Reinforcement Learning in Controlling Urban Traffic

Miles Kovach and Zihan Shang

Background:

Cars spend unneeded time and gasoline idling at stop lights in city traffic. According to one study done by edf.org, an individual car wastes anywhere from 0.2 to 0.7 gallons of gasoline when idling.

Furthermore, traffic is prone to changing in nontrivial ways. If there is a big event in a city, for example, and traffic needs to be rerouted, each intersection must change to adapt to its neighbors, which in turn adapt to their neighbors, and so on.

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward. And is one of three basic machine learning paradigms, alongside supervised learning and unsupervised learning. This paper seeks to explore the use of machine learning algorithms, specifically *Reinforcement Learning*, as a possibility for optimizing the process of automating self-adapting traffic lights.

Tools:

The premise of reinforcement learning (RL) is a simple one: RL represents one type of machine learning algorithm that learns in a way similar to humans and other animals. That is, initially beginning with a random (or already in-place) strategy and receiving a reward or punishment which the algorithm then uses to adjust its next action. In a simple RL algorithm, an *agent*, ie an implementation of the algorithm, will have a *state* to refer to its environmental conditions, and a palette of *actions* from which it can choose to respond to the state. Using an action to accomplish a desired goal rewards the agent with feedback. This will be in the form of a reward, such as an increase in a score variable, or more rarely a punishment, which is a decrease in the same. We feel that this is an effective method for observing and controlling traffic signals because of its ability to be trained over a range of both simulated and real situations; its gamelike nature would allow it to potentially be adapted from one situation to the next with minimal retraining.

Stated explicitly,

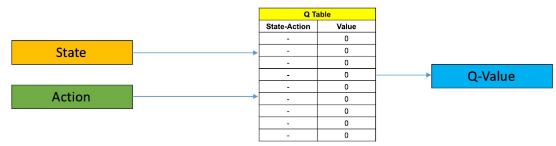
$$Q_{new}(s_t, a_t) = Q_{old}(s_t, a_t) + \alpha * (r_t + \gamma * Q_{max}(s_{t+1}, a) - Q_{old}(s_t, a_t))$$

Where $Q_{new}(s_t, a_t)$ represents the Altered Q value taken for a given state and action s_t, a_t Each such Q makes up a cell on the Q table and reflects, roughly, the probability that choosing that Q (by taking action *a* in state *s* at time *t* will lead to a reward; $Q_{old}(s_t, a_t)$ is the initial value that corresponded to that pair; α represents the learning rate, a tunable parameter common to many types of reinforcement learning algorithms. r_t is the reward received for that step if the corresponding action is taken, γ represents a “discount value” which prevents nondeterministic and unsuspected penalties in an otherwise successful algorithm from hindering the algorithm’s conversion. $Q_{max}(s_{t+1}, a)$ represents the maximum Q value that could be achieved for all given possible next actions on the table.

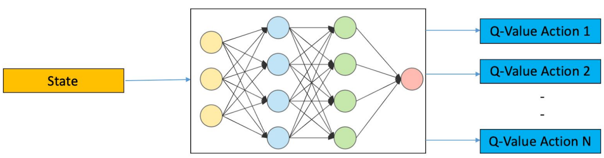
This is the basis for a simple Q-learning algorithm, which can be successfully employed in a simple situation like a game or even a single intersection. The parameters which constitute state for such an algorithm are number of cars at an intersection, location of each car, and whether they're seeking to turn.

To create a Q table for a whole network, such as a city, would be impractical. *Askwonder* reports that the ten most populated cities in the US have an average of just over 3500 signalled intersections apiece.[1] Even if we greatly oversimplified our assumptions about traffic lights such that the lights could only be red or green and opposing lights were always different colors, our action space of "toggle" or "remain the same" would still be mapped over millions of states to represent the parameters mentioned earlier for all the 3500 intersections in a given city. As a result, we looked into a few different machine learning techniques based on the Q-learning mechanics but more suited to deal with larger sets of environmental parameters.

Deep Q learning does just this: rather than a colossal table with millions of entries, the agent is driven by a neural network. Neural networks are an appropriate choice for this task, being adaptable to large input spaces by design. In this paper, we adapted the code laid out by Andrea Vidali - University of Milano-Bicocca, along with SUMo, the Simulation of Urban Mobility.



