, but the wait time under the baseline is constant.
Overall, the learner performs far better than the baseline, when the roads are not very congested. When the traffic density becomes too high, as with the Poisson distribution, the learner performs similarly to if not worse than the baseline.

Related work

Other research done on this problem has used Q-learning and Neural networks [2], or throughput balancing, simulated annealing, and Markov Decision Processes [1].
These were evaluated on a more complex simulation that included extra features like major and minor roads, and multiple intersections. On metrics similar to the one used here, these latter three techniques proved to be little better than their baseline. However, Q-learning and Neural networks proved successful in learning the patterns in traffic to switch lights more efficiently than alternative methods.
Due to the fact that these papers used different metrics, although they are somewhat similar, it is hard to compare performance between the results presented here and the results from these papers.

Conclusions

From the results it can be concluded that reinforcement learning can be applied to the problem of traffic light switching to reduce traffic congestion and improve road efficiency, except in the case of extreme congestion. Given that this occurs often in large cities, potential applications could be lights between major and minor roads, in cities at non-peak hour times, and smaller suburban roads.
The last experiment in results, comparing major and minor roads, shows that despite the limited assumptions, the learning algorithm still performs far better on a more realistic one-way road situation than the baseline. For the same major-minor road situation on a two-way road, the learner does not prove to be significantly better than the baseline, probably due to the high volume of traffic. This would clog the road, regardless of any greater efficiency of switching.

References:

http://cs229.stanford.edu/proj2005/RobinsonMosherEgner-LearningTrafficLightControlPolicies.pdf
http://dro.deakin.edu.au/view/DU:30058834
http://www.cse.unsw.edu.au/~mike/comp9417/ass2/data/traffic.txt

Appendix A

The following graphs are, in order, Figures 1 - 6. On each of the graphs in Figures 2-6, the x axis is the number of measurements of performance, and the y-axis is the performance number, explained above.

Figure 1: A snapshot of the two-way road intersection simulation

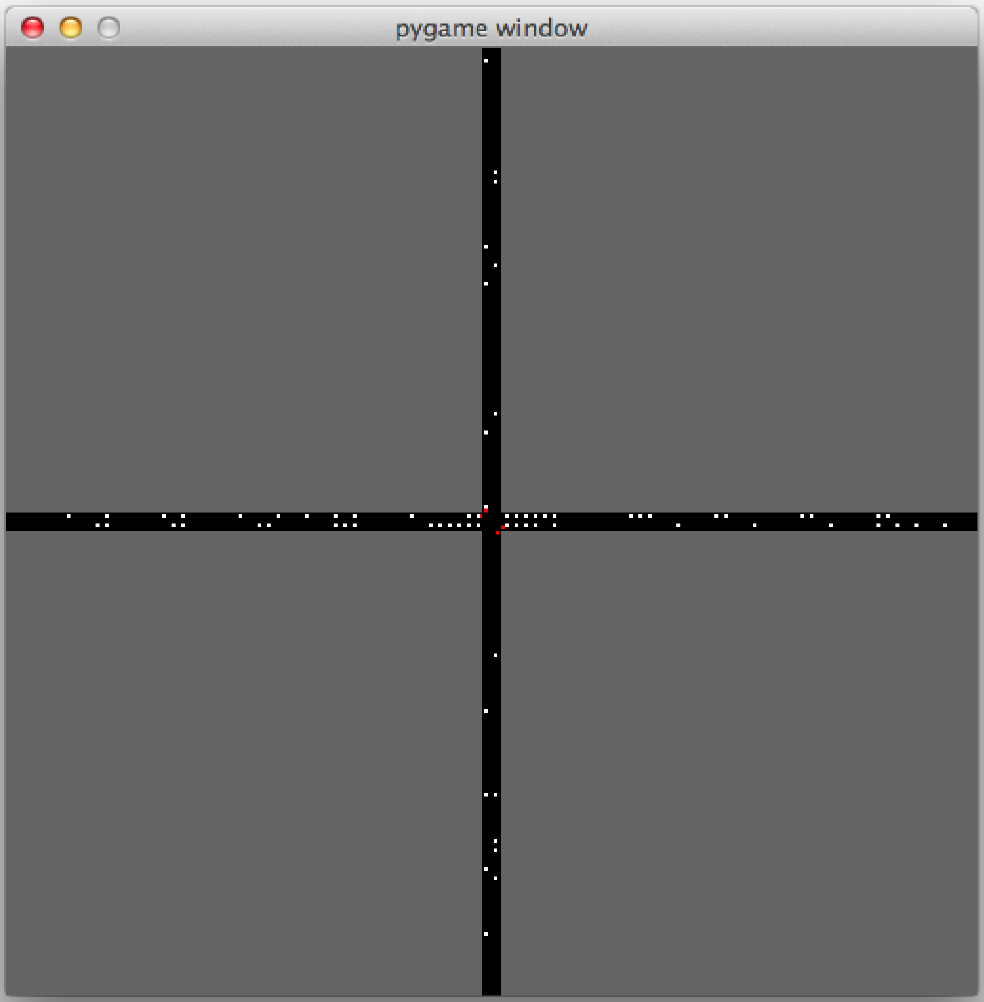

Figure 2: Performance over time graphs for random (standard - the left) and uniform (the right) distributions, for baseline

(top row) and learned (bottom row), for one-way streets.

