Training Intelligent Stoplights

Thomas Davids, Michael Celentano, and Luke Knepper

December 14, 2012

1 Introduction

Traffic is a huge problem for the American economy. In 2010, the average American commuter spent 34 hours stuck in traffic[1]. Some of this is simply due to too many cars on the road; however, much of this delay is caused by inefficient stoplights. Imagine if stoplights were intelligent; if they could look at incoming traffic in every direction, and make the best decision in order to minimize the delay for all drivers. Our project attempts to do exactly that.

To model the road, we use the Intelligent Driver Model. First developed in 2000 by Treiber, Hennecke and Helbing[2], it models the behavior of individual cars in traffic with the formula

$$\frac{dv_\alpha}{dt} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^4 - \left(\frac{s^\star(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right)$$

where $s^\star(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}$. We use the parameters $v_0 = 30$, $T = 1.5$, $a = 1.00$, $b = 3.00$, and $s_0 = 2$. Our traffic simulator models a grid of streets in a city, and generates cars with a fixed probability of appearing which behave according to this model. Our model has stoplights at each intersection to control traffic, and it is these stoplights to which we apply our learning algorithms.

2 Baseline

As a baseline algorithm, we used stoplights which waited 15 seconds before switching. They would then spend 5 seconds in the yellow light phase, leading to intervals with a period of 40 seconds. While obviously very rough, this algorithm provided an estimate of how a basic stoplight might behave, and a useful goal for us to try to beat. When running a simulation using these stoplights, the average speed for cars in our city was **10.9069 meters per second**.

[1]Schrank, Lomax, and Eisele, Texas Transportation Institute, *TTIs 2011 Urban Mobility Report*, 2012

[2]Treiber, Hennecke, and Helbing, *Congested Traffic States in Empirical Observations and Microscopic Simulations*, 2000.

3 Continuous State MDP

3.1 Algorithm

Our first approach involved implementing a continuous state MDP. In our algorithm, each light learns and makes decisions independently of the others, with any correlation between their decisions arising only through their responding to the same road. For the sake of simplicity, each light will only consider information describing the two roads meeting at its location. In particular, the state space any given light considers is parameterized by the following variables:

- A 6-dimensional vector S, with s_1 the number of cars on the vertical street, s_2 the average of their instantaneous speeds, and s_3 the sum of their instantaneous speeds. Components s_4 through s_6 contain the same information but for the horizontal street.
- A variable $l \in 0, 1$ describing the state of the light. The possible states of the light on the horizontal/vertical street are red/green and green/red, corresponding to $l = 0$, 1 respectively.

The set of actions a light can take at any given point in time, described by the binary variable a, are to not change ($a = 0$), or to change ($a = 1$). We require that $a = 0$ for five simulation steps after the beginning of the state either being in state red/green or green/red to avoid potentially erratic behavior. This design decision was made preemptively, prior to any observation of such erratic behavior arising from our algorithm or indication that it would.

A discount factor $\lambda = .95$ was used.

The reward function $R(S)$ was given by the average speed of cars on the two roads meeting at a given light. More precisely, $R(S) = \frac{s_3+s_6}{s_1+s_4}$. Thus each light tries to maximize the average speed of lights on the two roads it sees, even though the ultimate goal is to maximize the average speed across all streets. If the streets are empty, the average speed is undefined, so we set $R = 50$. This is a high value because empty streets suggest efficient flow of traffic.

To approximate a state update function, we sought a deterministic model in which $S_{t+1} = A \cdot S_t + B \cdot a$ where A is a 6×6 matrix and B a 6×1 matrix. Because we expect the update rule to depend upon the state of the light l, we would like to include this information in our model. But because l acts as an index of light states, it is unlikely that the dependence of our model on l is linear or can be expected to follow a simple mathematical relation that could be reliably applied to several scenarios. Thus, we seek not one update matrix A and one matrix B, but 2 such matrices $A_1, \text{and} A_2$ and $B_1, \text{and} B_2$. We will apply each matrix depending on what our light-state is at any given moment.

We find $\{A_l\}$ and $\{B_l\}$ by running linear regression on training data containing states in which both the light switched and the light did not switch. When the light does not switch, the training examples are of the form $\{S_t \longrightarrow S_{t+1}\}$. When the light does switch, the training data is of the form $\{S_t \longrightarrow S_{t+5}\}$. We attempt to predict the state five simulation steps in the future because a switch of the lightfrom red/green to green/red or vice versa- takes five simulation steps, during which one of the lights is yellow. We do this because a light will decide to switch because of the benefits in being in the opposite state (red/green or green/red), not to be in the state red/yellow or yellow/red which it must endure to get there. Attempting to predict each of the five intermediate states could compound the inherent inaccuracies in our predictions, preventing the foresight required to realize the potential future benefits of a switch. We decided use this approach after an initial attempt to predict even the red/yellow and yellow/red states led to lights which never decided to switch.

