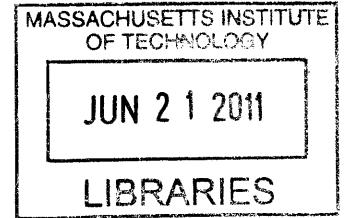


Traffic Delay Prediction From Historical Observations

ARCHIVES



by

Paresh Malalur

S.B. CS, M.I.T., 2010

Submitted to the Department of Electrical Engineering and Computer Science

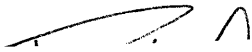
in partial fulfillment of the requirements for the degree of
Masters of Engineering in Computer Science and Engineering

at the

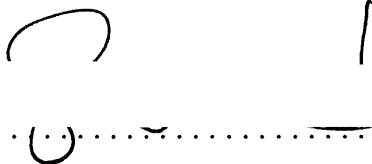
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

© Massachusetts Institute of Technology 2011. All rights reserved.


Author
Department of Electrical Engineering and Computer Science
May 18, 2011

Certified by
Hari Balakrishnan
Professor
Thesis Supervisor


Certified by
Samuel Madden
Associate Professor
Thesis Supervisor

Accepted by
Dr. Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

Road traffic congestion is one of the biggest frustrations for the daily commuter. By improving the currently available travel estimates, one can hope to save time, fuel and the environment by avoiding traffic jams. Before one can predict the best route for a user to take, one must first be able to accurately predict future travel times. In this thesis, we develop a classification-based technique to extract information from historical traffic data to help improve delay estimates for road segments.

Our techniques are able to reduce the traffic delay prediction error rate from over 20% to less than 10%. We were hence able to show that by using historical information, one can drastically increase the accuracy of traffic delay prediction. The algorithm is designed to enable delay prediction on a per-segment basis in order to enable the use of simple routing schemes to solve the bigger problem of predicting best future travel paths.

Acknowledgments

I would like to thank Arvind Thiagarajan for his guidance and insight in parts of this project. I would also like to thank him for providing the map-matched data of taxicab trajectories that is used for travel delay estimation.

Contents

1	Introduction	13
1.1	Traffic Congestion : A global epidemic	14
1.2	Finding an Immediate, Inexpensive solution	16
2	Background	19
2.1	Previous Work	19
2.1.1	Traffic Simulators	19
2.1.2	Hand-Picked Loops	20
2.1.3	Other Traffic Detectors	20
2.1.4	Jam-Bayes	21
2.2	Previous work done in the CarTel Project	22
2.2.1	CarTel Vehicular Testbed	23
2.2.2	Map Matching	23
2.2.3	CarTel Routing Engine	25
3	Design and Implementation	29
3.1	Design	29
3.1.1	Data Source	30
3.1.2	Traffic Influencing Factors	36
3.2	Data Pre-Processing	40
3.2.1	Map-matching	41

3.2.2	Finding Drives	41
3.2.3	Heuristic Outlier Removal	42
3.3	Algorithm	42
3.3.1	Traffic Model	43
3.3.2	Tree Classifiers	46
3.3.3	Prediction	52
4	Results	57
4.1	Methods and Results	57
4.1.1	Training and Test Data	57
4.1.2	Individual Road Segments	58
4.1.3	Trajectories	59
4.1.4	Visualization	63
4.2	Discussion and Future Work	64
4.2.1	Probabilistic Models	64
4.2.2	Grouping Segments	66
4.2.3	Using Bus data to predict car delays	67
4.2.4	Traffic Aware Routing	67
4.3	Conclusion	69
A	NWS Weather Code Table	71

List of Figures

1-1	A traffic jam in the Himalayas	14
1-2	A Taipei traffic jam comprising of primarily two wheeled	15
3-1	Flow chart of the traffic delay prediction algorithm	31
3-2	Prediction accuracy vs number of observations for individual road segments	32
3-3	A road segment on I-93 with some evident traffic influencing factors .	37
3-4	Classification of the I-93 road segment along the hours of the day . .	37
3-5	Classification of the I-93 road segment along the days of the week . .	38
3-6	Classification of the I-93 segment along the months of the year. Labels go from 0=January to 11 = December	39
3-7	Classification of the I-93 segment along for different weather codes . .	40
3-8	The probability densities for the I-93 segment. Note how the speed probability density appears to be a Gaussian mixture of two speed components	44
3-9	Probability distribution of a road segment on Rt 1 where a simple Gaussian mixture model assumption may not be valid	45
3-10	The road segment at the intersection of Austin St and New Rutherford Ave is very short and most of the data is available from interpolation. The data provided from such small segments typically prove to be very difficult to predict from	48

3-11	Some road segments like the intersection of Austin St and New Rutherford Ave have data that is not separable by a simple linear classifier .	48
3-12	The probability density for Main St in Boston. This segment has a traffic signal on one end. We can see how the distribution looks much more complicated than a simple mixture model	49
3-13	Some road segments like parts of Rt 2 near Athol have limited historical information	49
3-14	This section of Rt 2 in Athol is quite far away from central Boston and we have very limited data points for this road segment	50
3-15	A sample classification tree generated by our algorithm.	53
3-16	A sample classification tree generated by our algorithm	54
4-1	The distribution of the error rate for travel delays across 10,000 randomly chosen road segments shows that more than 50% of the time, travel delays are predicted with greater than 77% accuracy	59
4-2	The distribution of the error rate for travel delays across 1,000 randomly chosen taxi drives (> 10 min) shows that more than 50% of the time, travel delays are predicted with greater than 93% accuracy . . .	63
4-3	Green segments correspond to low error rates and red segments correspond to high error rates	64
4-4	Advanced statistics and information is available by just clicking on the desired segment	65

List of Tables

- 3.1 Data Sets 30

- 4.1 Prediction Errors on an individual road segment level 58
- 4.2 Prediction Errors on the first batch of data (September 2007 - October 2007) 60
- 4.3 Prediction Errors on the second batch of data (January 2009 - January 2010) 60
- 4.4 Prediction Errors on the third batch of data (February 2010 - July 2010) 60
- 4.5 Prediction Errors with delay observations from all three data sets . . . 62

Chapter 1

Introduction

As commuters, we have all experienced long traffic delays and have wished that we knew the shortcuts that the “locals” know which would help us avoid congested streets, traffic lights, stop signs and the like. Though we have access to road maps and the speed limits in each of these roads, this information is often insufficient for us to find the least frustrating route to our destination. We can experience drastic deviations from our estimated travel time due to factors like traffic light cycle lengths, rush-hour times, construction and special events like a Red Sox game at Fenway Park. In order to address this issue, it becomes critical that our routing software incorporate some of these factors during route planning. This chapter introduces some of the benefits of developing traffic-aware routing systems.

Chapter two describes existing techniques used to predict traffic delays and some of the merits and demerits of these methods. Additionally, I also overview some of the prior work done in the CarTel project [1] on which my work relies.

Chapter three describes the structure of the traffic prediction system including the prediction algorithm, routing system and visualization suite as well as the motivations for the various design choices.

Chapter four describes the performance of the traffic prediction system on various

data sets when compared to the previously used scheme. Possible future expansions to the project are also discussed.

1.1 Traffic Congestion : A global epidemic

Traffic congestion is one of the biggest time-sinks for the daily commuter. The average person is estimated to lose roughly 5 days a year to traffic jams [25]. With every major city facing this problem, the US experiences roughly 3.9 billion gallons in wasted fuel and \$115 billion in lost productivity, or about 0.8% of the nation's GDP [30][3] due to traffic. By finding efficient routes using a traffic-aware system that also reduces the idle engine time (time wasted in traffic jams, signals, stop signs, etc.), we can not only decrease the commute time, but also reduce the amount of fuel consumed, thereby reducing emissions and alleviating some of the pressure on our natural resources.

This problem of traffic congestion is not just restricted to the United States. It is a global phenomenon prevalent in almost every urbanized city - from cities far north like Moscow, Russia to Auckland, New Zealand in the Southern Hemisphere, from Tehran, Iran in the Eastern Hemisphere to Toronto, Canada in the West. Traffic jams have been frequently



Figure 1-1: A traffic jam in the Himalayas

observed in more low population areas that lie on major travel routes such as the high mountainous roads of Garhwal, India in the Himalayas (Figure 1-1) and the China National Highway 110 that connects the coal mines in Inner Mongolia to Beijing. Perhaps surprisingly, the United States doesn't hold the title for the most

congested city in the world. Sao Paulo, Brazil claims this title [11] and set this record on June 10, 2009 with more than 293 km (182 mi) of accumulated traffic out of 835 km (522 mi) monitored road segments. A solution to alleviate traffic conditions will hence be globally beneficial to people and the environment.

In many cases, restrictions are imposed to help reduce the number of cars on streets. Hong Kong has resorted to imposing a high vehicle purchase tax [16]. Bogota, Colombia, used the *Pico y placa* system [9], which restricts the use of vehicles for several days each week based on the last digits of license plates. The UK has introduced a Road Pricing system to encourage drivers to switch to public transport [13]. These systems of taxation and restriction have been unpopular amongst citizens. Alternative strategies could help reduce congestion while not imposing such constraints on the general public.

Since varying the hour of commute by introducing a non-standard work day is not economically feasible [12], people in many countries have tried alternate modes of transport like switching to two wheeled vehicles to better weave through traffic. However, cities like Taipei, Taiwan (Figure 1-2) and Ho Chi Minh City, Vietnam have daily traffic jams comprising of mainly motorcycles and other two wheeled vehicles. Other cities have tried introducing public transit which has helped alleviate traffic to some extent, but not solved problem.



Figure 1-2: A Taipei traffic jam comprising of primarily two wheeled

1.2 Finding an Immediate, Inexpensive solution

Rather than imposing restrictions or switching modes of transport, another option for dealing with traffic congestion is to find alternative routes that help users bypass backed-up road segments during periods of congestion. Some highways now have displays indicating congestion further down the road, thereby providing the commuter with some real-time information about traffic congestion levels.

In some other cases, the government makes the traffic levels that they monitor in some regions publicly available. For example, the Transport Department in Hong Kong has set up websites with maps showing congestion for the Cross Harbor Tunnel and provides commuters with signs on both sides of the harbor informing drivers of the quickest way to get to the other side of Victoria Harbor [10].

Unfortunately such systems are not common because they require the deployment and maintenance of sensors. Even when this information is available, drivers must go to the website and manually find a low traffic route to their destinations. Current traffic routing algorithms used by GPS navigation devices and Internet websites. often don't account for these traffic patterns and typically predict estimated travel times based on the speed limit of roads. Though the speed limit is a good indicator of the time taken to reach our destination under ideal conditions, it fails to provide a more efficient route during periods of congestion.

The difficulty of *predicting* traffic delays along different routes is a major deterrent for traffic-aware routing systems. In order to tackle the problem of predicting the most efficient possible route, one must first be able to predict traffic delays along any given path. Given this ability, one may then use a route planning algorithm to find the most optimal route. By making the reasonable assumption that traffic delays are additive over traffic segments, the problem reduces to predicting the traffic delay across any given road segment at any time.

The problem of determining the delay across a particular road segment is itself

not a trivial problem. Though one way to predict traffic delays along a road segment is to build an extensive physical model of the entire road system that factors in signal times, rush hour traffic, etc., such a system quickly becomes complex due to the large number of parameters involved. Further, obtaining the data to set each of the parameters (such as the cycle times of each traffic light and their relative offsets) can be near impossible. Additionally, such a system can not account for segments where vehicles show characteristic human behaviors — like traveling faster than the speed limit.

Another way to predict delays is to extract them from observed delays obtained from vehicles actually traversing the roads. This would help us frame the problem as an inference problem rather than a simulated physical model. Correspondingly, there are other obstacles to overcome in order to predict traffic trends from vehicular observations. Nodes that collect vehicle positions must first be built and installed which can be a slow and expensive process. GPS data is invariably noisy in downtown areas and this data is available for a limited number of road segments. Due to the unpredictability of traffic, estimating levels of traffic congestion is extremely difficult.

Chapter 2

Background

2.1 Previous Work

Many different approaches have been attempted to estimate and predict traffic delays in urban city streets and highways and each system has its merits and demerits.

The first hurdle that makes developing traffic estimation algorithms challenging is obtaining good data. Ideally, one would like fine-grained (to the level of seconds), low error and extensive data for each road segment for all times of day. Obtaining such data is difficult because typically vehicles are tracked with lower granularity (level of minutes), GPS data is noisy and only sparse data is available for inner streets and areas further away from the main city. However, many different mechanisms have been developed to circumvent these problems.

2.1.1 Traffic Simulators

Many algorithms have been developed using traffic simulators like [5][2][4] to obtain extensive amounts of fine-grained data for every road segment in a simulated city. This has led to the development of methods like Kernel regression [5] to capture traffic trends and queuing techniques [2] for modeling delays introduced by traffic

signals.

The advantage of using a simulator is the large amount of data that one can use to rigorously train and test the prediction scheme. It is possible to see how the system would behave in extreme scenarios that we may not be easily able observe in the real world and we have the ability to introduce hand crafted changes in the environment that would give us a strong indication of the flexibility of the prediction scheme.

On the other hand, while these algorithms perform very well on simulator data, they have been untested on real traffic data. Another common problem with such methods is that they rely on large amounts of data to learn the parameters of the model.

