UNIVERSITY OF CAMBRIDGE
Department of Computer Science and Technology

Machine Learning and the Physical World (L48)
Course Project Report

Gaussian Processes for Traffic Optimisation

Authors: Albert Kutsyy, Charlie Hennys, Josh Cowan
Code Repository: https://github.com/akutsyy/traffic-optimization

Chapter 1: The Problem

Section 1.1: Background

Modern traffic light controllers allow for on-the-fly adjustments to light timing and duration in order to facilitate enhanced traffic control. While some cities such as Toronto attempt to centrally control the traffic lights, they utilise simple algorithms that just synchronise nearby lights to allow for long stretches of green lights [3].

With current traffic detection equipment, it is possible to estimate traffic across an entire city, and dynamically adjust traffic light timing in order to decrease commute times. While this may seem minor, it would have a dramatic impact on the lives of millions - Americans spend around 5 hours a week commuting to and from work [1]. Further, cars are a major polluter even when idling, so reducing overall traffic has a significant impact on greenhouse gas emissions.

Section 1.2: Solution

Determining the control pattern of traffic lights to optimise traffic on a city-wide level is an enormous task. There are far too many emergent effects for a human to be able to do more than a basic job, and it is impossible to collect enough data in the real world to determine how the millions of possible combinations will affect traffic. Thus, it is necessary to use a computer to compute and apply a good traffic light combination based on observed conditions. Simulating enough data points to do this optimisation on the fly is incredibly computationally intense, and pre-simulating enough points is similarly infeasible.

Thus, we have developed an emulator that trains on a traffic simulator and can deliver estimates of how different traffic control decisions will affect traffic in a fraction of the time. Then, using Bayesian optimisation, we can determine the optimal traffic light configuration given arbitrary traffic condition measurements.

Section 1.3: Simulation

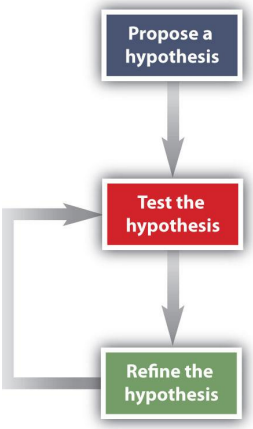
There are a number of traffic simulators that have been developed to estimate traffic conditions, with approaches ranging from simulating individual driver decisions to treating traffic as a fluid dynamics problem. In order to control traffic on a per-traffic light basis, it is most useful to use a high-fidelity simulation. Simulator of Urban MObility (SUMO) is an open-source traffic simulator developed at the German Aerospace Center that simulates traffic on a per-car level [2]. It can also model a mix of vehicle types, including trucks and bikes. This high-fidelity behaviour simulation allows us to account for emergent behaviour in road networks. However, it also means that it’s too slow to use for real-time traffic control on a large scale, even with high power computing capabilities.

Section 1.4: Emulator

We developed an emulator for this traffic simulation, which takes a number of traffic and traffic-control parameters (see Chapter 4: Development) and estimates the mean traffic speed. This is a surrogate model for the simulator, and we perform bayesian optimisation in order to optimise the mean traffic speed. We used mean speed because this is a good proxy for throughput, and is simpler to compute. By pre-training this emulator using SUMO, we can now optimise traffic control in real-time, reducing commute times and greenhouse gas emissions.

Chapter 2: The Model

Section 2.1: The Scientific Method



In order to optimise traffic across a city, we need a surrogate model. We can create this model using the scientific principle. The model contains an hypothesis of how input parameters (including traffic conditions) affect the speed of traffic. We can then conduct an experiment, where we run the simulator with a series of parameters and record the output. Finally, we use this evidence to update our hypothesis. This principle is illustrated right.

Our hypothesis space is a Gaussian Process. We inject our knowledge of traffic dynamics (that they are quite smooth with regard to traffic density and light parameters) into the RBF kernel.

If we were to run experiments in a naive fashion, such as a grid over all possible options, we would dramatically over-sample areas of low variation and under-sample areas of high variability. Instead, we seek to minimise our lack of knowledge of the world. We achieve this by sampling at the x value with the highest variance in the GP. This is effective because it means we are constantly refining our knowledge of the regions we know the least about, such as non-smooth regions or simply those where we haven't sampled much information.

