2014

# Crowdsourcing traffic data for travel time estimation

Chanukya Chowdary Gadde
*West Virginia University*

# CROWDSOURCING TRAFFIC DATA FOR TRAVEL TIME ESTIMATION

by

Chanukya Chowdary Gadde

Thesis submitted to the

Benjamin M. Statler College of Engineering and Mineral Resources

at West Virginia University

in partial fulfillment of the Requirements

for the degree of

Master of Science in

Electrical Engineering

Yaser P. Fallah, Ph.D., Chair

Vinod Kulathumani, Ph.D.

Avinash Unnikrishnan, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia

2014

Keywords: Travel Time Estimation, Map Matching, Crowd-sourcing, Support Vector Regression

# ABSTRACT

## CROWDSOURCING TRAFFIC DATA FOR TRAVEL TIME ESTIMATION

Chanukya Chowdary Gadde

Travel time estimation is a fundamental measure used in routing and navigation applications, in particular in emerging intelligent transportation systems (ITS). For example, many users may prefer the fastest route to their destination and would rely on real-time predicted travel times. It also helps real-time traffic management and traffic light control. Accurate estimation of travel time requires collecting a lot of real-time data from road networks. This data can be collected using a wide variety of sources like inductive loop detectors, video cameras, radio frequency identification (RFID) transponders etc. But these systems include deployment of infrastructure which has some limitations and drawbacks. The main drawbacks in these modes are the high cost and the high probability of error caused by prevalence of equipment malfunctions and in the case of sensor based methods, the problem of spatial coverage.

As an alternative to traditional way of collecting data using expensive equipment, development of cellular & mobile technology allows for leveraging embedded GPS sensors in smartphones carried by millions of road users. Crowd-sourcing GPS data will allow building traffic monitoring systems that utilize this opportunity for the purpose of accurate and real-time prediction of traffic measures. However, the effectiveness of these systems have not yet been proven or shown in real applications. In this thesis, we study some of the current available data sets and identify the requirements for accurate prediction. In our work, we propose the design for a crowd-sourcing traffic application, including an android-based mobile client and a server architecture. We also develop map-matching method. More importantly, we present prediction methods using machine learning techniques such as support vector regression.

Machine learning provides an alternative to traditional statistical method such as using averaged historic data for estimation of travel time. Machine Learning techniques played a key role in estimation in the last

two decades. They are proved by providing better accuracy in estimation and in classification. However, employing a machine learning technique in any application requires creative modeling of the system and its sensory data. In this thesis, we model the road network as a graph and train different models for different links on the road. Modeling a road network as graph with nodes and links enables the learner to capture patterns occurring on each segment of road, thereby providing better accuracy. To evaluate the prediction models, we use three sets of data out of which two sets are collected using mobile probing and one set is generated using VISSIM traffic simulator. The results show that crowdsourcing is only more accurate than traditional statistical methods if the input values for input data are very close to the actual values. In particular, when speed of vehicles on a link are concerned, we need to provide the machine learning model with data that is only few minutes old; using average speed of vehicles, for example from the past half hour, as is usually seen in many web based traffic information sources may not allow for better performance.

# ACKNOWLEDGEMENTS

First I would like to express my sincere gratitude to my advisor, Dr. Yaser Fallah, for giving me an opportunity to work with him. His continuous guidance on how to think, analyze and solve problems helped me personally to develop my skills in doing research.

On this occasion, I would like to thank Dr. Vinod Kulathumani for his advice and valuable guidance to do my research. His motivation for students to understand, analyze and come up with new ideas further helped me build good foundation in doing my research.

I am also very thankful for Dr. Avinash Unnikrishnan for his valuable comments and suggestions in my research. I also want to thank him for providing me a dataset for my thesis.

I would like to thank my family for the continuous support they provided throughout my studies and my friends who stood by me in doing research. I want to thank West Virginia University for my graduate studies and for creating a platform to achieve all my dreams.

# TABLE OF CONTENTS

## Contents

LIST OF FIGURES ..... iv
LIST OF TABLES ..... vi
Chapter 1: Introduction ..... 1
1.1 Problem Statement ..... 3
1.2 Literature Review ..... 5
1.2.1 Development of Traffic Measurement Systems ..... 5
1.2.2 Location Detection and Traffic Sensing using GPS ..... 7
1.3 Travel Time Estimation Methods ..... 9
1.3.1 Extrapolation Methods ..... 9
1.3.2 Statistical Methods ..... 10
1.3.3 Theoretical Methods ..... 11
1.3.4 Methods based on Historic data ..... 13
1.3.5 Regression Models ..... 15
1.3.6 Time series methods ..... 15
1.3.7 Machine Learning Models ..... 16
1.3.8 Kalman Filtering Models ..... 19
1.4 Similar Projects ..... 20
1.4.1 Mobile Century Project ..... 20
1.4.2 The PUMAS Project ..... 22
1.5 Thesis Structure ..... 22
Chapter 2: Crowdsourcing of Traffic Information ..... 24
2.1 Client Architecture ..... 25