2.1.2 Hand-Picked Loops

In many cases, authors hand-pick a few road paths of interest and periodically drive their vehicles through these paths [8]. This allows the authors to collect data at the granularity they desire and error reduction techniques can be effectively applied to obtain accurate training data.

While it may be possible to increase the accuracy of predictions for these segments, it is not clear how well these methods generalize to other segments, especially since we may not be able to obtain such regular accurate data for these other segments. These methods do however, provide insight into some of the local factors that influence traffic.

2.1.3 Other Traffic Detectors

Obtaining GPS data typically requires installing units on vehicles or distributing a smart-phone application or buying this data from other vendors. These methods can be time-consuming, expensive and in the case of smart-phones a significant power drain [7]. To resolve these issues, many systems obtain traffic information from loop

sensors [8][20] installed on roads. These devices are buried under the road and use changes in electric and magnetic fields to “count” the number of cars that cross them. Such loop detectors are typically installed and maintained by the government.

Loop detectors serve as an accurate (3.5% error [32]) stationary source of traffic flux information for a given lane of a road. By placing two successive loop detectors, speed-traps can be created to estimate the speed of passing vehicles.

Unfortunately, loop detectors are expensive to install and prone to failure. Therefore, these detectors are only placed in areas of interest, resulting in data coverage that is not extensive. Since the sensors can only count cars, there is no information available about the behavior of the car, such as its acceleration, signal stops and turn information. Traffic estimation algorithms using loop detectors have also been developed but face the problem of information gathering.

The wide-spread use of traffic cameras has led to the development of schemes that use frames captured from these cameras to count vehicles and estimate their speed [32]. One scheme out-performed Gaussian-based machine learning scheme, but achieved a counting accuracy of only 72% which is not competitive with loop detectors (3.5% error in count [6]). This scheme faces the same problem as loop detectors; it is unable to effectively track the trajectory of a vehicle.

2.1.4 Jam-Bayes

The Jam-Bayes [19] system developed at Microsoft Research labs in for traffic prediction in Seattle has many common goals with our effort. The traffic prediction system at Microsoft incorporates not only monitored traffic from sensed traffic cells in the Seattle road network, but also uses traffic incident reports, major city events and weather information like precipitation, visibility, temperature and sunshine.

In their paper, the Jam-Bayes authors identified 22 key traffic “bottleneck” regions of the Seattle network. Once these regions were identified, the Jam-Bayes system then

learned a Bayesian network over all the variables (like events, time, weather) in order to obtain the best estimation model to determine the time taken for traffic jams to clear up in these bottleneck regions of Seattle.

The Jam-Bayes development team also created a website and smart-phone application to make their service more accessible for subscribers. This service provided users with estimated traffic conditions as well as a confidence measure of the estimation.

Our project draws inspiration from the Jam-Bayes project. We aimed to design a system that could incorporate multiple traffic influencing factors as well. We also wanted to have a system that provides users with feedback on the conditions. However, we wanted to do more than just identify bottlenecks and predict conditions on these bottlenecks.

Instead, our goal is to provide traffic predictions on an individual path basis in order to obtain an estimated travel time between any two points in our region of coverage. We also wanted to use this system to design a routing algorithm for vehicles that could take these traffic predictions into account and suggest the best route to take at some point in the future that would be able to avoid congested segments.

2.2 Previous work done in the CarTel Project

The CarTel system provides a traffic-aware routing engine for finding the best route for people to take to their destination. This system developed two key components that I utilize for the traffic prediction problem:

- GPS nodes on taxi-cabs (CarTel Vehicular Testbed)
- Map-matching and delay extraction
- A traffic-aware routing engine

2.2.1 CarTel Vehicular Testbed

The CarTel testbed [14] consists of nodes deployed on a few taxicabs in Boston. Each node includes a collection of hardware sensors placed on vehicles that constantly relay GPS position information to our servers and record the data for future use.

The hardware unit comprises of a GPS unit, a WiFi communication module and some on-board storage. GPS information obtained is first locally buffered on the on-board storage device, and then relayed via WiFi or cellular modems to our servers.

These hardware units were deployed on about 27 taxicabs in the Boston area as well as a few personal vehicles of lab members. Since taxicabs are not confined to bus-routes or highways, are operational for a significant portion of the day (and some times in the night) and since they face traffic that a daily commuter would face, we decided that taxicabs were the best set of vehicles to piggy-back our traffic collection devices on.

These devices record GPS information roughly once every second and hence provide us with fine-grained information about the position of the vehicle. This is our primary source of traffic information in this thesis. We hope to expand our coverage by deploying nodes on many more taxicabs in the future.

2.2.2 Map Matching

Once the vehicular testbed was set up to obtain GPS information from various taxicabs that housed the GPS recording units, we needed a way to convert these GPS points into routes taken by the cab drivers with the corresponding route delays.

CarTel’s vehicular tracking and map-matching system, VTrack [31], was developed as an efficient solution to this problem. The Map-matching algorithm uses the road adjacency constraints imposed by the road network and a simple physical model of vehicles to calculate the probability the sequence of GPS points corresponds to a particular sequence of road segments. The adjacency constraint ensures that the

trajectory is continuous, thereby ensuring that observation error in raw GPS points does not result in physically impossible paths of cars briefly jumping to other road segments. The simple physical model of a vehicle captures the fact that the velocity of a car changes smoothly. This allows us to model the true position of the taxicabs as a hidden state variable and ensures that we don't see erratic behavior like rapid acceleration and deceleration.

This modeling of the true position of the taxicabs enables us to infer the speed of the vehicle from the positions that we extract from the hidden state variable. Further, this process also allows us to interpolate the position of the vehicle over time to determine the travel time along really short road segments which otherwise would have been impossible. Further, this interpolation enables us to predict when a vehicle entered and exited a segment. It also enables us to infer the direction of travel on a road segment.

Another major advantage of interpolation is our ability to roughly infer the speed of the vehicle when no GPS observations are made. This occurs when a taxi travels through a tunnel or when the GPS satellite is temporarily lost due to environmental factors.

The Map-matching algorithm processes the sequence of GPS points using a Hidden Markov Model (HMM) with transitions specified by simple dynamic model in conjunction with the road network information of the United States to compute the most likely trajectory traversed by the cab. This system uses the Viterbi algorithm [15] under the hood and therefore provides accurate matching of the observed trajectory of GPS points to the true trajectory of road segments. uses the Viterbi algorithm [15] to find the hidden variables that maximize the likelihood of our observations under our constraints. The efficiency and error reducing properties of the Viterbi algorithm makes it a highly attractive algorithm for us to use for map matching.

This algorithm plays a key role in extracting travel delays across individual road

segments from taxicab trajectories, which serves as an input to the travel delay prediction algorithm developed in this thesis.

2.2.3 CarTel Routing Engine

Once the GPS points were map-matched to road segments, delay observations for road segments traversed by taxis became available. The next step was to determine the most efficient route from the source to the destination. However, since road traffic delays vary over time, an efficient routing algorithm that could adapt to changes in traffic delays was required.

The CarTel routing engine answers this problem with a three component routing pipeline :

- iCarTel portal
- Stochastic routing system
- Dynamic Route Planning via Continuous Queries

iCarTel Portal

The iCarTel portal [22] provides a web interface for users to obtain their optimal traffic-aware travel routes by utilizing the underlying routing machinery. Additionally, the iCarTel portal serves as a platform for the collection of GPS data from nodes using the CarTel vehicular testbed (Section 2.2.1) and the map-matching of this data into traversed trajectories.

Stochastic Routing System

The stochastic routing component [21] of the CarTel routing engine utilizes a stochastic model of travel delay on road networks in order to enable routing under uncertainty. Such a stochastic model allows the optimization of cost functions of the delay data

probability distribution, thereby enabling the CarTel routing algorithm to find routes that satisfy particular travel deadlines.

Dynamic Route Planning

The traffic-aware routing problem varies from the standard minimum cost routing problem in that the delays (cost) estimates along road segments constantly change over time. It would be impractical to recompute the optimal route every time the delay estimate across a road segment changes - especially when we are dealing with large road networks like that in Boston. Recomputing paths becomes even more infeasible in a practical setting with a large number of incoming routing queries.

The dynamic route planning algorithm [23] offers an approximate solution to this problem while still guaranteeing run-time efficiency. By using *K-Paths* and *proximity measures*, the algorithm speeds up the processing of the routes specified by incoming routing queries by computing likely good paths and by recomputing routes only when delay estimates have changed significantly.

Since this system delivers routes that are within 5% of the best shortest path under frequent delay updates while maintaining a run-time efficiency at an order of magnitude under the naive re-computation scheme, this algorithm made it possible for me to design a traffic prediction system that could interact with the routing system without incurring a heavy computational penalty.

In order to ensure that the delay updates given to this routing algorithm are valid updates, one must first understand the principles behind the backbone routing scheme of the dynamic route planning algorithm, the A-star search algorithm.

Predicted Delays in the A-star Routing Algorithm

The A-star algorithm [18], a popular search algorithm known for its performance and accuracy, lies at the core of the dynamic route planning algorithm. The A-star

algorithm plays a critical role in determining how traffic delay estimates effect the resulting minimum cost path from the routing algorithm.

The A-star algorithm performs a “best-first” search of trajectories from point A to point B by expanding along the node C that minimizes

$$f(A, B) = d(A, C) + h(C, B)$$

Where $d(x, y)$ is the distance function between points x and y and $h(x, y)$ is an admissible heuristic distance from x to the goal y .

In the setting of searching for the shortest travel time routes, the distance function $d(x, y)$ becomes the time taken to travel from x to y . The heuristic function $h(x, y)$ used was the as-the-crow-flies time, taking into account the curvature of the Earth using the Haversine Formula [29]. By setting the as-the-crow-flies time to be smaller than the time taken for a car traveling faster than the maximum speed to traverse the straight line path from x to y , $h(x, y)$ qualifies as an admissible heuristic since

$$h(x, y) \leq d(x, z) + h(z, y)$$

for any point z where equality holds only if z is on the shortest arc from x to y .

Since there is a known effect of the hour of day on the traffic delays on road segments, this feature was taken into account for computing the distance function $d(x, y)$ in the A-star search algorithm. Rather than using just the travel time at the speed limit for the road segment, the expected travel time based on hour of day $d(x, y|t)$ was used.

To compute this quantity, all delay observations for the road segment were first split into 24 bins, one for each hour of the day. Then then expected travel time $d(x, y|t)$ was computed as the sample mean of bucket hour(t) (hour of day corresponding to time t) for the shortest computed path from x to y .

While this system performed reasonably well, it was evident that one could improve on the prediction accuracy by utilizing various traffic influencing factors (Section 3.1.2). Further, it was not clear if splitting the observations into 24 buckets, one for each hour of the day, was the optimal granularity for delay estimation. My goal in this project was to develop a road traffic prediction technique that could significantly reduce errors in traffic delay predictions while still being computationally feasible to integrate into the CarTel's online routing engine.

Chapter 3

Design and Implementation

Estimation of traffic delays is a complex problem and has many components. Firstly, we need a data source that gives us the traffic information for delay prediction and validation. We then need mechanisms to filter noisy data from our source to increase the data accuracy. Next, we need to come up with a traffic model and determine the variables that influence traffic. With this information, we can then develop an algorithm for predicting traffic delays. Finally, we need a mechanism to validate the predictions and determine the accuracy and efficiency of the algorithm.

3.1 Design

Our prediction system begins with the collection of GPS data from taxicabs. Travel delay information is then extracted from the raw GPS points through the Map-matching algorithm. Weather conditions obtained from the National Weather Services database, day, time and other information are all compiled which each delay observation to populate a database of historical observations for each road segment. With the data for a road segment all in one place, we then use this information build a classification tree for traffic delay predictions for each road segment.

When our system receives a query to estimate the traffic delay for a route, the route

is first broken down into its corresponding road segments. Using the classification trees, we then predict delays for individual segment and finally sum the predictions together to obtain an estimate for the expected travel time along the route. Figure 3-1 summarizes the flow of data in or system and how predictions are obtained.

Before we design an algorithm to extract traffic delays, let us first explore the data sets we use to determine the strengths and limitations of our information.

3.1.1 Data Source

This project uses two primary types of data - traffic data from vehicles traversing roads and street layout information of the corresponding cities.

Traffic Data

As described in Section 2.2.1, we get traffic information primarily from GPS devices on taxicabs in and around the Greater Boston area. The information collected by these devices is relayed to our database via the Cabernet system [14]. The data is available across a three year period from January 24th 2007 to February 4th 2010 in three chunks (Table 3.1). As we go from the first chunk to the third chunk, our lab updated the number of hardware GPS nodes, made some improvements to the Map-matching algorithm, etc.