We then add this experiment's result into the GP, thus refining our hypothesis as in the scientific process.

Section 2.2: Using the Simulator

Practically, we conduct each experiment as follows. First, we use EmuKit's experimental design loop to select a batch of 10 experiments to run. Then, for each set of parameters, we generate a SUMO config file, car route file, and road network file. In parallel, we have one instance of the SUMO simulator on each of these configurations. We then parse the output file, trim away the start and end of the simulation (where traffic is building up or dispersing),

and take the average speed of all cars in all time steps. This datapoint is fed back into the GP.

Section 2.3: The Result

After creating and refining our hypothesis, we are left with a gaussian process that, for each set of traffic and traffic control parameters, has a hypothesis for the average speed of traffic. It additionally has an estimate of the variance for each of these points, allowing us to not only make guesses, but measure our knowledge. Critically, getting a specific hypothesis from this Gaussian Process is very quick, allowing us to efficiently search for a minimum.

Chapter 3: Bayesian Optimisation

The parameters for the simulator/emulator can be split into three groups. The first group are the observable parameters. These are conditions that can be observed on the road but that cannot be directly changed such as the proportion of vehicles that are cars. The second group are the unobservable parameters. These are also conditions on the road that cannot be directly controlled but they also cannot be observed/measured in a realistic way, for example driver politeness. The final group are the controllable parameters. This group only consists of the traffic light ratios that we wish to control.

Parameter	Type
Traffic light ratios (4 parameters)	Controllable
Driver competence	Unobserved
Space drivers leave between cars	Unobserved
Ratios of trucks, cars and bikes (3 parameters)	Observed
Probability of car going from X to Y (12 parameters)	Observed

The ratio parameters are given to the simulator unnormalised which means that the acquisition function can just pick any combination of values. This does, however, mean that there are multiple sets of parameters that have the same output since, for example, 1:1:1 and 2:2:2 are the same ratio. This, however, didn't cause any problems.

In use, the observed parameters are measured using sensors at the junction and the unobserved parameters have some fixed value for that particular junction that can be determined through experimentation or through more 'hands on' measurement. The Bayesian optimisation then aims to determine the set of controllable parameters that maximise the emulator function for the given observed and unobserved parameters. This is

quick enough that the returned value can then be used on the traffic lights. The process can be ran, eg. every minute, to update the ratios between the lights in real time.

The BO process starts by creating a new GP with only four dimensions which aims to model the situation given by the observed and unobserved values. We are trying to model a function of the four traffic light ratios for the rest of the parameters fixed and, rather than trying to model the whole function, we are only interested in the maximum as these are the values that should be used for the traffic lights. The acquisition function, rather than just choosing the point with the greatest variance, chooses the point with the greatest expected improvement of the maximum. This will likely be either an area near to the current best where some improvement could be expected or an area with high variance that has a chance of a large improvement.

The emulator is a GP but, throughout the BO process, we are only concerned with the mean function of the GP. This makes things more simple as we only need to look for the maximum of the function without worrying about the variance at that point. This could mean that we choose a value for the traffic lights which has high variance in the emulator and doesn't actually have good performance on the simulator. However, areas of high variance are unlikely to be maxima of the emulator's mean function since there is clearly little data in this area to suggest a maxima is there.

Since the function being called for each selected point is just the mean of another GP, the optimisation is performed quickly: even for many iterations. The function is fast enough that a naive approach to finding the maximum such as grid search could work, however, since this system is to be used in real time, the reasonable performance improvement offered by BO is valuable.

