Reinforcement Learning for Intelligent Traffic Network Control

Matt Stevens, Alex Tamkin, Christopher Yeh – Stanford University

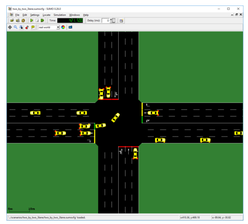
CS 229 (Machine Learning) Final Project, Spring 2016

Motivation

In 2014, traffic congestion caused American to waste about 7 billion hours and 3 billion gallons of fuel, for a total cost of $160 billion. This makes up roughly 2% of all of the gasoline consumed in the U.S. We use reinforcement learning to learn the optimal traffic light policy to minimize the average time that cars spend idling in front of traffic lights as well as their CO_2 emissions.

Data and Simulation

SUMO (Simulator of Urban Mobility)



Our simulations are based off of SUMO, a versatile traffic simulator. It runs a per-vehicle simulation, in which the user can configure road layouts, traffic light patterns, and vehicle flows. We control the traffic lights and extract detailed information about the system state using a python interface called TraCI.

Features

At each time step, we collect the following features for each traffic light and neighboring traffic lights, aggregated over the past 5 time steps

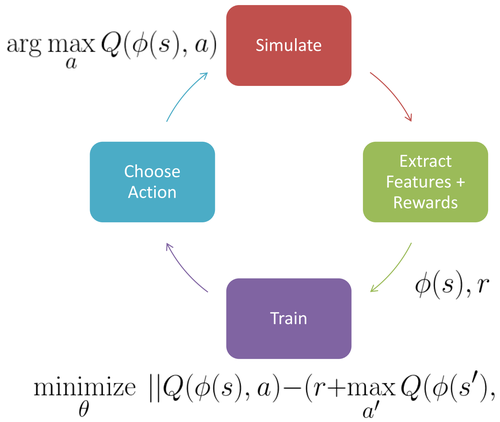
- Sensor (Induction Loop) in each lane
  - Number of cars
  - Speed of each car
- Action taken (direction of traffic light chosen to be green)

Reward Function

We maximize the throughput of cars by using the distance traveled by cars within the intersection as a reward function because the two are proportional.

Technique

Q-Learning



$$\underset{\theta}{\text{minimize}} \; ||Q(\phi(s), a) - (r + \max_{a'} Q(\phi(s'), a')||$$

Q-Function

Linear Regression

- Mini-batch SGD
- L2 loss function
- L2 regularization
- Simple to train

Neural Network (in progress)

- Simple MLP model
- Nonlinear feature mappings
- High danger of overfitting

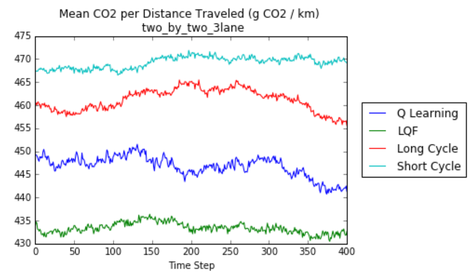
Our regression functions are implemented using scikit-learn

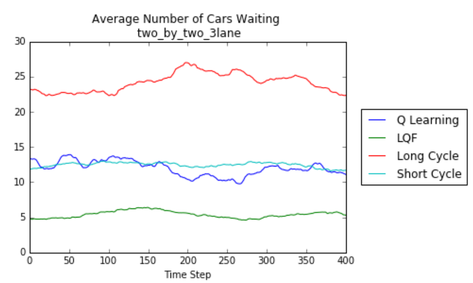
Training

We used random search for hyper-parameter optimization to explore search spaces too large for grid search. Since usually only a small number of hyper-parameters are relevant, most of the detail in grid search is unnecessary.

Due to the recursivity of the Q function, regularization is important to prevent exploding gradients. We used regularization ($\lambda \approx 1$) and a low learning rate ($\alpha = 10^{-5}$) to achieve convergence.

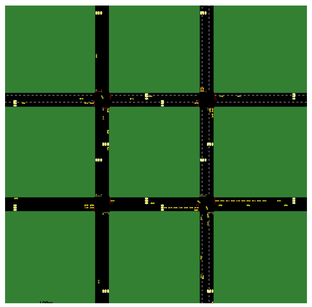
Experimental Results





We compare our results with Q-learning to multiple baselines: a long cycle of 45 seconds for each green light, a short cycle of 15 seconds per green light, and a heuristic algorithm called Longest Queue First (LQF). LQF measures the number of cars in each lane and turns lights green for the lanes that contain the largest number of cars.

Our algorithm outperformed the cycle algorithms but did worse than the LQF algorithm. It is worth noting that LQF has more information available to make decisions, making it a simpler problem to solve.



Our experiments were run on the 2x2 network shown above, where each road had 3 lanes. Cars were routed evenly between every entrance and exit in the network. Sensors were placed in every lane.