Vehicle	Start Date	End Date	count
Taxi	Sun Sep 16 14:14:28 2007	Tue Oct 16 16:45:29 2007	7,383,910
Taxi	Mon Jan 5 08:10:23 2009	Thu Feb 4 16:53:44 2010	12,209,422
Taxi	Thu Feb 4 16:55:16 2010	Wed Jul 28 15:26:09 2010	3,485,718
Bus	Wed Apr 28 22:16:10 2010	Tue May 18 23:59:58 2010	13,115,450

Table 3.1: Data Sets

An important characteristic of the GPS data procured from taxicabs is that the data coverage across different road segments is not uniform. Some of the streets in Boston have very good historical coverage. These streets typically correspond to those

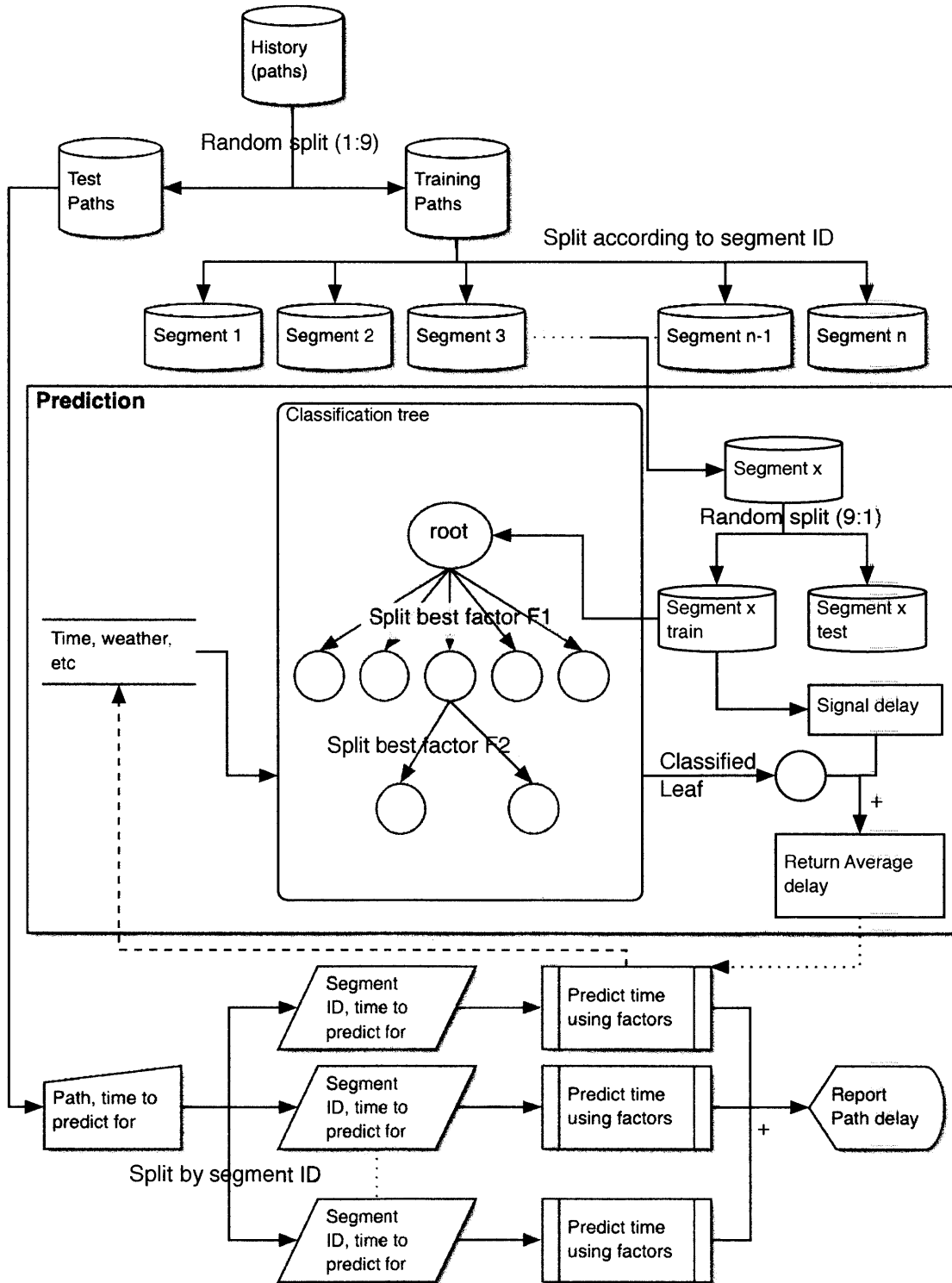


Figure 3-1: Flow chart of the traffic delay prediction algorithm

around the Boston Logan airport and the streets around the taxicab parking garages. Some commonly used highways and streets have moderate historical coverage like segments of the Massachusetts Turnpike within the Boston city limit.

However, most other road segments have very limited coverage across history. In fact, there are many road segments that our taxis never used during the entire two year period. We have no information about the actual traffic behavior on these segments as they have no coverage.

The number of observations per road segment is an inverse exponential distribution. This trend is displayed in Figure 3-2 using a random sample of 10,000 road segments, each with more than 100 observations.

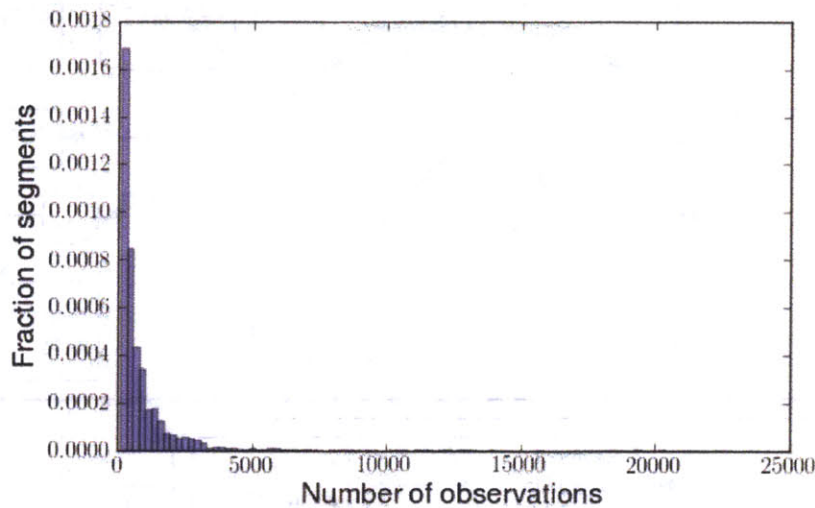


Figure 3-2: Prediction accuracy vs number of observations for individual road segments

This sparsity in coverage primarily arises from the limited number of deployed nodes available to us. We currently have at most 27 taxi-cabs providing GPS data at any given instant. This number decreases as we go further back in history since we had even fewer tracker nodes deployed. The reason we have some well covered segments and some poorly covered segments is due to the behavior of taxis. Since taxis pick up passengers primarily from places like the airport and commonly drop off people at locations such as shopping complexes and hotels, we tend to have good

coverage of road segments around these hubs. However, most road segments in Boston do not have daily data available across history.

It is important to note one caveat regarding the information available from these regions; taxis tend to exhibit behaviors that cause reported speeds to deviate from that of normal traffic speeds. This effect is most pronounced when taxis are picking up or dropping off passengers. Since the only information we obtain from the taxi cabs is their location, we have no means of directly distinguishing between a taxicab parked on a side of a street and one stuck at a traffic signal. Taxis driving around slowly waiting to be called have GPS traces that are indistinguishable from traces of cabs moving on a congested road segment.

Yet another artifact of GPS data is that it can be noisy. This noise can become the quite significant in regions with cloud cover, high rise buildings, tunnels and other regions with environmental disturbances. Therefore, our data from downtown Boston is typically more noisy than those from the suburbs. Our data must be pre-processed (Section 3.2) to reduce this noise before it can be used for inference .

MBTA Bus Data

In addition to the GPS data collected from taxi-cabs, we also had access to GPS trackers installed by the MBTA (Massachusetts Bay Transportation Authority) in public transit buses in Boston for a period of about a month in May 2010 (Table 3.1). The sampling frequency here was at a much lower rate (on average, one sample every two minutes) than what we had for cars (roughly one sample very second). This low sampling rate made it difficult for us to accurately determine the speed of travel of the bus or the travel time on an individual road segment.

Additionally, this data contained no information about how much time each bus spent at a bus stop. This combined with the low sampling rate made filtering out the true travel time on a road segment very difficult.

Street Layout Data

Individual GPS coordinates themselves don't directly enable us to determine which street segments the vehicle traveled. The actual road traversed must be inferred from the sequence of GPS points. In order to identify these roads, we first need a map of the road network.

We use two road segment maps : one compiled by Navteq [26] and the other from Open Street Maps [17]. Both maps have detailed information available for all streets in the USA, but we restricted ourselves to streets in and around the Boston, MA.

Both the Navteq and Open Street Maps databases split up streets into multiple road segments to ensure that proper road adjacency information is provided and all intersections are taken into account. Many times, the intersection itself is reasonably long, in which case, a separate tiny road segment is created to account for the road length of the intersection. This process of splitting streets into smaller road segments yields road segments that vary widely in length but uniquely identify each segment. Additionally, information about one-way and two way streets, altitude (when there are overlapping roads), etc are provided which allows us to distinguish between road segments and determine the direction of flow of traffic on these segments.

One important feature of this data set is that intersections occur at the end points of road segments. When a vehicle stops at a traffic signal, the GPS error causes the observed points to lie on multiple segments rather than the segment the vehicle is on. By tracking trajectories rather than looking at raw GPS points, we can narrow this down to two segments — the segment with the stop and the segment the vehicle travels on next. Even then, it is not trivial to determine how long vehicle stayed on the first segment versus the second. The problem of determining how long a vehicle stayed on each segment becomes even harder when cars take left turns (when they are in the middle of the intersection) or when the roads are congested since simple heuristics (like associating all the stop time with the first segment) do not work as

well.

Sometimes, these road segments can get as small as 5m, especially at intersections. Such small segments make traffic prediction difficult due to the same problem of associating GPS points to road segments. It becomes hard to determine whether the vehicle got stuck on the short road segment or on any of its adjacent ones. This results in a large variance in the observed travel times for such segments where the observed distribution of travel times can deviate quite drastically from the true travel time. Conversely, observed delays across longer road segments more accurately reflect the actual delay experienced by the vehicle. This results in our distribution of observed travel times to more closely follow the ground truth. More details about matching GPS points to road segments can be found in Section 3.2.

The Navteq database provides us with the road segments in USA with their lengths (in km) and speed limits (in kph). For this project, we use kph as our unit for speed and km as the unit for distance.

Weather Data

A goal of this project was to consider the effect of weather on traffic conditions to help us more accurately predict traffic delays. To do so, we first needed a source of weather information.

We obtained weather data from the National Oceanic and Atmospheric Administration's weather service database [28]. This data provides weather recordings from various weather stations around a given area. We used the KBOS weather station [27] located at Boston Logan International Airport as the weather provider for the Greater Boston.

The weather data consisted of hourly observations from a large range of weather sensors that provided information on temperature, humidity, cloud cover, amount of precipitation and more. We used the data provided in their *Surface Hourly Abbrevi-*

ated Format (Appendix A). This format consists of an enumeration of one hundred different weather conditions such as “Mist” or “Patches of shallow fog or ice fog at the station, whether on land” and has separate enumerations for different types and intensities of precipitation such as “Drizzle (not freezing) or snow grains not falling as shower(s)” or “Rain, freezing, slight” or “Rain, freezing, moderate or heavy” or “Snow shower(s), moderate or heavy.” This set of enumerated values provided an adequate summary of weather information to utilize for traffic estimation.

3.1.2 Traffic Influencing Factors

Traffic is affected by a number of factors including time of day, day of week, weather, etc. These factors lead to changes in the delay of a road segment. Rush hour tends to be more congested than 1am traffic, snowy days lead to slower moving vehicles and weekend traffic tends to be lighter than weekday traffic. In Boston, we can even observe a clear effect of the month of the year on traffic speeds. The traffic tends to be lighter in August when compared to May, which can possibly be explained by the fact that most of the student population is gone due to summer vacation in August.

Here, we briefly explore the effects of the factors we account for when predicting traffic delays: time of day, day of week, month, weather conditions and previous and next segments.

- *Time of day:* Traffic is typically congested during rush hour whereas most segments are not congested during early morning hours. Using the time of day to classify traffic is probably one of the most important factors to help us estimate traffic delays due to the fixed routines of most humans. People typically commute to work between 7-8am and head back home around 5pm. These patterns increase traffic on road segments during the stated times, which gives rise to rush hour delays.



Figure 3-3: A road segment on I-93 with some evident traffic influencing factors

On the other hand, people are typically asleep at 3am in the morning resulting in unobstructed traffic flow during this time. At 3pm, most people are at work and so we observe traffic that moves faster than rush hour, but slower than 3am. By accounting for the time of day when estimating travel times, we can capture these periodic behaviors which very strongly dictate the behavior of traffic. We can clearly see the evening rush hour (4pm to 6pm) on a segment on I-93 (Figure 3-4) with significantly lower observed speeds when compared to the other hours of day.

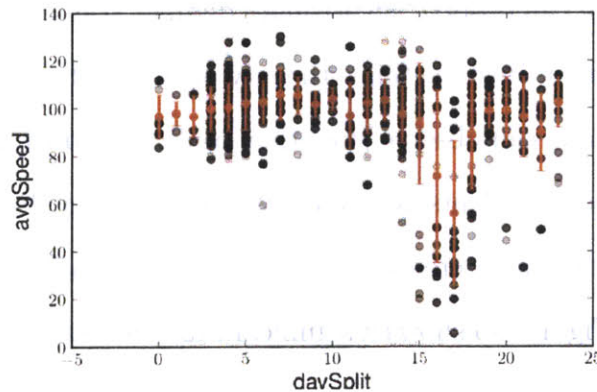


Figure 3-4: Classification of the I-93 road segment along the hours of the day

- *Day of week:* Weekend traffic exhibits different characteristics than weekday traffic. Rush hour traffic observed on weekdays are typically not present when

looking at weekend traffic. Also, people tend to commute to different places over the weekend. During weekends, routes to shopping centers and highways en route to beaches tend to experience heavier than average traffic conditions whereas most other roads experience a decrease in traffic. On the I-93 segment, we can observe higher speeds over the weekends and slower speeds over the weekdays. We can also see the increased variance in traffic over weekdays (Figure 3-5).