Chapter 4: Scaling

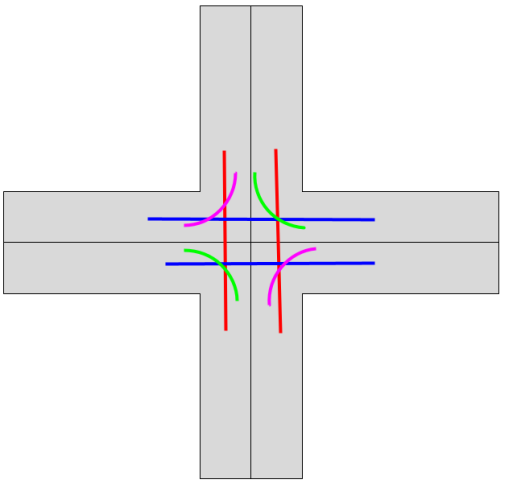
Whilst modelling individual junctions is useful for understanding traffic dynamics on the microscopic level, more complex road systems are impacted by phenomena occurring *across* junctions; that is, the impact of one junction's dynamics impacts more than just that junction. The interactions between these junctions are very complex. Each junction impacts the dynamics of its neighbours, which in turn affect the junction again. This causes emergent phenomena such as *grid-lock* to occur. In situations such as gridlock, it becomes difficult to establish which junction is responsible for the seizures of the other junctions. Gyratories are a prime culprit for both non-stationarity over a localised time-period.

It is for these reasons that city-level traffic modelling needs to happen on a higher level of abstraction than per-junction modelling. A lower-fidelity surrogate model is useful here to avoid the state-space explosion engendered by attempting to model an entire city's worth of traffic lights without a markov assumption. In this simulation, we sought to model some general parameters governing *actuated* signals. Actuated signals dynamically set traffic phase parameters based on the input of sensors measuring traffic flow on the immediate approaches to the junction.

This reduces the number of parameters, making it more feasible to model an entire city. We continue to run the microscopic model provided by SUMO, but use actuated signals. However, the computational cost of simulating an entire city does restrict how many times the simulator could be called, and as such each query must make the maximum possible exploration/exploitation of the state space, something more easily achieved within a smaller state space. These parameters used are thus generalised over the entire city, and include factors such as whether vehicles turning across traffic have a ‘protected’ turning phase and what the upper and lower bounds on actuation-informed periods are (e.g. how long should someone have to wait if the sensor is broken?).

Chapter 5: Development

In order to constrain the massive space of “optimising traffic”, we chose, early on, to focus on controlling traffic light patterns. This can be defined as determining how long the light stays green for each possible configuration of traffic (each colour in the diagram on the right).



We began development using the MovSim simulator (movsim.org). However, after a significant amount of time and effort setting up the simulator, we found that it actually mis-interpreted the OpenDRIVE standard in such a way that rendered it unusable. (It evidently doesn’t allow traffic from one road to directly merge into another without a highway-style ramp, making even simple junctions impossible. We thus switched to SUMO.

We used Emukit because it provides simple interfaces for both Bayesian optimisation and Experimental design. The GPBayesianOptimization class only requires the list of variables to optimise and the function on which they are being optimised. All of the GP is managed internally. The ExperimentalDesignLoop class still requires us to build and manage the GP ourselves. However, the batch_size parameter allows for multiple points to be returned by the acquisition function. This allowed us to have multiple simulations running in parallel which greatly improved performance.

One issue that we encountered was the size of the space; 21 dimensions is a lot and it requires a very large number of datapoints to sufficiently cover it. We reduced the size of the space by limiting the range of possible values for the various parameters.

Parameter	**Range**
Traffic light ratios (4 parameters)	0.1-0.5

Driver competence	0.5
Space drivers leave between cars	1
Ratios of trucks, cars and bikes (3 parameters)	1
Probability of car going from X to Y (12 parameters)	0.01-0.5

Several of the parameters are now fixed due to having only one possible value. This isn't ideal but allowed us to be able to reasonably train the emulator on our machines. On a more powerful machine or with more time, these constraints could be relaxed.

Chapter 6: Results and Conclusions

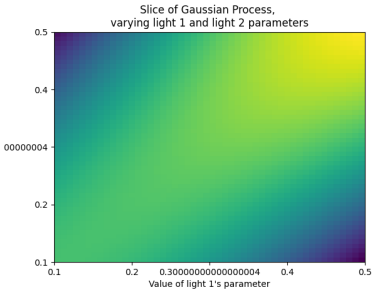
Section 6.1: Computation Time

We ran a number of experiments on a simple one-junction network in order to get data that is small enough to be human-interpretable. The junction is still complex enough to be interestingly computationally, as there are 12 different routes through it and 4 traffic light parameters (amber and amber-red light times are fixed proportional to the speed limit). This gives a total of 16 parameters for the GP to optimise. The relatively high dimensionality actually resulted in us needing to train the GP for a large number of experimental design iterations (400 executions of the simulator). The computational intensity of the traffic simulation meant that even utilising 100% of all cores on a high-end laptop, this process took over two hours.