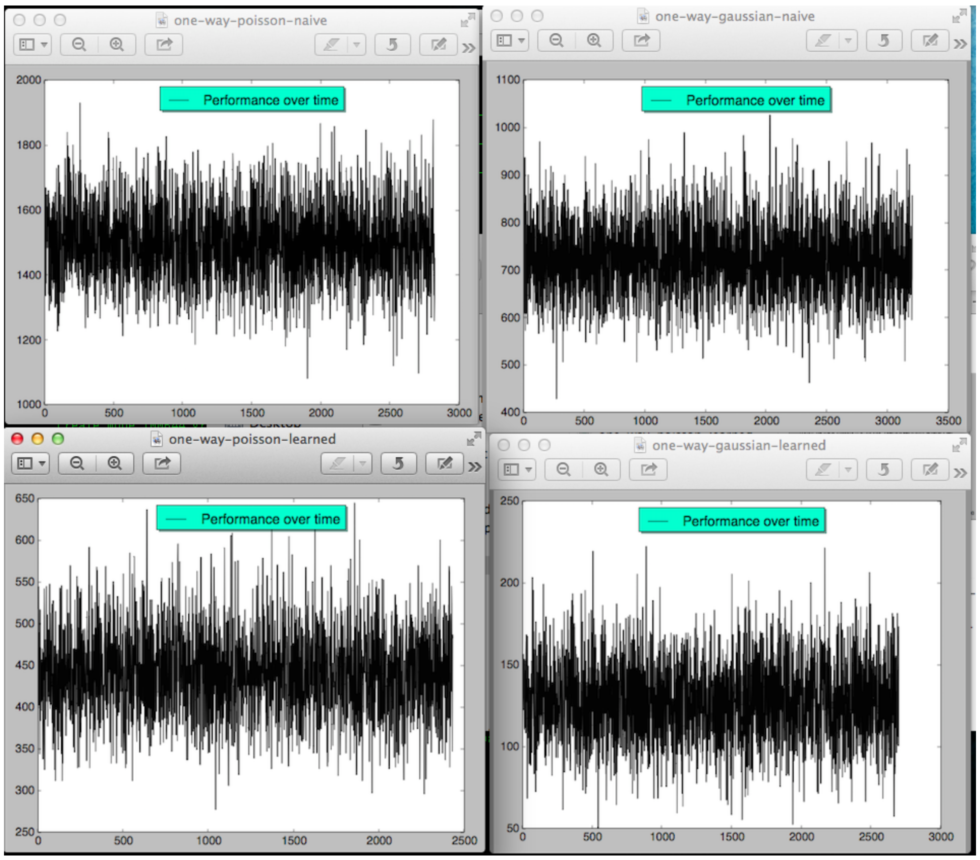

Figure 3: Performance over time graphs for gaussian (right) and poisson (left) distributions, for baseline (top row) and learned (bottom row), for one-way streets.

Figure 4: Performance over time graphs for uniform (right) and random (left) distributions, for baseline (top row) and learned (bottom row), for two-way streets.