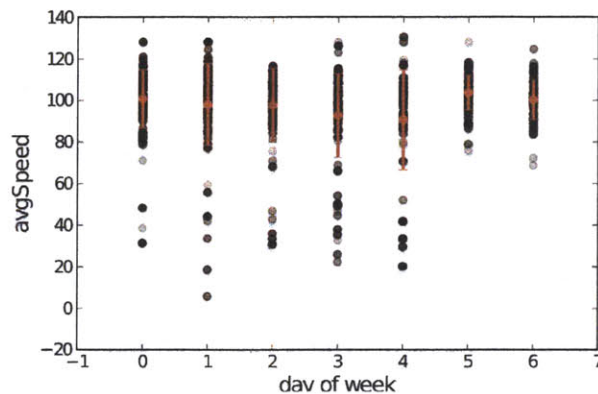


Figure 3-5: Classification of the I-93 road segment along the days of the week

- Month:* While the effect of the month of the year may not be that apparent in most places, we can observe surprisingly distinct traffic patterns for different months of the year in Boston. Since the city of Boston has a high percentage of college students, one can observe smoother traffic flow during summer vacation months. Since most students go home over the summer, Boston experiences a drop in population which creates a notable ease in traffic congestion on some road segments. Figure 3-6 shows a significant increase in observed traffic speeds in August.
- Weather:* Snow and heavy rain typically cause traffic to slow down significantly. These conditions experience an elevated rate of traffic accidents thereby further hampering traffic conditions. Furthermore, people tend to drive more cautiously

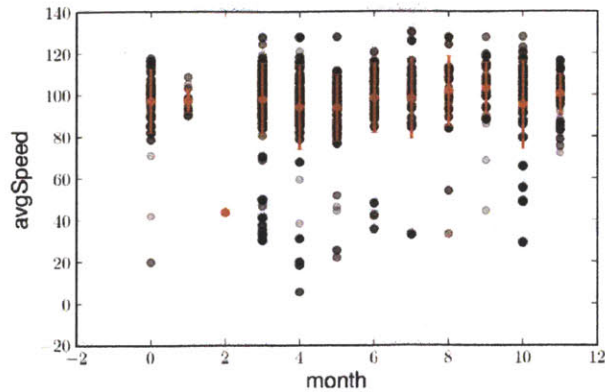


Figure 3-6: Classification of the I-93 segment along the months of the year. Labels go from 0=January to 11 = December

during inclement weather conditions, thereby decreasing the free-flow speed of traffic, even if the roads are not congested. Correspondingly, good weather conditions yields better flowing traffic, but also causes increased congestion on highways to local beaches during weekends.

It must be noted that we can not capture this behavior by just considering the months of the year that correspond to winter or summer. On winter days after snowstorms, the roads are typically clear thanks to snow plowing and salting, allowing the traffic on these days to flow better than on the days with snowstorms.

As we can see in Figure 3-7, the presence of precipitation results in slower traffic speeds (weather codes higher than 40). Further, we can also observe that we have very limited data from extreme weather conditions like snowfall (weather code 73).

- *Previous and next segments:* As described in Section 3.1.1, we have information about the segment a vehicle was previously on and information about the segment the vehicle took after the current segment. While this information is not so important in segments like the highways, it can become critical in better estimating travel times at intersections and other city streets.

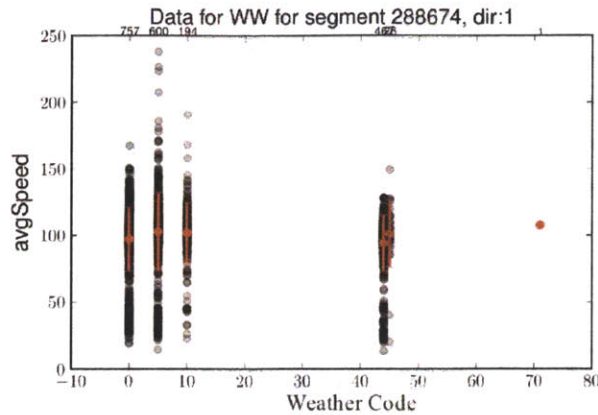


Figure 3-7: Classification of the I-93 segment along for different weather codes

In Boston intersections, those taking left turns have to yield to through traffic on the opposite side before turning on to the cross street. Consequently, on average, it takes vehicles much longer to take a left turn when compared to going straight through the light. Similarly, on intersections where right turns are allowed on red lights, drivers don't have to wait for the signal to change, thereby completely bypassing the signal delay.

Since right are not allowed on all red lights, not all intersections have signals or stop signs and signal patterns and cycle lengths can vary drastically from one intersection to another, sparse observations can severely impair our ability to provide low error estimates on traffic delays.

In order to deal with these differences across road segments, we estimate traffic delays at a segment by segment granularity.

3.2 Data Pre-Processing

In order to obtain individual traffic delays for each road segment, the raw GPS data is subject to a few pre-processing steps aimed at noise reduction, delay extraction and route determination. GPS points are Map-matched, drives are extracted from raw trajectories and outlier observations are removed.

3.2.1 Map-matching

The first step of pre-processing GPS points is the Map-matching process (Section 2.2.2) In this thesis, we assume that the Map-matching process provides us with the ground truth taxi trajectories. All predictions are based on travel times extracted from the map-matching algorithm and all prediction errors are calculated between predicted travel times and delays reported by the map-matching algorithm. Our traffic delay prediction model does not take into account possible errors that may arise through incorrectly mapped data points or erroneous delay times and average speeds.

If the road segments are small, then the assumption that Map-matching estimates correspond to the actual path the car took can break down due to the difficulty of accurately determining how much delay to associate with small segments.

3.2.2 Finding Drives

Once we have the GPS points mapped to their corresponding road segments, we need to process the data a little further before extracting the experienced delay. Since the data is gathered by taxicabs, we need to distinguish between a stop at a signal and when the taxi is simply parking on the roadside or dropping a passenger. Since traffic signals have much shorter cycle lengths than 5 minutes, we split a trajectory into a new drive when we observe a taxi that is stationary for more than five minutes. This allows us to exclude some of the taxicab specific delays from our observations.

Since we know the sequence of road segments at this point, we can use the road network database to identify the intersections at which the vehicle took a turn and the direction the vehicle was traveling. We drop the delays obtained from the first and last segments since they do not accurately determine our travel time on these segments.

After breaking trajectories into drives, we are finally ready to extract the travel

delay on each individual road segment from interpolation of the estimated hidden states. We store these separately for each road segment along with the other information such as previous and next road segment, observed speed and time of day.

3.2.3 Heuristic Outlier Removal

Despite these pre-processing steps, some errors propagate through, resulting in delay observations that seem physically impossible. For example, some processed observations indicate travel speeds on highways at twice the speed limit, which is highly unlikely and therefore a potentially spurious observation.

We filter these errors out by upper-bounding the speed limit to allow observations that are strictly under a threshold speed of 160 kph (roughly 100 mph).

In using the all the above pre-processing steps to extract traffic delay times, we are implicitly making the following assumptions:

- Taxicabs approximate the behavior of the general traffic
- Map matching accurately allocates data points to the corresponding road segments
- The extracted average speed and travel times are close to the ground truth of what the vehicle actually experienced

3.3 Algorithm

Given this background, we now present our delay prediction algorithm.

We use a modified classification tree approach to predict delays from our sparse data set. The predictions for individual segments are then combined together to estimate the delay experienced along a full trajectory from a source to a destination location. The high-level operation of this algorithm is summarized in Figure 3-1.

These steps involve finding the optimal traffic factor for classification for each level of the classification tree, merging branches with similar distributions and estimating the travel delay for the leaves of the classification tree.

We will first explore the traffic model we are using to construct our splitting criterion for classification.

3.3.1 Traffic Model

For a given road segment, we have a continuous distribution of observable speeds. This distribution for the road segment s is represented as a probability distribution over the traffic metric (either speed or travel time) X , given by $Pr_s[X = x]$. Figure 3-8 shows the typical shape of the probability density of speeds and travel times for a well behaved road segment. We consider a segment to be “well behaved” if the underlying traffic distribution looks like a mixture model of a few Gaussian distributions.

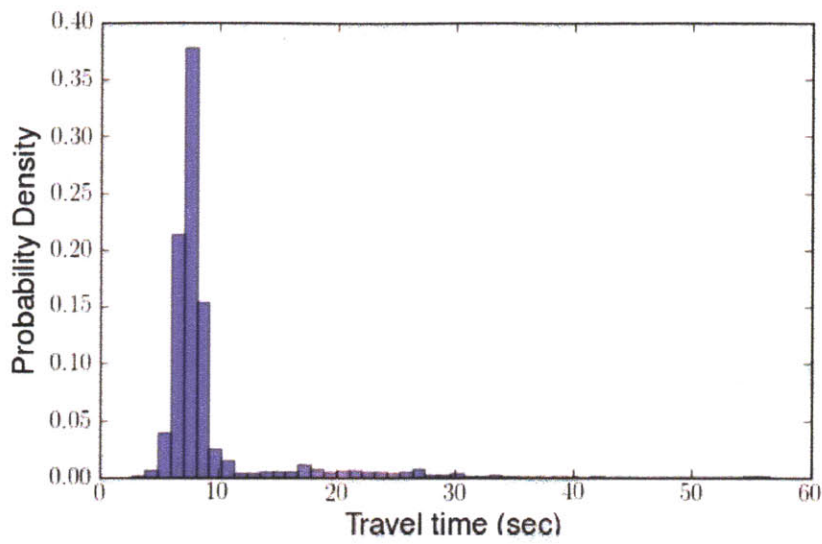
Figure 3-9 shows an road segment on Rt 1 where the probability distribution over vehicle speed does not show simple mixture model behavior.

For a given set of factors F and historical observations for the segment H_s , we can compute the speed that we would expect a car to observe on that road segments using the expected travel speed, as follows:

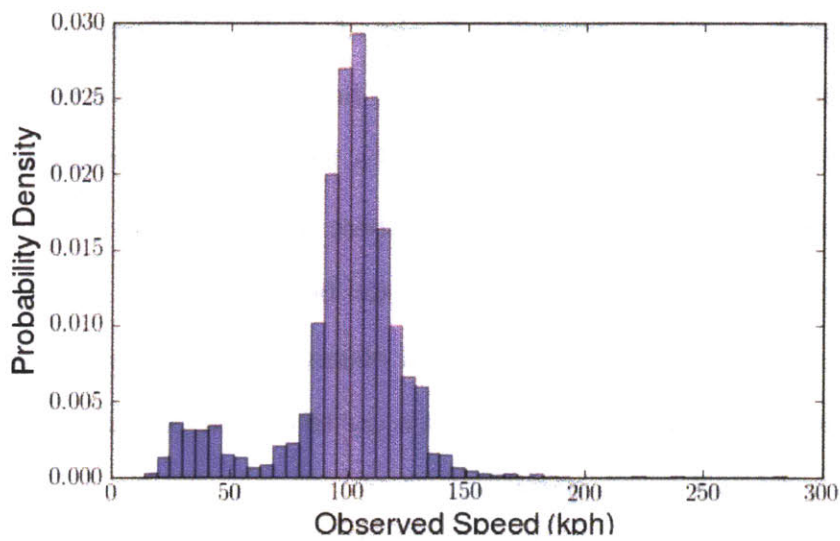
$$E[X|H_s, F] = \sum_x Pr_s[X = x|H_s, F] \cdot x$$

Where $E[Y|Z]$ is the expectation of the random variable Y , given Z and X is our variable of interest (traffic delay or speed). These factors F are the traffic influencing factors we explored in Section 3.1.2.

Given historical observations H_s for segment s , we can compute the set of observations $H_s|F$ that correspond to speeds observed under similar factors F in the past.