Q Learning



Deep Q Learning

SUMo is a simulation tool which

[1]https://askwonder.com/research/total-number-traffic-light-intersections-us-ouykt0v63#:~:text=Total%20Number%20of%20Traffic%20Light%20Intersections%2C%20US%20Major%20Cities,cities%20in%20the%20United%20States.

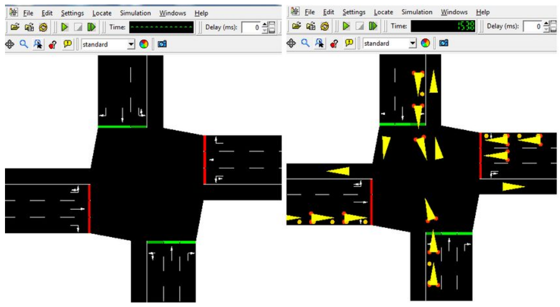
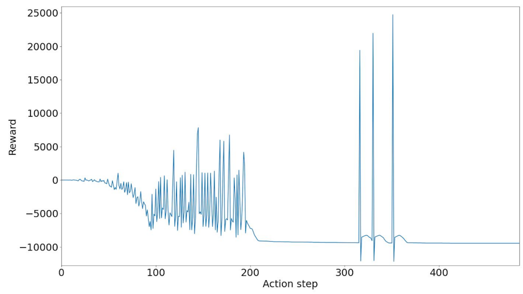
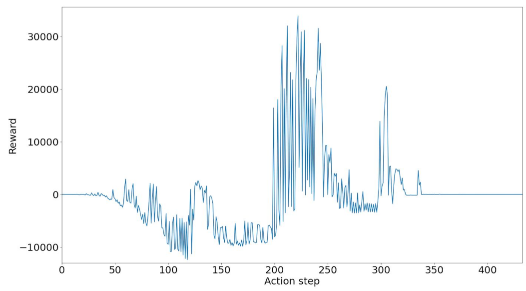


Image credit:
https://www.researchgate.net/figure/Simulator-of-Urban-Mobility_fig3_316564054

Results:

We use SUMO to train and test our model, the parameters for our training is 100 episodes each run and 5400 times of action in total. For the test, we first use 1000 cars to train and test, then use 2000 cars to test our algorithm and if not work, we will train the model again with 2000 cars and so on. After changing the number of cars in total from 1000 to 5000, the reward does converge to a smaller value. Which can be seen as below.





The time taken for each episode is around 80 seconds with CUDA toolkit (CPU: AMD 3800X Memory: 32G GPU: RTX 2070SUPER).

Future Work:

In the future, we see great potential for the application of this technique on a much larger scale. Our initial implementation would involve comparison of deep-Q with other machine learning algorithms, observing for both training and inference efficiency.

Next, we would like to see how effectively this kind of machine learning can be effectively scaled to large sets of intersections and cities. It should also be considered how the size of the road nets that are used affects the performance of the algorithm.

Another goal is to assess a given deep-Q algorithm's cross-applicability; can an algorithm trained on one set of intersections be successfully applied to a different set with minimal retraining? Are there peculiarities that can occur with specific intersections or traffic flow patterns which will cause critical impairments to either training or implementing the algorithm.

Our final and most important goal is to test the RL's ability to self-train on the fly, as the original premise is to minimize the amount of time spent by all cars on the roads within the bounds of the simulation, in this future case, a city. If an unexpected change in traffic flow happens, RL has the advantage over other forms of machine learning in that it learns in real-time rather than relying on a static training set; if it was found that the algorithm performed correctly by adapting quickly to unexpected conditions, it will successfully demonstrate this stated purpose and is conceivably ready for trial implementations in real intersections.

The algorithm we choose for our training, which is reinforcement learning now could also be changed in the future with a better algorithm found for unsupervised learning. Also great potential for the efficiency of our algorithm can be developed to make it run faster since the performance of the current model is not enough to fit in the real life conditions. We could change the number of parameters in our algorithm to make it finally work for complicated real time situations.

In conclusion, we would like to thank *Andrea Vidali* of the University of Milano-Bicocca, whose work was essential in forming the basis of our study.

References:

1. https://en.wikipedia.org/wiki/Reinforcement_learning
2. https://askwonder.com/research/total-number-traffic-light-intersections-us-ouykt0v63#:~:text=Total%20Number%20of%20Traffic%20Light%20Intersections%2C%20US%20Major%20Cities,cities%20in%20the%20United%20States.
3. https://github.com/AndreaVidali/Deep-QLearning-Agent-for-Traffic-Signal-Control