Finally, we sought a value function of the form $V(S) = \theta^T \cdot S$ for θ a 6×1 vector. As with the update matrices A_l and B_l, we find two vectors θ_1, θ_2 corresponding the the different values of l. Thus, $V(S_t) = \theta_l^T \cdot S_t$ where the light-state of S_t is l. These θ's were found using fitted value iteration, treating $l = 0$, 1 separately.

3.2 Results

The implemented algorithm resulted in enormous car build-ups because the decision to switch was rarely made. Our initial attempt to address this was discussed briefly above and involved predicting the state five simulation steps after a switch was made instead of the state on the following step. This change increase the propensity of lights to switch. We identified two further problems that may be preventing our algorithm from working.

- The failure of lights to decide to switch may result from their inability to notice the benefits of switching, potentially because they cannot accurately predict the state of the road after a switch. This may result from biases in our training data. In particular, time-steps in which a switch is made are far less common than time steps in which a switch was not made, even in an optimally functioning traffic-light system. Thus, our linear regression to find A_l and B_l will value accuracy on predicting the outcomes of not switching far more than accuracy on predicting the outcome of switching.
- Because continuous MDPs cannot be guaranteed to converge in general, it is possible that the parameterization of the state space and value function we chose did not allow our algorithm to converge.

4 Linear Regression

4.1 Algorithm

Our second approach involved implementing a linear regression algorithm. Our feature vectors $x^{(i)}$ consisted of seven elements: the number of cars approaching the light on each street, the average speed of those cars (normalized by the speed limit), and the average squared speed of those cars (normalized by the square of the speed limit), as well as a binary variable that was equal to one if the light decided to switch, and zero otherwise. A seven-dimensional weight vector θ was also stored, and the value of $\theta^T x^{(i)}$ represented the prediction of the average speed of the system after a given number of time steps. This value would give the decision: if the value was higher for the switched state than the constant state, the light would switch; otherwise, it would remain constant.

The weight vector was learned using stochastic gradient descent for linear regression: we used the formula

$$\theta := \theta + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x^{(i)}$$

with parameter $\alpha = 0.001$ and $h_\theta(x^{(i)}) = \theta^T x^{(i)}$. This updated value of θ was then used to make the decision for the next step of the light.

4.2 Results

After running our algorithm through $1,000,000$ time steps, we converged on a value of θ of approximately

$$\begin{pmatrix} 0.3820 \\ 0.0831 \\ 0.0388 \\ -0.0090 \\ -0.0133 \\ 0.0412 \\ 0.0805 \end{pmatrix}$$

where the elements represent, in order: the number of cars on the street which is currently green or will be green soonest, the normalized average squared speed on that street, the normalized average speed on that street, the normalized average speed on the perpendicular street, the normalized average squared speed on the perpendicular street, the number of cars on the perpendicular street, and the binary variable indicating whether a switch has just occurred.

While being run, this algorithm increase the total average travel speed (total distance traveled by cars in the system divided by total time traveled by cars in the system) by **1.9962 meters per second**. More impressively, however, when these weights were hard-coded and used to run make the decisions without learning, this algorithm increased total average travel speed by up to **7.8455 meters per second**.

5 Conclusion

While it was disappointing that our FVI algorithm failed to converge, the results of our linear regression are very encouraging. Saving over 7 meters per second on a baseline of 10 meters per second is a very significant gain. Furthermore, this algorithm, once learned, can be applied on a very basic level; a light simply needs to use the available information to make a decision, and no new data needs to be collected.

A couple of options are available for future steps. One interesting option would be to experiment with a different number of features; using a larger feature set would probably improve the model, but a smaller feature set might better approximate real-world situations, when a light might only know whether at least one car is waiting at the light.

Another option would be to use some method to discretize the data. As mentioned before, some of our problems with the FVI algorithm came from the fact that continuous FVIs are less reliable than discrete ones. If we were to use a clustering algorithm such as K-means to model the state space as finite, this would improve that method and hopefully open up a whole new set of options.

Overall, although we suffered some setbacks along the way, we are very optimistic about the results of this project. Although there remains a lot of work to be done before it is ready to be put into practice, we are hopeful that methods such as these could eventually cut into some of those 30 to 40 hours spent in traffic.