(a) Probability density of travel times



(b) Probability density of speeds

Figure 3-8: The probability densities for the I-93 segment. Note how the speed probability density appears to be a Gaussian mixture of two speed components

For example, if our factor is hour of day at 2pm-3pm, then our set $H_s|F$ are those observations across history on segment s that were taken between 2pm and 3pm. The inferred traffic metric (speed or travel time) \hat{X} is given by

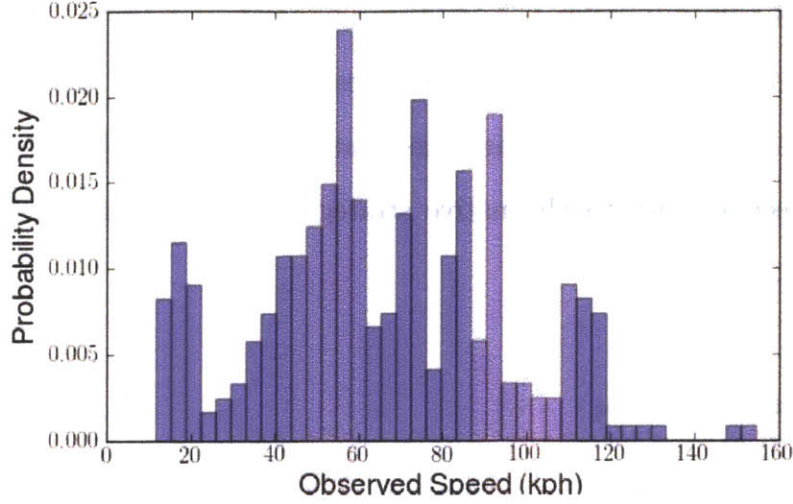


Figure 3-9: Probability distribution of a road segment on Rt 1 where a simple Gaussian mixture model assumption may not be valid

$$\hat{X}_s = \frac{1}{|H_s|F|} \sum_{x \in \{H_s|F\}} x \quad (3.1)$$

Now, as we obtain more observations for a given road segment, by the strong law of larger numbers, we have that $\hat{X}_s \rightarrow E[X|H_s, F]$. Note that we are making the assumption that our inferred travel times and speeds are independently and identically distributed. This is a reasonable assumption since different observations on the same street are made by different taxis at different times. Given the set of traffic influencing factors, we assume that the Markov property of time independence to hold.

If we had a very large number of observations recorded for every road segment and if we still had sufficient observations for every possible combination of factors, then we could simply predict the speed directly from the sample average by conditioning on the corresponding factors.

However, we do not have such extensive observations for any road segment. If we assume 24 different values for the time of day (one for every hour), 7 days of week, 12 months, 100 weather conditions and two previous and two next segments, we already

have more than 8 million possible factors. For our sample average to converge to the true average speed for a particular set of features, we would need much, much more than 8 million observations, which our sparse data set falls short by a huge margin. Therefore, we need to turn to alternative strategies for estimating and predicting traffic conditions.

3.3.2 Tree Classifiers

Rather than splitting our data along all the factors at once, we can instead consider splitting our data along one factor at a time until we reach a state with too few observations to continue the splitting process. If at some point of the classification process, we have split along features F_1 , F_2 and F_3 , then we can view our traffic metric estimation as

$$E[X|H_s, F_1 = f_1, F_2 = f_2, F_3 = f_3] = \sum_x x \cdot Pr_s[X = x|H_s, F_1 = f_1] \cdot Pr_s[X = x|H(H_s, F_1 = f_1), F_2 = f_2] \cdot Pr_s[X = x|H(H(H_s, F_1 = f_1), F_2 = f_2), F_3 = f_3]$$

Here, $H(X, F_i = f_i)$ is the subset of observations in X satisfying $F_i = f_i$.

This process allows us to make predictions based on at least a few features before running out of observations to further classify. However, a standard tree classifier still does not provide optimal splitting for our sparse data set. For example, if our first factor was *Hour of Day*, then we would have 24 partitions, one corresponding to each hour of day, at the top level. This would put 4:00pm and 12:00am in different buckets and we would be able to make different predictions of traffic delay for rush hour and midnight. Though this does not take all the features into account, we would still increase our accuracy of prediction.

However, though traffic at 2:00am is very similar to that at 3:00am, the classifier still splits the data along these buckets. Hence, when we split the second time (say along *Weather Condition*), we will have only the data from 3:00am in the 3:00am bucket which may not contain sufficient samples for further splitting without risking over-fitting our data. If we had somehow decided to combine the 2:00am and 3:00am buckets, then we potentially would have had sufficient samples to further split according to weather, which would have helped us better estimate our traffic metric.

To classify as much as possible without over-fitting our model and to intelligently split a factor only into buckets that help increase our prediction accuracy, our splitting criterion should satisfy the following conditions:

1. We should have sufficient data points in each sub-tree after a split for the sample average to reflect the true average.
2. We should merge two partitions of a factor together if they have similar predictions of the traffic metric.

The first rule ensures that we do not split if there are very few observations for given conditions and the second rule ensures that we split only if we obtain useful information from the split so as to provide the next splitting step, with more data.

Determining Useful Splits

If traffic observations across different factors were highly separable, then a simple linear classification strategy would suffice to determine the strength of classification. However, the traffic data obtained is not always so separable. We can see in Figure 3-11, that across different hours of the day, the observed speeds have large variances, but similar means. Here, a simple first order fit would not yield us an improvement in prediction accuracy. There is high overlap in the traffic metric probability density

between the buckets when we split based on a traffic influencing factor. Hence, the linear classification strategy does not provide us with a feasible classification criterion.



Figure 3-10: The road segment at the intersection of Austin St and New Rutherford Ave is very short and most of the data is available from interpolation. The data provided from such small segments typically prove to be very difficult to predict from

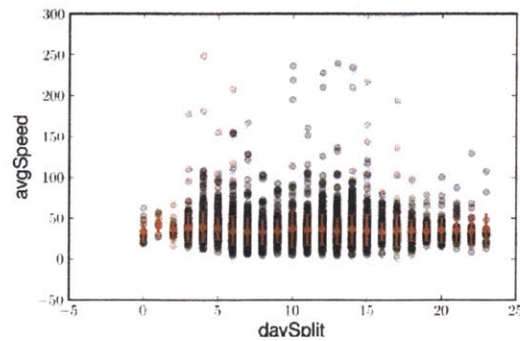


Figure 3-11: Some road segments like the intersection of Austin St and New Rutherford Ave have data that is not separable by a simple linear classifier

From Figure 3-8(b), it seems like we can use a simple Gaussian mixture model to estimate the parameters of the mixture components and then use the traffic factors to infer the probabilities of each of these components. However, it is not so simple. When we split based on factors, we almost always end up with another mixture model. This is especially true when we are looking at a segment with a traffic signal (Figure 3-12). There is typically an exponential mixture component that is associated with cars that had to wait at the signal and there is another Gaussian mixture component associated with those that travel through the signal without having to stop. When the mixture model starts becoming complicated, the number of parameters increase,

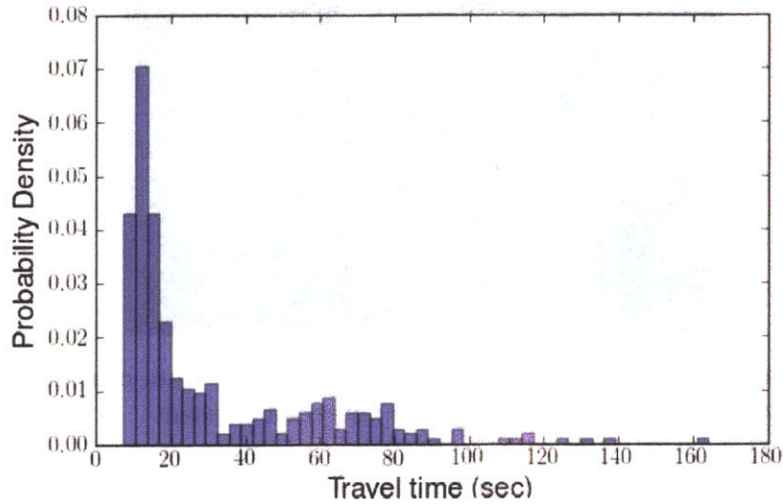


Figure 3-12: The probability density for Main St in Boston. This segment has a traffic signal on one end. We can see how the distribution looks much more complicated than a simple mixture model

thus requiring more samples to accurately estimate delays. This leads to the same problem as above of either over-fitting or limiting the number of features we are able to account for.

As mentioned before, there are some road segments that have limited historical data due to differences in coverage (Figure 3-13). Therefore, we must try to find a classification strategy that performs well when we have many observations but that doesn't completely break down when we have many fewer observations.

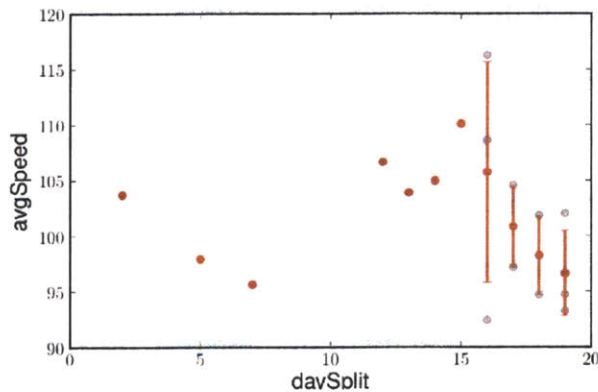


Figure 3-13: Some road segments like parts of Rt 2 near Athol have limited historical information



Figure 3-14: This section of Rt 2 in Athol is quite far away from central Boston and we have very limited data points for this road segment

Taking all these factors into account, we define a “useful” split to be one which separates two statistically significant speed distributions when comparing their cumulative distribution functions. Such splits yield buckets with statistically different traffic metric distributions when compared. The advantage of defining a useful split based on an arbitrary distribution (rather than a parametrized distribution) is that we can bypass the step of inferring the parameters of the distribution and skip directly to estimating the sample traffic metric average \hat{V} .

In order to compare distributions without directly computing the parameters of the model, we can use non-parametric statistical tests such as the Mann-Whitney U rank sum test or the Kolmogorov-Smirnov test to determine if two buckets represent statistically significant distributions. These tests provide us with a *p-value* that indicates the probability with which we can reject the null hypothesis that the two buckets have the same underlying distribution. We can then create a classification strategy that decides to split the observations only if the traffic influencing factor yields statistically significant results.

Note that if we can find such statistically different distributions, our prediction accuracy will improve since such differences imply that traffic behaves very differently under the different values of the traffic factor. By splitting the distributions, we are predicting different values for different traffic behaviors, which is exactly what we

want to do. Our algorithm uses the Mann-Whitney U test [24] as the non-parametric test of distribution similarity.

Modified Classification Strategy

With the splitting strategy defined, we can now build a classification tree using the desired factors. The ideal mechanism would be to construct all possible classification trees and pick the most accurate one via cross validation. However, the number of trees grows exponentially in the number of factors; finding the best classifier by constructing all possible classifiers quickly becomes infeasible, especially since such a classification tree needs to be built for each segment.

We instead use the following simple greedy strategy.

1. For each factor F , compute the splits that would be generated by grouping the observations for each value of F
2. Select the factor F that produces the best split (as defined below)
3. Classify using F to produce the required splits
4. Combine the branches that are not significantly different from each other
5. Delete branches with insufficient reported observations (less than 10)
6. Recurse on the branches

In order to determine which factor produces the “best” fit, we can consider the following strategies:

- Contains the most significant splits between two factors (largest difference in means). This enables us to immediately separate buckets where different predictions provide more accurate predictions.

- On average across all splits, provides the lowest p-value. This corresponds to the case where a split produces multiple significant splits. The average p-value is dominated by the highest p-value across the splits.
- Provides lowest geometric mean of p-value across splits. This strategy penalizes the highest p-value less.
- Provides lowest average variance across splits. Here, we aim to directly reduce uncertainty by finding splits that result in tighter delay distributions.

At the end of this process, we will be left with the classification tree that we can use for prediction. An example classification tree is shown in Figure 3-15. During the classification process, splits with no observations are ignored. This is evident in the classification split based on time of day where, for example, 1:00am is not a part of any sub-branch since there were none of our taxicabs traversed this road segment at that time.

3.3.3 Prediction

Once we have our classification strategy in place, we can go build a classification tree for each road segment. Before classification, we first split drives into test and train drives. This allows us to have separate training and test sets for evaluating the prediction scheme for both individual road segments and for full drives. We now use the training data for each road segment to build up the classification tree.

In order to predict the traffic delay for a particular segment we can simply traverse down the classification tree based on the factors that we used for each split until we reach a leaf node. At this point, we estimate the traffic metric by computing the sample average of the observations at the leaf node as described in equation 3.1.