2.1.1 Application Components ..... 26
2.1.2 MainActivity ..... 28
2.1.3 Menu Items ..... 28
2.1.4 Android Location Provider ..... 28
2.2 Server Architecture ..... 31
2.3 Data Preprocessing ..... 31
2.3.1 Map Matching ..... 34
Chapter 3: Travel Time Estimation using crowdsourced data ..... 38
3.1 Overview ..... 38
3.2 SVM Architecture ..... 39
3.3 Support Vector Regression ..... 41
3.3.1 Linear Regression ..... 41
3.3.2 Non-Linear Regression ..... 44
3.4 SVR Development using LIBSVM ..... 46
3.5 SVR Training, Testing and Validation Datasets ..... 46
3.6 Kernels Functions ..... 48
3.7 Feature Scaling ..... 49
3.8 Model and Parameter Selection ..... 49
3.8.1 Model Selection ..... 49
3.8.2 Parameter Selection ..... 50
3.9 TTE using SVR technique ..... 51
3.9.1 Crowdsourced half hour mean speed and Current mean speed models ..... 52
3.10 Experiments ..... 54
Chapter 4: Conclusions ..... 73
Bibliography ..... 75
Appendix ..... 79

# LIST OF FIGURES

Figure 1.1: Simple road network ..... 4

Figure 1.2: Above road network representation using nodes and links ..... 4

Figure 1.3: Schematic diagram for illustrating extrapolation methods ..... 9

Figure 1.4: Stretch of freeway I-880 CA, used in the Mobile Century experiment [36] ..... 21

Figure 2.1: Proposed Server client architecture ..... 33

Figure 2.2: Simple road network representation using directed graph ..... 34

Figure 2.3: Simple Matching Process ..... 35

Figure 2.4: Systematic representation of map matching for road network ..... 36

Figure 3.1: Optimal separating plane for data ..... 39

Figure 3.2: Loss Functions ..... 42

Figure 3.3: Systematic illustration of how SVR predictor works ..... 53

Figure 3.4: Travel time estimation system representation ..... 53

Figure 3.5: Validation of SVR Estimator performance on Mobile Century dataset with true inputs ..... 57

Figure 3.6: Comparison of estimated travel time with historic mean on Mobile Century dataset for link 1 ..... 58

Figure 3.7: Comparison of estimated travel time with historic mean on Mobile century dataset for link 3 ..... 58

Figure 3.8: Estimation error for all the trips points using true values of speed and ToD as inputs ..... 59

Figure 3.9: CDF plot for errors in Mobile Century data with true values of predictor ..... 60

Figure 3.10: Estimation using Crowd-sourced historic half hour mean speed vs historic mean ..... 61

Figure 3.11: (left) Estimation error for all trip points using crowdsourced data and rbf kernel and (right) cdf plots for errors ..... 61

Figure 3.12: Travel time (sec) distribution on a link 8 at a give time 11 AM ..... 62

Figure 3.13: Travel time (sec) distribution on a link 12 at a give time 11AM ..... 63

Figure 3.14: Distribution of Speed on link 6 and link 10 ..... 63

Figure 3.15: Estimation for all trip points using crowdsourced current mean using rbf kernel ..... 64

Figure 3.16: Estimation error for all trips with crowdsourced current mean as input ..... 65

Figure 3.17: Effect of noise on input speed ..... 66

Figure 3.18: Estimation for last trip on VISSIM generated data ..... 68

Figure 3.19: Estimation error for last trip on VISSIM generated data ..... 68

Figure 3.20: Estimation using historic mean speed for last trip data from VISSIM ..... 69

Figure 3.21: Estimation error for last trip with historic mean of speed for VISSIM data ..... 70

Figure 3.22: Estimation for trip 34 using rbf kernel and true inputs from Morgantown dataset ..... 71

Figure 3.23: Estimation error for trip 34 from Morgantown ..... 71

# LIST OF TABLES

Table 2.1: List of Location providers ..... 29

Table 2.2: Sample GPS data coming from android device application ..... 34

Table 3.1: MSE variation with different values of C and γ on one link ..... 55

Table 3.2: Sample representation of Prediction model from LIBSVM SVR ..... 56

Table 3.3: Current mean speed for the data with missing values..... 56

Table 3.4: RMSE representation for different scenarios with Mobile century dataset ..... 65

Table 3.5: Effect of noise on predictor performance ..... 66

Table 3.6: Link node representation of VISSIM generated road network..... 67

Table 3.7: Performance measurement table for VISSIM generated data..... 69

Table 3.8: Performance measure for dataset from Morgantown ..... 72

# Chapter 1: Introduction

Travel time is considered a fundamental measure which provides traffic information. Based on this travel information, which is a key element, users can make decision whether to plan the trip to stay on the same route or choose different route. Travel time is a critical input for planning of the network, modeling, and decision-making applications such as traffic and performance monitoring, congestion management, travel demand modeling and forecasting, traffic simulation, air quality analysis, evaluation of travel demand, and traffic operations strategies.