In contrast, doing the bayesian optimisation to determine an optimal control configuration given the traffic took only a few seconds. This underscores the necessity of using an emulator such as ours to solve on-the-fly traffic control.

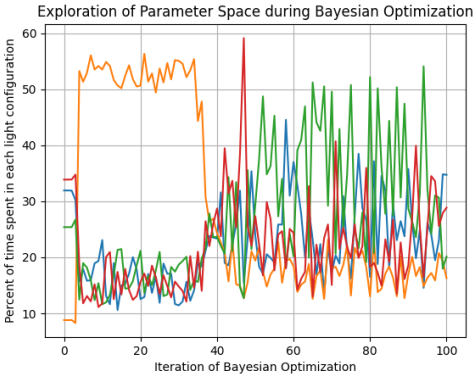
Section 6.2: Junction-level Results

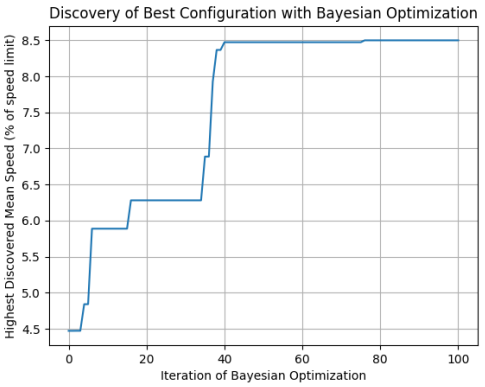
Our junction-level experiments show very good results. The Gaussian Process successfully models the interaction between traffic parameters and the control parameters. On the right is a diagram of the estimated traffic flow (shading) based on the timing of two of the lights (all other parameters are randomly set). This is essentially a 2D slice through the gaussian process, and accurately models the interactions.



The Bayesian optimisation successfully finds optimal solutions within seconds, which meets our goals of an emulator fast enough to use for real-time traffic control. Below left is a diagram of how the bayesian optimisation explores the search space

given a random set of traffic flow parameters. Each light pattern described in Chapter 5 is plotted as one colour. Below right is the mean traffic speed of the best configuration that bayesian optimisation has discovered at each iteration, which always converges within 100 iterations (taking a few seconds). Here, it discovers that a solution of 27.6%, 21.7%, 24.9%, and 25.8% of time in each pattern gives an optimal solution. Note that it continues to explore the search space even after finding a local minima at iteration 40. This is to find a better solution if one exists anywhere in the search space.





Section 6.3: Future Work

Future work largely will focus on the city-scale emulation. Due to computational limitations, we were unable to run the GP experimental design loop on an entire city, thus using a high-power computing facility to do so is a very attractive goal for future work.

Additionally, a possible avenue of exploration is to use real-world data and other simulators to do multi-fidelity analysis. This would allow us to reduce the number of times SUMO has to be called, dramatically speeding up the experimental design loop. Additionally, the use of real-world data as part of a multi-fidelity emulation would result in a higher accuracy model that can account for factors such as road rage and pedestrian traffic.

Finally, a very useful addition would be sensitivity analysis. This would mean that in addition to simply being able to find a good traffic control layout, engineers can observe which factors influence traffic. For example, if a particular intersection has a disproportionately large impact on the overall traffic, it may be a good idea to expand the road, add a bypass, or convert the junction into a roundabout.

Bibliography

1. “Census Bureau Estimates Show Average One-Way Travel Time to Work Rises.” *Census Bureau*, 18 March 2021, https://www.census.gov/newsroom/press-releases/2021/one-way-travel-time-to-work-rises.html. Accessed 17 January 2022.

2. Lopez, Pablo Alvarez, et al. “Microscopic Traffic Simulation using SUMO.” *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2018. *https://elib.dlr.de/124092/*, IEEE, https://elib.dlr.de/124092/.

3. “Signal Optimization (Coordination) Program – City of Toronto.” *City of Toronto*, 2020, https://www.toronto.ca/services-payments/streets-parking-transportation/traffic-management/traffic-signals-street-signs/signal-optimization-coordination-program/. Accessed 17 January 2022.