If we need to predict the delay for a trajectory instead, we first split the path into its corresponding road segments $S = \{s_i\}$. We then predict the traffic delays on each

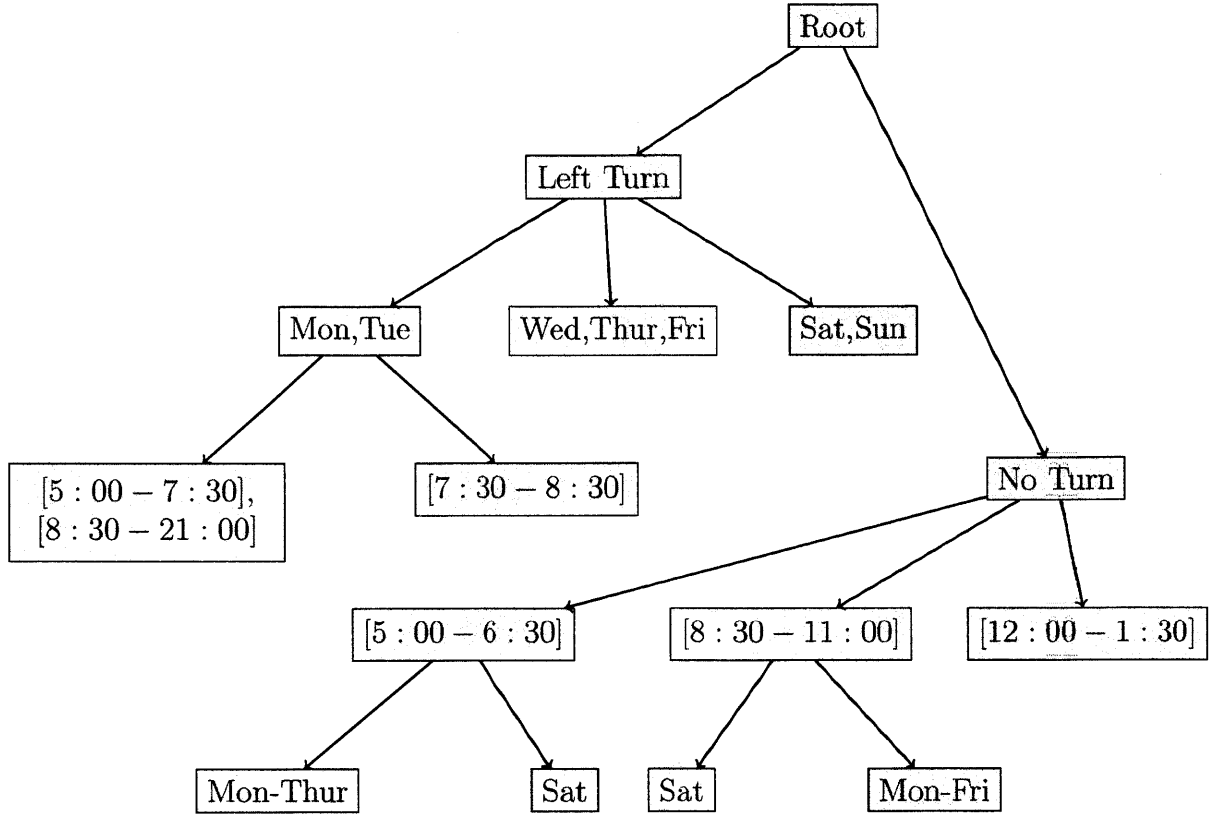


Figure 3-15: A sample classification tree generated by our algorithm.

road segment \hat{X}_i from the classification tree of s_i . Finally, we simply add the delays up to obtain the final delay.

$$\hat{X} = \sum_{s_i \in S} \hat{X}_i$$

This process of summing the delays actually helps us reduce the prediction error and increase the accuracy of prediction as discussed in Section 4.1.3.

A special case for obtaining delay predictions is when there are no observations for a particular value of a traffic factor. For example, let us consider a case where the classification tree splits for a particular road segment first based on the day of week and then splits according to the time of day. Let us further assume that there are no recorded observations for Tuesday on a particular road segment. Let the resulting

classification tree look like Figure 3-16

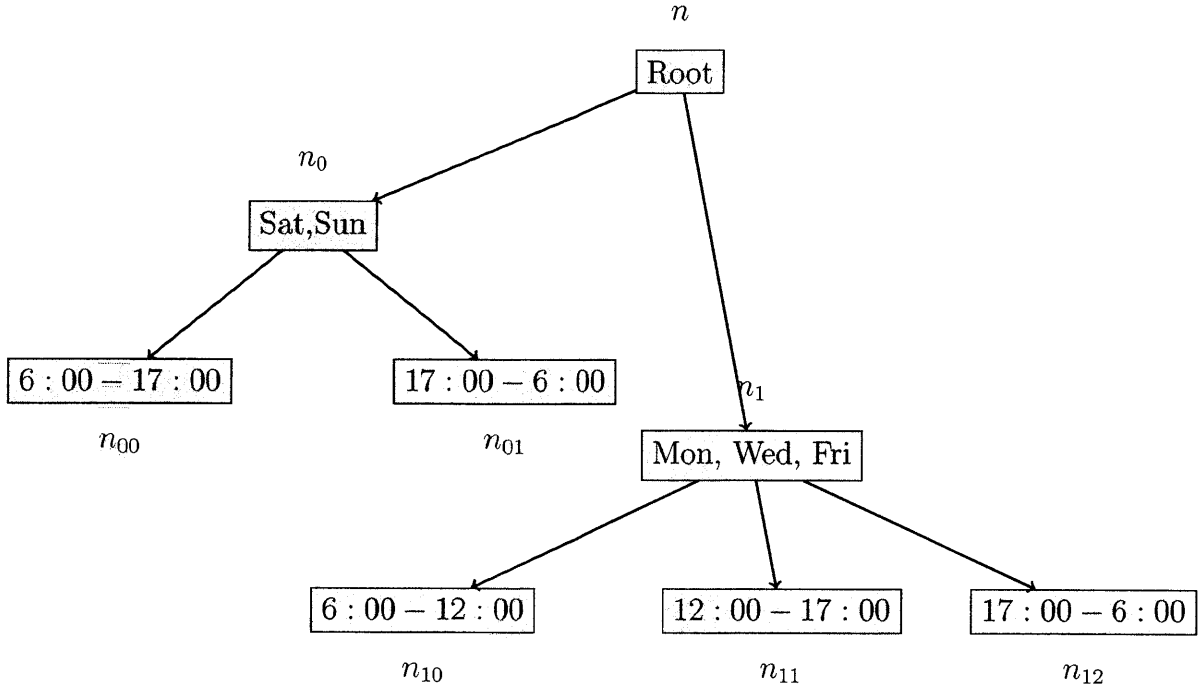


Figure 3-16: A sample classification tree generated by our algorithm

If we are then asked for a travel estimate for Tuesday at 8:00am, then following the standard classification procedure would result in estimating the delay as the mean of observed delay estimates in the root node. This prediction strategy does factor in the time of day.

In order to overcome this limitation, we continue classification by “marginalizing out” the *day of week* variable. This step involves finding the weighted mean of the predictions for the two branches n_0 and n_1 for the remaining factor (hour of day). This process can be mathematically described as

$$\hat{X} = \frac{|n_{00}| \cdot \hat{X}_{00} + |n_{10}| \cdot \hat{X}_{10}}{|n_{00}| + |n_{10}|}$$

Here, n_i is the number of observations associated with branch n_i of the classification tree and X_i is the predicted travel delay estimate for the classification sub-tree

with n_i as the root.

This step of marginalizing out traffic factor for which we have no observations allows the classification tree to utilize other factors for travel delay estimation, thereby improving the accuracy of predictions.

Chapter 4

Results

4.1 Methods and Results

In this section, we evaluate the accuracy of prediction on our taxicab data. We explore the performance of the prediction scheme at the level of individual road segments as well as on full routes. We limit our prediction to estimating traffic delays rather than traffic speeds since most people are interested in the time it would take to reach their destination.

In order to determine the performance of our method, we compare the prediction accuracy of travel times from it to that of classifying according to the hour of day only (referred to as the Naive Method).

4.1.1 Training and Test Data

As described in Section 3.3.3, we split our trajectories from taxicabs into a training set and a test set. These sets were randomly created such that the test data set comprises of 10% of the total paths (10 fold cross validation).

The training paths were then divided into their component road segments and the delays for these road segments were recorded as our “observations” for an individual

road segment.

When measuring performance at an individual road segment level, the observations for the road segment were further divided into test and training observations via 10-fold cross-validation.

4.1.2 Individual Road Segments

In order to determine how the classification strategy performs relative to the naive strategy of just splitting according to the hour of day, I first compared the predictive power of the two methods at an individual road segment level.

10,000 road segments were randomly chosen from the entire Boston network. Road segments with fewer than 10 testing observations (less than 100 observations overall) were not chosen as candidate test segments because no prediction strategy will perform well on segments with so little data.

From Table 4.1, we can see that the classification method gains a slight edge over the naive method at the individual road segment level. We can also see that the algorithm provides an exponentially decreasing probability of generating inaccurate predictions (Figure 4-1).

Statistic	Naive Strategy	Classification	Improvement
Average Mean abs(Error)	0.4023	0.3655	1.10
Average Median Error	0.2765	0.2258	1.22
Variance	0.3375	0.3000	1.13

Table 4.1: Prediction Errors on an individual road segment level

Though the predictive edge gained by the classification strategy seems marginal when looking at individual road segments, the classification approach significantly out-performs the naive strategy when we run predictions on full trajectories, as we show in the next section.

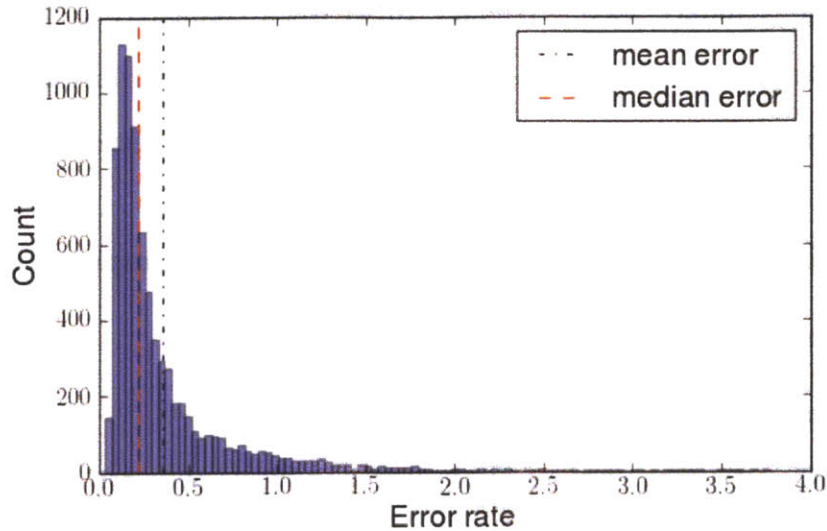


Figure 4-1: The distribution of the error rate for travel delays across 10,000 randomly chosen road segments shows that more than 50% of the time, travel delays are predicted with greater than 77% accuracy

4.1.3 Trajectories

In most cases, it is more important for a user to accurately know the end-to-end commute time accurately rather than the amount of time spent on each individual road segment.

As mentioned in Section 3.1.1, we obtained the data in three installments where slight changes were made in the number of cabs and the Map-matching process. We ran our algorithm on each chunk individually to determine the performance of our algorithm under these different conditions before we examined its performance with the combined data set.

Test paths were chosen to have travel time of more than 10 minutes when comparing the accuracy of travel delay predictions of the classification approach and naive approach. This ensured that trajectories closely reflected those that an actual commuter would take.

As we can see from Table 4.3 our algorithm provides best improvement in delay prediction accuracy and prediction confidence on the data chunk with the longest

span of coverage (January 2009 - January 2010) when compared to the other two data sets (Table 4.2 and Table 4.4). Our algorithm brings the median prediction error down from 11.14% to 6.89% thereby providing a 1.62 fold decrease in prediction error.

Statistic	Naive Strategy	Classification	Improvement
Mean abs(Error)	0.1363	0.1019	1.34
Median Error	0.1138	0.0759	1.50
Variance	0.0301	0.0181	1.66

Table 4.2: Prediction Errors on the first batch of data (September 2007 - October 2007)

Statistic	Naive Strategy	Classification	Improvement
Mean abs(Error)	0.1410	0.0920	1.53
Median Error	0.1114	0.0689	1.62
Variance	0.0332	0.0139	2.39

Table 4.3: Prediction Errors on the second batch of data (January 2009 - January 2010)

Statistic	Naive Strategy	Classification	Improvement
Mean abs(Error)	0.1758	0.1150	1.53
Median Error	0.1433	0.0892	1.61
Variance	0.0462	0.0230	2.01

Table 4.4: Prediction Errors on the third batch of data (February 2010 - July 2010)

When predicting the traffic delay for a trajectory, the input is given in the form of a set of segment IDs and the time at which we desire to predict the traffic delay. With the given time of day, we can determine the weather conditions from the National Weather Services weather database. This algorithm assumes that the weather conditions are roughly the same for all parts of the drive.

The next step of the algorithm is to split the path into segments and run the delay inference for each segment individually. Given the training data, the algorithm constructs the classification tree and then determines the best leaf node to predict the traffic conditions at the desired input time.

The delays obtained from this leaf node are then added to the total travel time of the car on the path.

When this process completes for all segments, we have a prediction of the total travel time of the vehicle at the given time of day. This prediction is then compared to the ground truth to calculate the prediction error.

By using 1000 randomly selected paths from the test data set, we see that the classification approach significantly reduces travel delay estimation error versus the naive scheme. It also reduces the uncertainty of our prediction (Table 4.5). With a median error rate of 6.75%, the system is able to predict the travel time of hour long drives to an accuracy of within 4 minutes.

From the error distribution (Figure 4-2), we can see a strong trend of decreasing frequency of predictions at higher error rates.

By visualizing the errors through overlaying the paths on a real map (Section 4.1.4), we were able to observe that most of our inaccuracies in predictions arose from incorrect signal delay predictions. Typically, the algorithm underestimated signal delays.

There are many reasons why predictions for individual segments results in larger errors than those over longer paths. One of these reasons is the averaging out of prediction error over multiple segments; since the prediction of traffic delay for a trajectory is the sum of predicted traffic delays of the corresponding road segments, some amount of the noise gets canceled out in this summation. For example, the estimated total time at signals over a trajectory has less variance than the variance for an individual signal, assuming that the signals are independent of each other (this is a valid assumption except for a very small set of coordinated traffic lights).