Some real-time applications include Advanced Traveler Information Systems (ATIS), Route Guidance Systems (RGS) and STREAMS Integrated Intelligent Transport System etc., which are part of the Intelligent Transportation Systems (ITS). Thus, providing travelers with accurate and timely information play an important role in allowing them to make decisions regarding route selection, which is one of the important applications that use travel time information in recent times. To road system operator travel time information gives a basic knowledge to assess the operational management and planning of the network. Knowledge of travel time enables congestion management through operational strategies while real time information allows monitoring the evolution of congestion. From this context, travel time forecasting is a priority for road network managers/operators.

With importance in travel time estimation and increasing reliance on it, indicates a need to measure travel time accurately and cost effectively. The existing techniques as stated in [1] can be divided into two

categories namely direct methods and indirect methods. If travel times are directly collected from fields then it is referred as direct method. These methods as described in [2] include test vehicles, license plate matching, electronic Distance Measuring Instruments (DMI), video imaging, and probe vehicle techniques like Automatic Vehicle Identification (AVI) and Automatic Vehicle Location (AVL). On the other hand, travel time is estimated or calculated from other directly measured parameters like speed and this method is referred as Indirect method. Some examples of indirect sources of travel time are Inductance Loop Detectors (ILD), weigh-in-motion (WIM) stations, and aerial video.

Continuous probe vehicle techniques like AVI and AVL have less error, but they are more expensive. They often require new types of sensors as well as public participation, and hence, they are not widely deployed. Data collected from user needs to be kept private and it should be maintained in proper formatting. Because of privacy concerns video imaging techniques lead to public disapproval. Native methods like Test vehicle techniques like DMI, even though cost effective, are limited to a few measurements per day per personnel. They are only accurate as the driver's judgment of traffic conditions. Moreover, the test vehicle method and license plate matching are time consuming, labor intensive, and expensive for collecting large amounts of data. Other techniques like Weigh-in motion stations can collect data from only a selected group of vehicles, whereas aerial photography [1] is not very popular due to the prohibitive cost involved in using it. Loop detectors are widely used in North America which is in-fact a best source of traffic data over wide area. But the main drawback is due to high probability of error caused by prevalence of equipment malfunctions.

By rapid increase in number of cellular phones, decrease in telecommunication price, increase in cellular network broadcast, increase in accuracy of GPS data, finer levels of resolution, better accuracy in

measuring time and speed than the traditional methods led the shift in using GPS probes for travel time estimation. The data provided by the users gives valuable information about the traffic conditions and analyzing the data should be the primary motive. Proper models should be created which gives better travel time estimation. With advance in data analysis in the field of artificial intelligence, statistical methods play an important role in prediction. Our research poses two main challenges: 1) Designing a mobile application which can collect, store data; 2) create proper predictor models using machine learning techniques for better estimation. The empirical validation of the work is done by comparing the estimated values with the historic average of travel time.

## 1.1 Problem Statement

The problem of estimating travel time for a particular trip (Figure 1.1) could be broken down to the problem of estimating travel time over individual road links that comprise that trip Figure 1.2. In this case, the issue is to find out how long it takes a vehicle to travel through each link and then sum up all the travel times. The main objective of this thesis is to develop a methodology capable of estimating travel time using a database of sparsely sampled GPS data collected using crowd-sourcing of traffic data. To achieve this we first develop a GPS application on android mobile platform and collect the raw GPS data and store them in proper format. We next perform preprocessing in which the collected data is processed with techniques like map-matching, link formation, removing possible outliers. As stated earlier travel time estimation is an important traffic information passenger can use and for other network purposes.

A single road network can be represented as a group of links. A car traveling on this road from Origin to Destination traverses through all these links and the total travel time can be given by summing the travel times taken on individual links. So, we analyze the impact of parameters such as speed of entry ($V_{st}$) and time of day (ToD) at which the vehicle enters the particular link. In this link based network model,

determining the current speeds on these links as the vehicle traverses through them is a challenging task. We use crowdsourced data which is the concurrent data coming from the other vehicles traveling on that link for estimation of travel time.