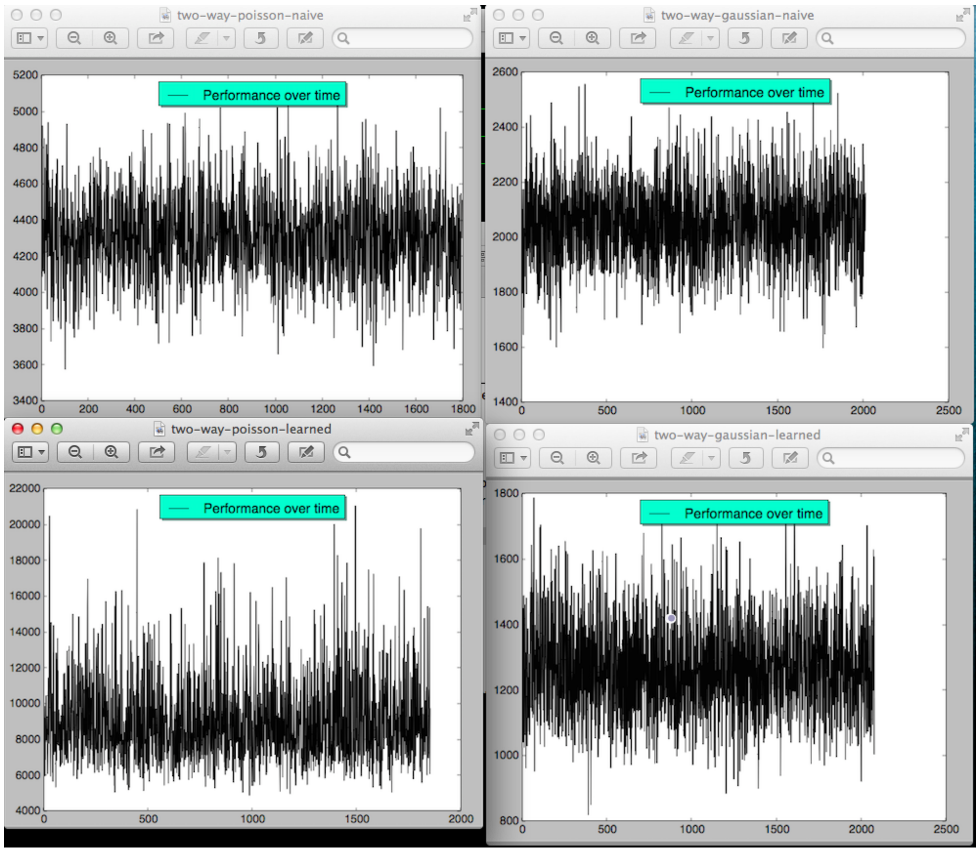

Figure 5: Performance over time graphs for gaussian (right) and poisson (left) distributions, for baseline (top row) and learned (bottom row), for two-way streets.

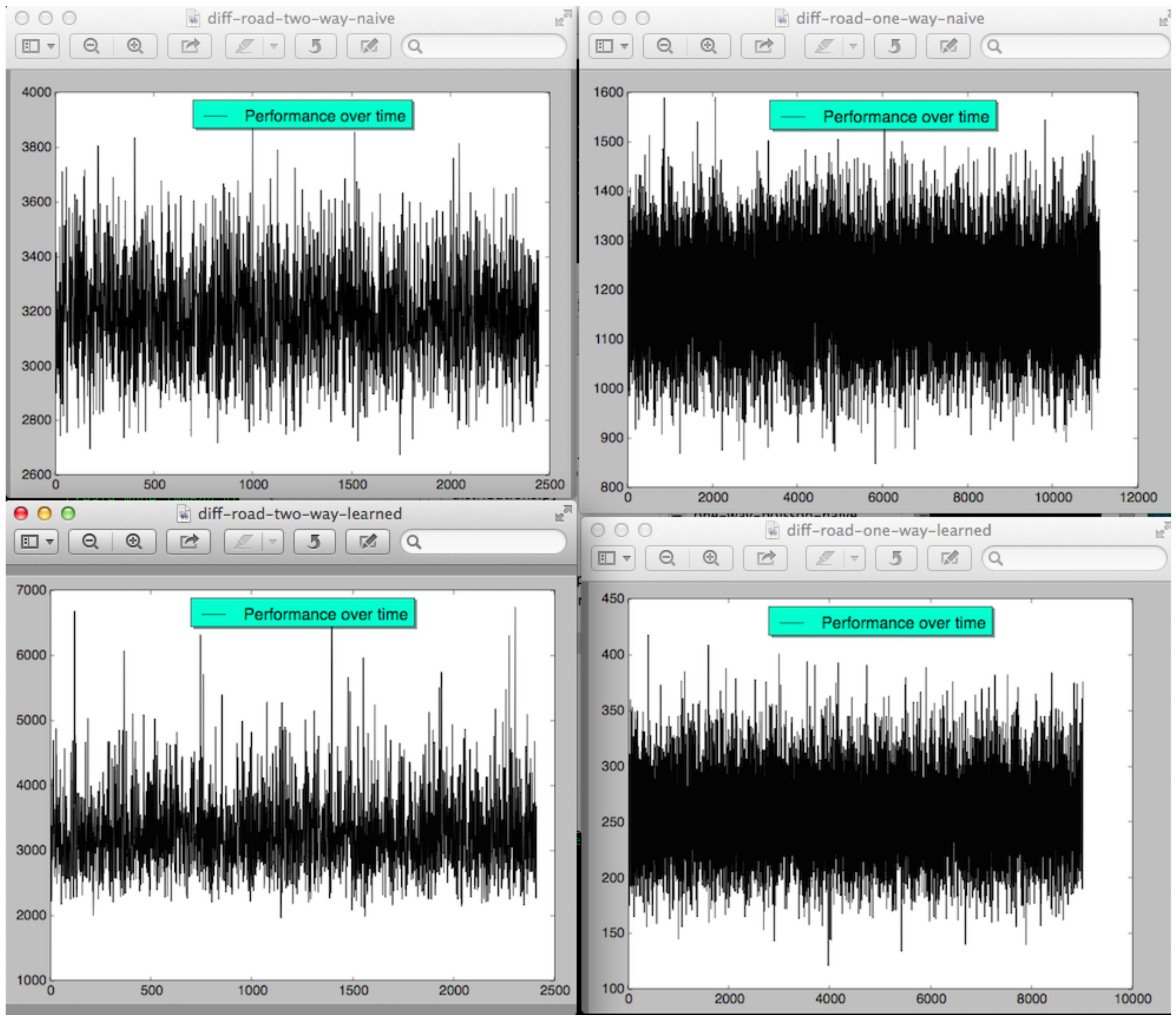

Figure 6: Performance over time graphs for one road random, the other poisson, for baseline (top row) and learned (bottom row), for one-way roads (right) and two-way roads (left).