Another source of error which is averaged out in this process that is less obvious is the map-matching error around end points of road segments. The Map-matching algorithm is oblivious of the details such as location of traffic lights and the width

Statistic	Naive Strategy	Classification	Improvement
Mean abs(Error)	0.1349	0.0908	1.46
Median Error	0.1083	0.0675	1.60
Variance	0.0303	0.0149	2.03

Table 4.5: Prediction Errors with delay observations from all three data sets

of intersections. Therefore, when the map-matching algorithm receives many points near an intersection, such as when a car has stopped at a traffic light, it is not evident which road segments correspond to the recorded GPS points. This results in the signal delays being allocated to different segments at the intersection for different runs at the same intersection.

By predicting traffic delays with different sets of factors, we found that the most dominant traffic influencing factors for highway segments are the hour of day and the day of week. We found that including weather conditions as a traffic factor marginally improved our predictions by reducing the average error rate by a little under 1%. Though severe weather conditions play a critical role in influencing traffic patterns, our taxicab data did not have good coverage of road segments during periods of inclement weather and therefore limited the observable impact of this factor.

In the case of city streets, the hour of day, day of week and the segment a vehicle turns onto at an intersection all strongly influence traffic conditions. The behavior of a vehicle at an intersection was typically the first splitting factor or the second (after hour of day) in the generated classification trees, indicating that, as expected, the direction a vehicle turns at an intersection has a significant, observable effect on vehicle travel times. Once again, weather played a marginal role in the creation of the classification trees due to insufficient data coverage for different weather conditions.

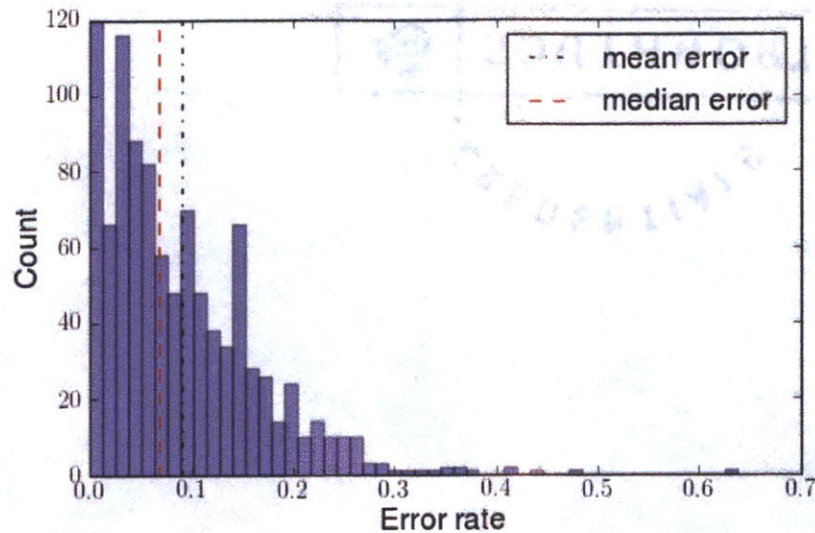


Figure 4-2: The distribution of the error rate for travel delays across 1,000 randomly chosen taxi drives (> 10 min) shows that more than 50% of the time, travel delays are predicted with greater than 93% accuracy

4.1.4 Visualization

To allow the user to visualize the predicted delays and the errors made by my algorithm, we implemented a Google Earth exporter that allows a user to open any path in GoogleEarth and see the segments and intersections that are predicted to be congested. (Figure 4-3).

The segments are colored according to the percentage error of predicted travel time vs the true travel time at the segment level. Green indicates good prediction accuracy and red indicates high error rates.

Additionally, one can click on any segment and obtain detailed information about the segment (Figure 4-4), including error statistics, the classification path down the classification tree and number of points used for classification.

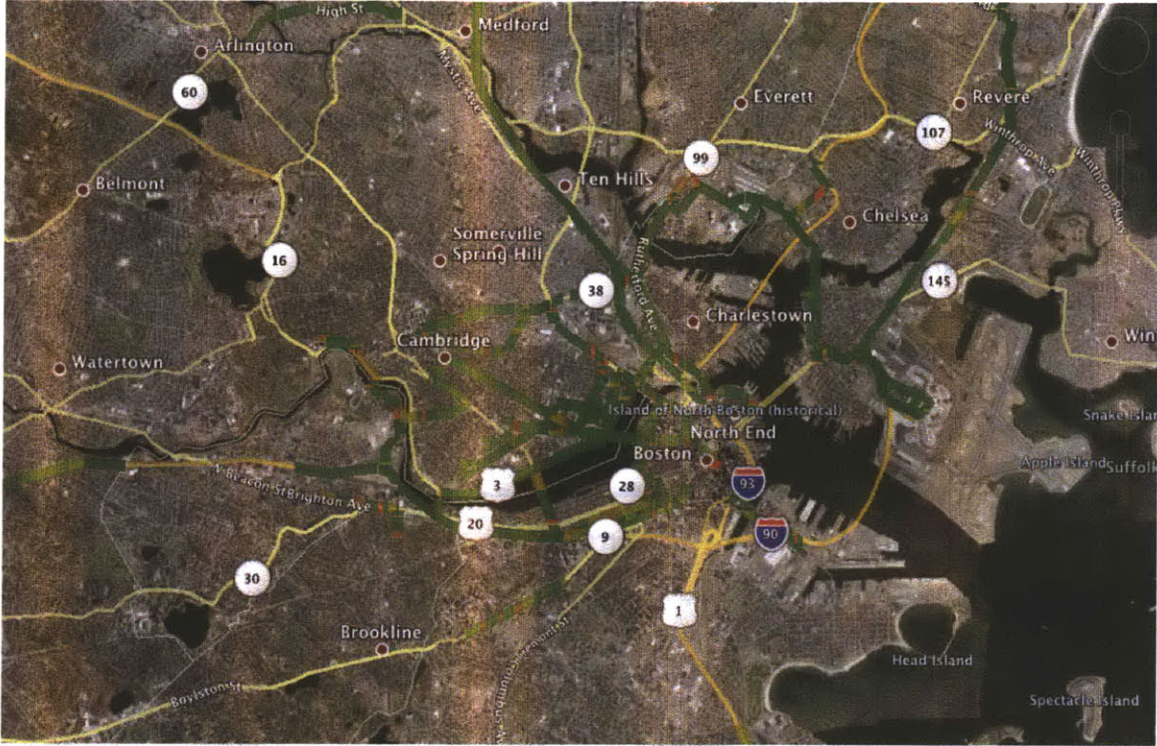


Figure 4-3: Green segments correspond to low error rates and red segments correspond to high error rates

4.2 Discussion and Future Work

While the above implemented algorithm significantly improves our prediction accuracy, there are still many improvements one can make to reduce the error rate and to increase the confidence of prediction. Clearly, enriching the data set by obtaining more data from taxicabs would enable us to provide better predictions, but since this process requires us to install hardware GPS nodes in more taxicabs, we will explore some other avenues of improvement.

4.2.1 Probabilistic Models

In our current traffic model, the our classification tree building procedure uses non-parametric statistical tests to create branches and predictions are made using the sample mean. We explored another approach with reported median speeds instead

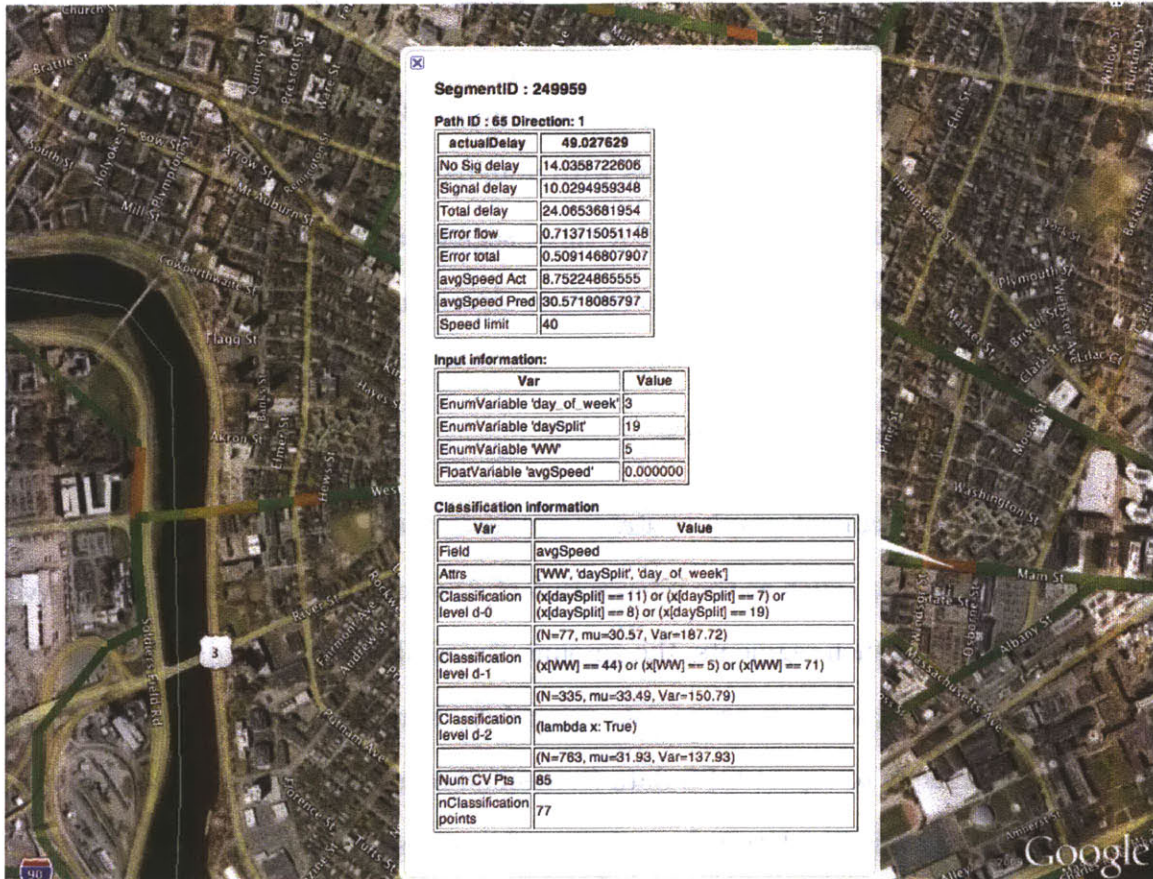


Figure 4-4: Advanced statistics and information is available by just clicking on the desired segment

of average speeds. This approach helped prevent outlier samples from significantly affecting prediction and increased our prediction accuracy at the level of an individual road segment. However, this scheme doesn't perform as well as predicting full trajectories due to the fact the linearity of expectations allows us to readily sum up the mean delays across segments to find the full path delay, whereas this property does not hold true for the median.

Rather than statistical methods, using a probabilistic model to build the tree and predicting speeds based on the maximum likelihood estimate could help provide more accurate estimates of travel delays. However, learning a continuous distribution (without making strong Gaussian assumptions) takes a lot of training data points. On the other hand, estimates of the mean converge more quickly to the true value

with fewer observations.

Finally, as we obtain more data, building more complicated probabilistic models may become feasible for those segments with abundant data points. In this scenario, we can use the classification strategy for those segments with insufficient data while using a more accurate and powerful prediction algorithm for the other segments.

4.2.2 Grouping Segments

The Navteq street segment graph contains segments that are very small (many that are less than 10m long) such as the intersection of Austin St and New Rutherford Ave (Figure 3-10). With such segments, the amount of available historical data is typically very small compared to longer segments around it. Further, since slow moving cars are much more likely to report GPS data from small segments when compared to fast moving cars, the available data is biased towards congested conditions.

A naive approach to solve this problem would be to merge adjacent segments. While this would work on highways, it is not possible to merge other road segments due to traffic intersections or forks in the road.

In such cases, a more optimal approach would be to implement an additional module to this algorithm that determines which streets have similar traffic characteristics to each other based on various factors. This would enable us to pool together data from various segments. Pooling data is especially helpful when we have limited information available for a road segment as it allows us to either make better predictions or use a more powerful machine learning algorithm.

We started experimenting with this technique, but simply scaling speeds or travel times from one segment to directly predict the estimated travel time on another did not work as expected. This is due to the fundamentally different traffic distributions that exist across road segments. For example, the addition of a traffic light, the presence of a yield sign, or the right-on-red policy results in drastically different

travel times across otherwise similar segments.

In order to overcome this problem, we are trying alternate approaches such as finding a measure of “congestion” in one road segment and using this information as an attribute to classify along when predicting the delay across another road segment where we have observed a strong correlation in traffic patterns. The large number of road segments in the Navteq network makes finding such correlations hard, but we are exploring some heuristics like using this strategy for segments where we have particularly high errors or when more recent traffic is available for a nearby segment.

4.2.3 Using Bus data to predict car delays

In order to further decrease our prediction error, we tried to use real time traffic delays from buses in order to predict the expected delays for cars. Using roughly three weeks worth of data from MBTA bus routes, we Map-matched these trajectories to obtain periodic observations for bus routes in the Boston area. When predicting delays for a route traveled by a car, we used the bus delays and historical car delays to predict the travel time on road segments that were on bus routes and just historical delays otherwise.

Unfortunately, this mechanism performed worse than just using historical delays. One of the primary reasons for this drop in performance was that the MBTA bus data was reported at a granularity of approximately one observation every two minutes for a single bus. These coarse observations not only increased our uncertainty in inferring the bus travel delay across a particular road segment, but also made it difficult to filter out delays introduced by bus stops.