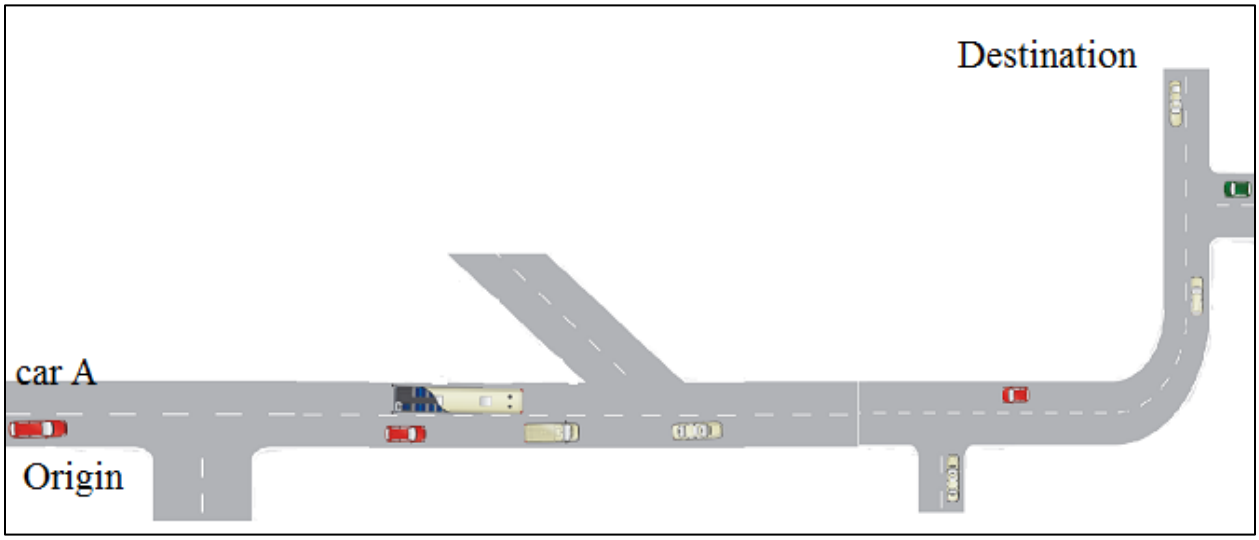


**Figure 1.1: Simple road network**

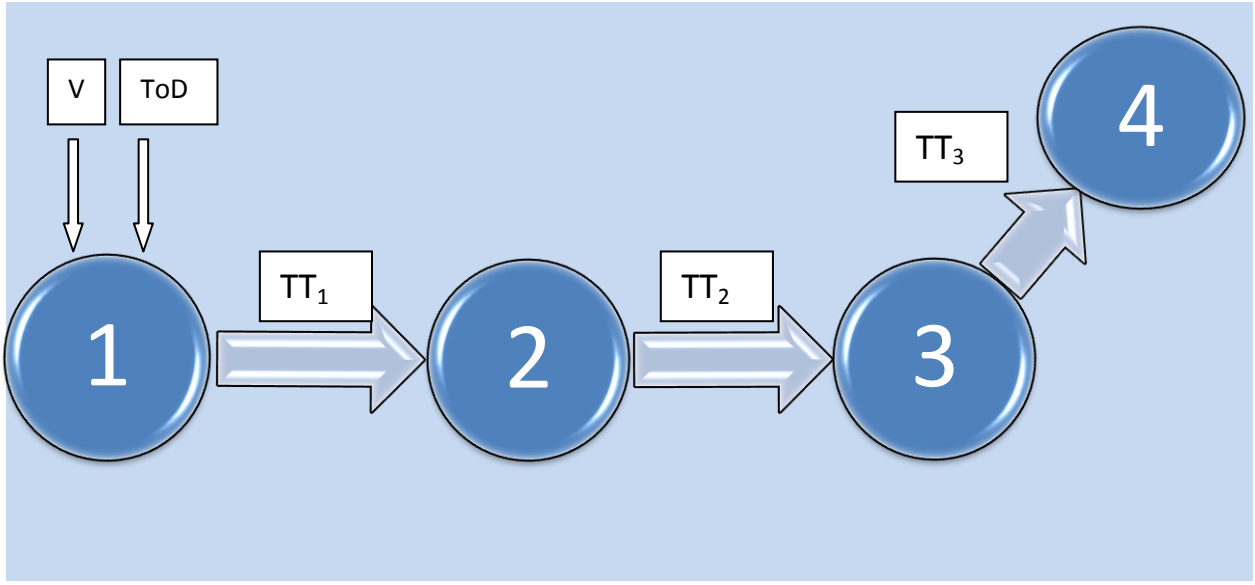


**Figure 1.2: Above road network representation using nodes and links**

## 1.2 Literature Review

### 1.2.1 Development of Traffic Measurement Systems

According to measuring site some important information in transportation planning and operating are volume, speed and density etc. Information gathered can be classified as either spot information or wide area information [3] and they can be further classified into instantaneous information and interval information. The authors in [3] stated traffic information produced through several years is developed in 3 different stages using technology. In first stage, clocker counted and then measured traffic parameters such as volume, speed, and travel time and so on. This stage accomplished by people manually measuring the traffic information. Thereby, the data may contain observation error.

The introduction of loop detector as a measuring instrument of traffic characteristics has become the most utilized sensor in the traffic management systems. There after various detectors are developed such as radar detector, ultrasonic detector, infrared detector, and video image processor etc. More effort has been laid on determining link travel speed using spot speed from various detectors. Although temporary loops are fashioned by taping wires to the roads, loops are typically permanent. An insulated electrically conducting loop is installed in the pavement. The principle involved [4] in this is the inductance loop behaves as a tuned electric circuit. When mass of a metal of a vehicle either passing or stopped, causes an inductance differential in a loop which can be detected with the appropriate equipment. These loops are costly to install and can be easily affected by environmental changes over time. Furthermore, the loop and detector combination can be out of tune causing system errors or variation of data from day to day. This

stage can be represented as second stage and can be classified as indirect method of travel time estimation. Radar sensors can also be used for speed collection but is cost inefficient as well as having limitations in high volume situations [5] in being able to record the speeds of specific vehicles in queues, and in bias towards lower speeds.

The last stage, computing through mobile devices, mobile data such as GPS or cellular phone's location can be acquired. With increase in satellite coverage and advance in cellular networks, the data gathered from mobile is more useful. Cellular phones have spread rapidly into each corner of the world making cellular phone's location information an important source in generating traffic information. This enabled researchers to have an interest in estimating travel time from the mobile data source such as cellular phone and GPS. The interesting point about mobile data is its large sample size. Point to point data collected helps the researchers to study and construct good models.

With the increasing availability of low-cost GPS technology, location-based services (LBS) applications are becoming available for the average consumer. While vehicular navigation systems have become extremely popular in recent years, the next evolution of the LBS environment is a transition to GPS-enabled mobile phones. In the recent times, Global Positioning System (GPS) has become a key application in many of the location-aware Smartphone. According to a RNCOS research report [6], "GPS Market Forecast to 2015", it is projected that shipment of GPS/LBS devices will grow at a CAGR of around 15% during 2012-2015 to reach around 1015 Million Units by 2015.

Some other application [6] reported in the literature where GPS can be used is in academic, business, and public media. More over increase in the use of GPS in aviation, maritime and waterways, highway and

construction, public transportation, railroads, communications, emergency response, surveying, weather, scientific, space, environmental protection, recreation, law enforcement and legal services, and agriculture and forestry have created a wide opportunity to the GPS market.

### 1.2.2 Location Detection and Traffic Sensing using GPS

The interesting thing about GPS is how it detects location with accuracy. A constellation of satellites continuously transmit radio signals to the users when requested. This constellation is maintained by Air Force to ensure the availability of at least 24 GPS satellites, 95% of the time. For the past several years, the Air Force has been flying 31 operational GPS satellites, plus 3-4 decommissioned satellites which are termed as "residuals" that can be reactivated if needed. The Global Positioning System (GPS) has been developed in order to allow accurate determination of geographical locations by military and civil users [7]. It is based on the use of satellites in Earth orbit that transmit information which allow to measure the distance between the satellites and the user. If the signals from three or more satellites are received, simple triangulation will make it possible to determine unambiguously the location of the user. The basic principle inherent in GPS is to determine with the best possible accuracy a point in space, as defined by three coordinates, here geographical latitude and longitude, as well as elevation above sea level. This is done by means of triangulation which is measurement of triangles.

The concept of using a probe vehicle for estimating travel time has been investigated for some time [8]. These studies have all shown that the concept is technically feasible. The authors in their work [8] used a methodology called Travel Time with GPS (TTG) which is based on GPS, for conducting travel time studies. TTG uses a general data model that includes a spatial model, GPS-based highway links, where links are defined as vector representations of directional centerlines delimited by physical discontinuities.

The authors in [9] presented, analyzed and compared two algorithms that use data from GPS equipped smartphones to estimate arterial traffic conditions. The two algorithms are based on Logistic Regression and Spatio-Temporal Auto Regressive Moving Average (STARMA), respectively. Each algorithm contains a learning component, which produces estimates of spatio-temporal parameters for describing interactions between the states of arterial links in the network.

The authors in [10] addressed the lack of sensor coverage across the arterial network thus motivates the use of GPS probe vehicle data for estimating traffic conditions. They proposed methods to extract time distributions from sparse, noisy GPS measurements from probe vehicles. A probabilistic model of travel times through the arterial network is presented along with expectation maximization (EM) algorithm for learning the parameters of this model. Their work is then extended to the case where the paths of the vehicles are unknown and we wish to infer the path taken which is based on identifying historical traffic patterns using Bayesian networks. The authors in [11] described a new approach to estimate travel time by using single probe vehicle based on the analysis of the speed-time profile. Their method describes associating the driving pattern with the difference between the travel time of probe vehicle and average travel time. Fuzzy set theory is utilized to describe the driver behavioral process during the measured journey with introduction of some defined variables as the fuzzy input.

The authors in [12] studied the operational parameters affecting the confidence of traffic monitoring system using cellular phone. This study includes parameters such as accuracy of the locations, the frequency with which location measurements are taken, and the number of locations available.

# 1.3 Travel Time Estimation Methods

## 1.3.1 Extrapolation Methods

Extrapolation technique can be defined as the one which estimates average travel time by assuming that spot speeds [13] can be applied for short roadway segments between detection devices. This method is simplistic but can be used for applications that do not require high levels of accuracy. Many extrapolation approaches have been implemented to estimate travel time with the data collected from loop detectors. Three approaches in specified below with the help of the Figure 1.2.