4.2.4 Traffic Aware Routing

As mentioned in Section 2.2.3, as a part of the CarTel project, our group already implemented a routing system that uses the A-star algorithm to find the trajectory

with the shortest travel time from a source location to a destination. We are currently working on integrating our predictions into the A-star routing scheme.

In the A-star scheme, the goal is to set the distance function $d(x, z)$ to the values estimated by my traffic prediction scheme. Hence, a run of A-star would result in the shortest predicted travel time from the source to the destination. We note that our admissible heuristic $h(z, y)$ still holds true, meaning the routes obtained from A-star should reflect the real least transit time paths if our predictions are accurate.

Another interesting application is routing using paths of low variance. This would be useful for cases where commuters have strict bounds on when they need to reach their destination. For instance, if we are traveling to Boston Logan Airport, then we would prefer to take a slightly longer path with low variation in travel time and leave our homes slightly earlier rather than just take the path of lowest travel time, which may have higher variance.

Though our predictions return a mean and a variance, we have so far only considered the mean travel times on individual road segments to compute the net travel time. However, just as linearity of expectation holds across road segments, variances are also additive. Therefore, we can determine our confidence of prediction in terms of the total expected variance of travel time across the trajectory. We can route through low variance paths by utilizing this variance as our distance function in A-star, thereby providing users with reliable routes when required.

The advantage of integrating our prediction values with the routing scheme is that we obtain routes that are traffic-aware. Different routes between the same two points will be returned at different hours of the day, allowing congested road segments to be avoided and realistic travel estimates to be obtained in various conditions, such as during snow showers or torrential rain.

4.3 Conclusion

As we can see from the results, the distribution based tree classification approach cuts down error rates by almost a factor of 1.5 versus the naive scheme. This approach results in a median traffic delay prediction error rate of 6.75% on routes of ten minutes or longer. The algorithm also reduced the prediction variance by more than a factor of 2, providing much more reliable delay estimates for travel delay queries.

Once the model is trained, predicting delays simply involves traversing a path from the root of the prediction tree to the leaf, which is computationally very fast. Hence, predictions are very efficient. This allows the prediction trees to be used in algorithms like shortest path routing, socially optimal routing and flow control where many delay computations may be required.

Appendix A

NWS Weather Code Table

Code	Weather Condition
00	Cloud development not observed or not observable
01	Clouds generally dissolving or becoming less developed
02	State of sky on the whole unchanged
03	Clouds generally forming or developing
04	Visibility reduced by smoke, e.g. veldt or forest fires, industrial smoke or volcanic ashes
05	Haze
06	Widespread dust in suspension in the air, not raised by wind at or near the station at the time of observation
07	Dust or sand raised by wind at or near the station at the time of observation, but no well-developed dust whirl(s) or sand whirl(s), and no duststorm or sandstorm seen or, in the case of ships, blowing spray at the station
08	Well developed dust whirl(s) or sand whirl(s) seen at or near the station during the preceding hour or at the time of observation, but no duststorm or sandstorm
09	Duststorm or sandstorm within sight at the time of observation, or at the station during the preceding hour
10	Mist
11	Patches of shallow fog or ice fog at the station, whether on land or sea, not deeper than about 2 meters on land or 10 meters at sea
12	More or less continuous shallow fog or ice fog at the station, whether on land or sea, not deeper than about 2 meters on land or 10 meters at sea
13	Lightning visible, no thunder heard
14	Precipitation within sight, not reaching the ground or the surface of the sea
15	Precipitation within sight, reaching the ground or the surface of the sea, but distant, i.e., estimated to be more than 5 km from the station
16	Precipitation within sight, reaching the ground or the surface of the sea, near to, but not at the station
17	Thunderstorm, but no precipitation at the time of observation

18	Squalls at or within sight of the station during the preceding hour or at the time of observation
19	Funnel cloud(s) (Tornado cloud or waterspout) at or within sight of the station during the preceding hour or at the time of observation
20	Drizzle (not freezing) or snow grains not falling as shower(s)
21	Rain (not freezing) not falling as shower(s)
22	Snow not falling as shower(s)
23	Rain and snow or ice pellets not falling as shower(s)
24	Freezing drizzle or freezing rain not falling as shower(s)
25	Shower(s) of rain
26	Shower(s) of snow or of rain and snow
27	Shower(s) of hail (Hail, small hail, snow pellets), or rain and hail
28	Fog or ice fog
29	Thunderstorm (with or without precipitation)
30	Slight or moderate duststorm or sandstorm has decreased during the preceding hour
31	Slight or moderate duststorm or sandstorm no appreciable change during the preceding hour
32	Slight or moderate duststorm or sandstorm has begun or has increased during the preceding hour
33	Severe duststorm or sandstorm has decreased during the preceding hour
34	Severe duststorm or sandstorm no appreciable change during the preceding hour
35	Severe duststorm or sandstorm has begun or has increased during the preceding hour
36	Slight or moderate drifting snow generally low (below eye level)
37	Heavy drifting snow generally low (below eye level)
38	Slight or moderate blowing snow generally high (above eye level)
39	Heavy blowing snow generally high (above eye level)
40	Fog or ice fog at a distance at the time of observation, but not at the station during the preceding hour, the fog or ice fog extending to a level above that of the observer
41	Fog or ice fog in patches
42	Fog or ice fog, sky visible, has become thinner during the preceding hour
43	Fog or ice fog, sky invisible, has become thinner during the preceding hour
44	Fog or ice fog, sky visible, no appreciable change during the preceding hour
45	Fog or ice fog, sky invisible, no appreciable change during the preceding hour
46	Fog or ice fog, sky invisible, has begun or has become thicker during the preceding hour
47	Fog or ice fog, sky invisible, has begun or has become thicker during the preceding hour
48	Fog, depositing rime, sky visible
49	Fog, depositing rime, sky invisible

50	Drizzle, not freezing, intermittent, slight at time of observation
51	Drizzle, not freezing, continuous, slight at time of observation
52	Drizzle, not freezing, intermittent, moderate at time of observation
53	Drizzle, not freezing, continuous, moderate at time of observation
54	Drizzle, not freezing, intermittent, heavy (dense) at time of observation
55	Drizzle, not freezing, continuous, heavy (dense) at time of observation
56	Drizzle, freezing, slight
57	Drizzle, freezing, moderate or heavy (dense)
58	Drizzle and rain, slight
59	Drizzle and rain, moderate or heavy
60	Rain, not freezing, intermittent, slight at time of observation
61	Rain, not freezing, continuous, slight at time of observation
62	Rain, not freezing, intermittent, moderate at time of observation
63	Rain, not freezing, continuous, moderate at time of observation
64	Rain, not freezing, intermittent, heavy at time of observation
65	Rain, not freezing, continuous, heavy at time of observation
66	Rain, freezing, slight
67	Rain, freezing, moderate or heavy
68	Rain or drizzle and snow, slight
69	Rain or drizzle and snow, moderate or heavy
70	Intermittent fall of snowflakes, slight at time of observation
71	Continuous fall of snowflakes, slight at time of observation
72	Intermittent fall of snowflakes, moderate at time of observation
73	Continuous fall of snowflakes, moderate at time of observation
74	Intermittent fall of snowflakes, heavy at time of observation
75	Continuous fall of snowflakes, heavy at time of observation
76	Diamond dust (with or without fog)
77	Snow grains (with or without fog)
78	Isolated star-like snow crystals (with or without fog)
79	Ice pellets
80	Rain shower(s), slight
81	Rain shower(s), moderate or heavy
82	Rain shower(s), violent
83	Shower(s) of rain and snow mixed, slight
84	Shower(s) of rain and snow mixed, moderate or heavy
85	Show shower(s), slight
86	Snow shower(s), moderate or heavy
87	Shower(s) of snow pellets or small hail, with or without rain or rain and snow mixed, slight
88	Shower(s) of snow pellets or small hail, with or without rain or rain and snow mixed, moderate or heavy

89	Shower(s) of hail (hail, small hail, snow pellets) , with or without rain or rain and snow mixed, not associated with thunder, slight
90	Shower(s) of hail (hail, small hail, snow pellets), with or without rain or rain and snow mixed, not associated with thunder, moderate or heavy
91	Slight rain at time of observation, thunderstorm during the preceding hour but not at time of observation
92	Moderate or heavy rain at time of observation, thunderstorm during the preceding hour but not at time of observation
93	Slight snow, or rain and snow mixed or hail (Hail, small hail, snow pellets), at time of observation, thunderstorm during the preceding hour but not at time of observation
94	Moderate or heavy snow, or rain and snow mixed or hail(Hail, small hail, snow pellets) at time of observation, thunderstorm during the preceding hour but not at time of observation
95	Thunderstorm, slight or moderate, without hail (Hail, small hail, snow pellets), but with rain and/or snow at time of observation, thunderstorm at time of observation
96	Thunderstorm, slight or moderate, with hail (hail, small hail, snow pellets) at time of observation, thunderstorm at time of observation
97	Thunderstorm, heavy, without hail (Hail, small hail, snow pellets), but with rain and/or snow at time of observation, thunderstorm at time of observation
98	Thunderstorm combined with duststorm or sandstorm at time of observation, thunderstorm at time of observation
99	Thunderstorm, heavy, with hail (Hail, small hail, snow pellets) at time of observation, thunderstorm at time of observation

Bibliography

- [1] The cartel project. <http://cartel.csail.mit.edu/>.
- [2] X.J. Ban, R. Herring, P. Hao, and A.M. Bayen. Delay pattern estimation for signalized intersections using sampled travel times. *Transportation Research Record: Journal of the Transportation Research Board*, 2130(-1):109–119, 2009.
- [3] World Bank. Gross domestic product (2009). *World Development Indicators Database*, September 27 2010.
- [4] M. Ben-Akiva, M. Bierlaire, H. Koutsopoulos, and R. Mishalani. Dynamit: a simulation-based system for traffic prediction. In *DACCORS Short Term Forecasting Workshop, The Netherlands*. Citeseer, 1998.
- [5] S. Blandin, L. El Ghaoui, and A. Bayen. Kernel regression for travel time estimation via convex optimization. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 4360–4365. IEEE.
- [6] Paul Briedis and Dr Stephen Samuels. The accuracy of inductive loop detectors. In *2nd International Sprayed Sealing Conference & 24th ARRB Conference 2010 Presentations*, 2010.
- [7] A. Carroll and G. Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, pages 21–21. USENIX Association, 2010.
- [8] B. Coifman. Vehicle re-identification and travel time measurement in real-time on freeways using existing loop detector infrastructure. *Transportation Research Record: Journal of the Transportation Research Board*, 1643(-1):181–191, 1998.
- [9] En Colombia. Pico y placa, July 2010.
- [10] Hong Kong Traffic Department. Live webcast of traffic conditions, March 2010.
- [11] Andrew Downie. The world’s worst traffic jams. Retrieved 2008-06-20, March 04 2008.
- [12] Anthony. Downs and Brookings Institution. *Stuck in traffic : coping with peak-hour traffic congestion / Anthony Downs*. Brookings Institution ; Lincoln Institute of Land Policy, Washington, D.C. : Cambridge, Mass. :, 1992.

- [13] Rod Eddington. The eddington transport study, December 2006.
- [14] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *14th ACM MOBICOM*, San Francisco, CA, September 2008.
- [15] G.D. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
- [16] GovHK. Become a car owner in hong kong, March 2010.
- [17] M.M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, pages 12–18, 2008.
- [18] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.
- [19] E. Horvitz, J. Apacible, R. Sarin, and L. Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. In *Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 244–257. Citeseer, 2005.
- [20] J. Kwon, B. Coifman, and P. Bickel. Day-to-day travel-time trends and travel-time prediction from loop-detector data. *Transportation Research Record: Journal of the Transportation Research Board*, 1717(-1):120–129, 2000.
- [21] S. Lim, H. Balakrishnan, D. Gifford, S. Madden, and D. Rus. Stochastic motion planning and applications to traffic. *Algorithmic Foundation of Robotics VIII*, pages 483–500, 2009.
- [22] Samuel Madden, Paresh Malalur, and Hari Balakrishnan. icartel traffic portal. <http://icartel.net>.
- [23] N. Malviya, S. Madden, and A. Bhattacharya. A continuous query system for dynamic route planning. 2011.
- [24] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 1947.
- [25] MSNBC. Drivers losing five days a year to traffic jams. <http://www.msnbc.msn.com/id/20829879/>, Sep 2007.
- [26] Navteq. Navteq map database. <http://www.navteq.com/>.
- [27] NOAA. Kbos weather station. <http://weather.noaa.gov/weather/current/KBOS.html>.
- [28] NOAA. National weather service database. <http://nws.noaa.gov/>.

- [29] CC Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.
- [30] D.L. Schrank, T.J. Lomax, and Texas Transportation Institute. *The 2010 urban mobility report*. Texas Transportation Institute, Texas A & M University, 2010.
- [31] Arvind Thiagarajan, Lenin Ravindranath Sivalingam, Katrina LaCurts, Sivan Toledo, Jakob Eriksson, Samuel Madden, and Hari Balakrishnan. VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones. In *7th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Berkeley, CA, November 2009.
- [32] Peng Zhuang, Yi Shang, and Bei Hua. Statistical methods to estimate vehicle count using traffic cameras. *Multidimensional Syst. Signal Process.*, 20:121–133, June 2009.