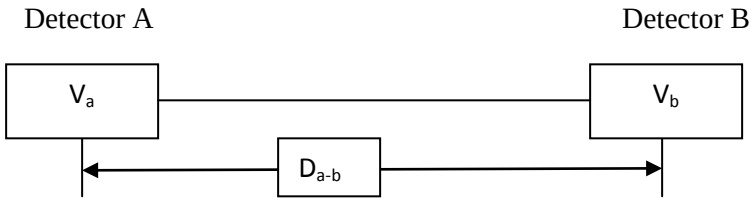


**Figure 1.3: Schematic diagram for illustrating extrapolation methods**

### 1.3.1.1 Half Distance Approach

In this approach the speed measured by a set of dual loop detectors is valid to the half distance on both sides.

$$T_{a-b} = \frac{1}{2}\left(\frac{D_{a-b}}{v_a} + \frac{D_{a-b}}{v_b}\right) \qquad (1.1)$$

### 1.3.1.2 Average speed Approach

The speed is assumed to be the average speeds measured by detectors 1 and 2.

$$T_{a-b} = \frac{D_{a-b}}{(v_a + v_b)/2} \qquad (1.2)$$

#### 1.3.1.3 Minimum speed Approach

The speed is assumed to be the minimum speeds measured by detectors 1 and 2.

$$T_{a-b} = \frac{D_{a-b}}{Min(v_a, v_b)} \quad (1.3)$$

where $v_a$and $v_b$ are the space mean speeds calculated based on the measurements at detector A and B respectively, $D_{a-b}$ is the distance between detectors A and detector B, and $T_{a-b}$ is the travel time from detector A to detector B.

### 1.3.2 Statistical Methods

Time series analysis reported in [14] states that any model for which exact calculation can be made possible is termed as deterministic. Probably no phenomenon is deterministic. In many problems we have to consider time dependent phenomenon, such as monthly sales of newsprints in which there are many unknown factors and for which it is not possible to write a deterministic model that allows exact calculation of future behavior of the phenomenon. But it may be possible to derive a model that can be used to calculate the probability of future value lying between two specified limits. Such a model is called a probability model or a stochastic model.

Kalman filtering can be applied in many research areas. Its basic function is to provide estimates of the current state of system and also to serve as a basis for predicting future values of prescribed variables at earlier time. The author in his research [15] conducted study on travel time prediction on the arterial roads. Based on the historic and real-time data, a recursive, discrete-time Kalman filter is used. Other

major methods include using Bayesian data augmentation [16] in which they proposed a method which can simultaneously estimate the path taken for each ambulance trip and the distribution of travel times on each road segment.

Two new statistical methods for estimating vehicle travel time distributions on a road network, using Global Positioning System (GPS) are proposed in which parameters needed for the models are generated using Markov chain Monte Carlo Methods [13]. Their models assume that travel times on links are independent. The other method is Whole Trip estimation in which they included parameters such as types of roads, time of the day etc. Their results showed that due to the assumption of independence between links travel times, the estimator leads to underestimation of the variability in the total route travel time and thus overly narrow interval estimates.

### 1.3.3 Theoretical Methods

Several theoretical models have been developed for the estimation of travel time from loop detector data based on traffic flow theory. Two methods which are widely used for calculation are queuing theory and shockwave analysis. These models take the advantage of capturing dynamic characteristics of the traffic flow. Traffic flows as stated in [17] fluctuate temporally and spatially. These fluctuations are due to the interactions among the drivers, the vehicles, and the road. A comparison between the delay that is calculated from the shockwaves and queuing theory is made in their report. Their approach is in using the principle of conservation of vehicles by comparing the inflow of a section during previous time period with its outflow during the current time period. Queuing theory as stated earlier, does not account for

effects due to traffic dynamics. In detail, queuing theory the difference in density between two points of interest is not considered.

The authors [18] in their research work proposed a methodology to estimate link travel times directly from the single-trap loop detector data with less dependency on the possibly flawed speed calculations. Flaw speeds are estimated using conventional methods by assuming the single-trap data with a common vehicle length assumption and to use a resulting identity relating density, flow, and speed. Their proposed methods are based on a simple stochastic model in which vehicles that arrive at an upstream point during a given interval of time have a common probability distribution of travel times to a downstream point.

The author in [19] considered link travel time to be more informative to the users than flow, velocity and occupancy measured at a point detector. The author utilized the linear approximation of the flow density relationship to estimate travel time from dual-loop detector data assuming constant shockwave speed. The accuracy of their results lends further evidence that the linear approximation of the flow density relationship is reasonable during congestion except at the transition periods from congested to un-congested conditions and vice versa. The other limitation is that their methodology assumes that all the signals travel through the entire freeway link and these assumptions fails when a queue partially fills a link. Most research work presented in traffic flow models are designed with input and output flows around the study road section which in most cases don’t monitor ramp flows and even they are not monitored by traffic detectors.

The methods described above are most for the data coming from Inductive loop detectors and for travel time estimation. Based on the mathematical methodology used, the rest of the prediction methods can be broadly classified into three categories: Regression methods, time series methods, and Machine Learning methods. Some methods use historical data in addition to real-time data.

### 1.3.4 Methods based on Historic data

Using average time and average speed: One would like to have the best information available on the travel times between two points in the network. Current systems find this shortest path based on hypothetical travel times calculated from distance and assumed speed. The authors in [20] pointed out that one way to approximate these current travel times is to use historical travel time information to forecast current road conditions[20]. Traffic patterns can be observed based on time, specifically time of day. Using the observations made in the past, predictions can be made. If similar observations are received in real-time, the accuracy of predictions for future travel times increases, especially in the short term. Therefore these models are reliable only when the traffic patterns in area of study are relatively stable. This approach is relatively easy to implement and computationally efficient.

$$T(t, \Delta) = \overline{T}(t) \qquad (1.4)$$

Where $\overline{T}(t)$, is average of past travel time at time t and $T(t, \Delta)$ is travel time at time step $\Delta$.

The authors in [21] conducted an operational test in the town of Blacksburg, Virginia to develop a rural traveler information system with Automatic Vehicle Location (AVL) techniques. Four algorithms were developed using GPS data of a bus for estimation of travel time. They represented the road network as nodes, links and thereby avoiding repeating paths (ordered links essentially convert a two-dimensional

route into a one-dimensional path). The input of the four algorithms is formed by taking an additional parameter from GPS bus location data, bus schedule table, delay and time check point respectively. Performance of the system varies [21] with addition of each parameter in input. Since all the methods proposed and implemented were for rural areas where traffic congestion is not seen, it cannot be applied to urban network. The authors in [22] proposed a method for travel time prediction which takes the basis of the current traffic situation in combination with historical data. They used statistical methods such as principle component analysis and windowed nearest neighbor which is an attempt to use information prior to time t. Nearest Neighbor aims to find that day in the past that is most similar to the present day. Their approach is computationally expensive.

The authors in [23] explored the travel time prediction problem with better accuracy by using travel time data collected directly through roadside terminals (RST) installed on New York State Thruway. The emphasis of their study is focused on using aggregated real-time and historic data for travel time prediction. Their results reveal that during peak hours, the historic path-based data used for travel-time prediction are better than link-based data due to smaller travel-time variance and larger sample size.

As stated in the previous sections, historical data based models require an extensive set of historical data [24] or real time information. Gathering information might be difficult when the traffic pattern varies significantly. So these methods are not applicable when in large cities when there is large deviation in variance which is caused due to congestion and free flow conditions. Similarity between the real time and historic traffic patterns plays an important role in deciding the accuracy of such systems.

### 1.3.5 Regression Models

Regression model is formed by forming a relation between dependent variable with independent variables. These models contradict historic prediction models even predicting under unstable traffic conditions. Any changes in independent input parameters are included into Regression models, affecting the dependent variable. The authors in [25] presented an approach to predicting travel time using linear regression, with the stepwise variable selection method using flow and occupancy data from single-loop detectors and historical travel time information. Their results are promising but it is applicable to shorter stretches of highways.

The authors [22] in their work compared travel time prediction using linear regression which is the linear combination of the current times and the historic means with Historic mean. Their further analysis by comparing the results with Principal Component analysis (PCA) and Nearest Neighbors showed that their method of linear regression performed well even when prediction goes to one hour. Researchers found that artificial neural network models performed considerably better than either historical data based models or multi linear regression models. In literature it was assumed that the ANN was able to identify the complex non-linear relationship between dependent variable travel time and the independent variables like speed, time of day and this led to superior results. Due to high inter correlation between input variables in transportation systems; the application of regression models is limited.

### 1.3.6 Time series methods

A time series is a collection of observations made sequentially in time [26]. Any quantity recorded either at discrete or continuous time leads to discrete time series and continuous time series respectively. As

stated in [26] a time series is a joint distribution of a sequence of random variables. The basic goal of time series analysis is to capture the underlying mechanism that generates the observed data and on a random noise. This can be therefore represented as a linear combination of previous time events. Their proposed model is ARIMA process and it constitutes of three parts, an autoregressive part (AR), a differencing part (I), and a moving average part (MA). Time series analysis has been used in many areas such as economics, finance, etc. The authors [26] adopted ARIMA model to study the modeling of arterial section travel times.

### 1.3.7 Machine Learning Models

In the last two decades Machine learning models which started evolving from the stage of neural networks to other methods such as support vector machines, decision trees etc. They have established themselves as serious contenders to classical statistical models in the area of forecasting. It all started in 1980 with the development of the neural network model. These models are classified as data driven models and are examples of non parametric models which use historical data to learn the stochastic dependency between past and future [27]. A similar study has been conducted by the authors in [28] and they conducted performance of ANN as a global model in chaotic time series predictions compared to the widely used local prediction models (the local averaging models and the local polynomial models).

#### 1.3.7.1 Artificial Neural Network methods

An artificial neural network is a massively parallel distributed processor made up of simple processing units with many interconnections. Artificial neural network does not need any priori knowledge of the actual physical processes and given that there is an exact relationship between input and output data. A neural network is characterized by its architecture. A typical ANN consists of a number of nodes that are

organized according to a particular arrangement. Based on the connection patterns Artificial Neural networks are grouped into two categories. Feed forward Neural Networks and Recurrent Networks (feedback networks).

A multilayer perceptron (MLP) is a feed-forward artificial neural network model that maps sets of input data on to appropriate outputs with no loops. It architecture is a multiple layers node arranged in a directed graph, with each layer fully connected to the next one. MLP utilizes a supervised learning technique called backpropagation for training the network. Unlike standard linear perceptron, MLP is a modification and it can distinguish data that are not linearly separable. However, for the ease in programming, a particular arrangement where neurons are arranged in layers that have a unidirectional connections.

Back-propagation algorithm has been found to provide excellent performance with regard to input-output function approximation and pattern recognition.

### 1.3.7.2 Support Vector Machines

ANN is one of the most popular methods in use for the prediction travel time. However, due to the shortcomings conventional ANNs including the difficulty in selecting the optimum number of hidden layers and hidden neurons. An alternative approach is for prediction is using support vector machines. This thesis explores an alternative technique, namely SVM for the prediction of travel time. Several studies [29] compared the performance of ANN and SVM in other applications. The authors in [30] reported that Traditional neural network approaches have suffered difficulties with generalization,

producing models that can overfit the data. This deficiency is due to the mainly because the ANN is parameter based algorithm.

Support Vector machines is a widely used technique for performing regression and classification. One widely used technique is in Optical Character Recognition (OCR) [31]. In our case we used for travel time prediction using regression functionality. The authors in [31] tried to use two different cost functions ε-insensitive loss function and Huber's robust loss function and present their performance. Their experiments showed that SVR methods particularly well if the data is sparse. The main difference between SVM and ANN is in the principle of risk minimization (RM).

Experiments conducted by some researchers [32] showed that linear SVR and non-linear SVR can have similar MSE values if the data set has many features. The bias term in SVR does not have effect on MSE of when the data is large and sparse data sets. The author in [1] compared travel time prediction models with other Machine Learning models like ANN and historic method by building a model that predicts the next 2-minute interval. Their results are promising for short time prediction. In finance [33], SVR plays equal role to minimizing conditional value at risk (CVaR) of the distribution of the l1-loss residuals. This allowed them to derive an upper bound of the generalization error. They applied the proposed method to portfolio selection based on index tracking and the results showed better theoretical performance. More details on how support vector machine works is presented in chapter 3.

### 1.3.8 Kalman Filtering Models

R.E. Kalman first published his famous paper in 1960 describing a recursive solution to the discrete-data linear filtering problem [34]. With advance in time and world of digital computing, kalman filter has its application in various fields such as area of autonomous or assisted navigation, computer vision etc. The authors in [35] discussed how to improve travel time estimates by incorporating data from a small sample of probe vehicles, and proposed an Adaptive Kalman Filter (AKF) based method that can dynamically estimate noise statistics of the system model by adapting to the real-time data. They used PARAMICS, a microscopic simulation model to generate recurrent and non-recurrent congestion data. They focused on the point stating that data coming from single loop or double detectors have noise under congested traffic conditions. With use of AKF they tried to fuses both point detection data and probe vehicle data. This addresses the solution to the noise covariance matrix. Their method failed to show better performance when the probe rate is increased.

The author in [15] used Test vehicle technique for travel time estimation which is a traditional way of collection of data using vehicle within which an observer records cumulative travel time at predefined checkpoints along a travel route. Using the GPS test vehicle technique, the total and section travel times were collected for a selected special event in their case for a graduation ceremony. The collected data is fed to kalman filter model which provides the estimates of the current state of the system. They modeled a recursive model using the results of the current step to obtain the results for the next step. To adjust the prediction error they included historic data with current prediction by which they showed improvement in prediction results.

## 1.4 Similar Projects

### 1.4.1 Mobile Century Project

In this section we focus on different projects which believed in using GPS data for travel time estimation. Some of the projects we choose to discuss here are Mobile Millennium project and project PUMAS. The objective is to collect technical elements and research methods which will give different technologies used to implement and applied methods.

The Mobile Century Project [36] is field experiment conducted in the San Francisco Bay Area, California and it aimed at assessing the feasibility of a traffic monitoring system using GPS-enabled mobile phones on freeway. This project was launched after a public-private partnership agreement involving the University of California Berkeley, NAVTEQ (Digital map producer) Caltrans (The California department of transportation) and Nokia research center. The experiment is conducted using Nokia N95 which is capable of transmitting a time stamped geo-position every 3s. Using the time and position instantaneous speed is calculated. The sampling strategy proposed in their work is based on the use of Virtual Trip Lines (VTLs), which are spatial triggers for phones to transmit updates, and provides enough data for traffic monitoring purposes while managing the privacy of participants. Their proposed system consists of four layers. The first layer is GPS-enabled smartphones in vehicles, second is a cellular network operator (network operator), third cellular phone data aggregation and finally traffic estimation (Nokia/Berkeley), and information dissemination (Info Consumers). Each mobile device is capable of downloading and caching the VTL from VTL server. Whenever the vehicle passes, using the current and previous positions it makes checks if it intersects a trip line and if so it creates an encrypted VTL update. The update comprises of a speed reading, time-stamp, trip line ID, and the direction of the trip line crossing. Safety

and reliability measures are taken not to give position and speed information of user for privacy. This experiment is nicknamed Mobile Century and was first conducted on February 8, 2008 which involved 100 vehicles carrying GPS-enabled Nokia N95 phones.

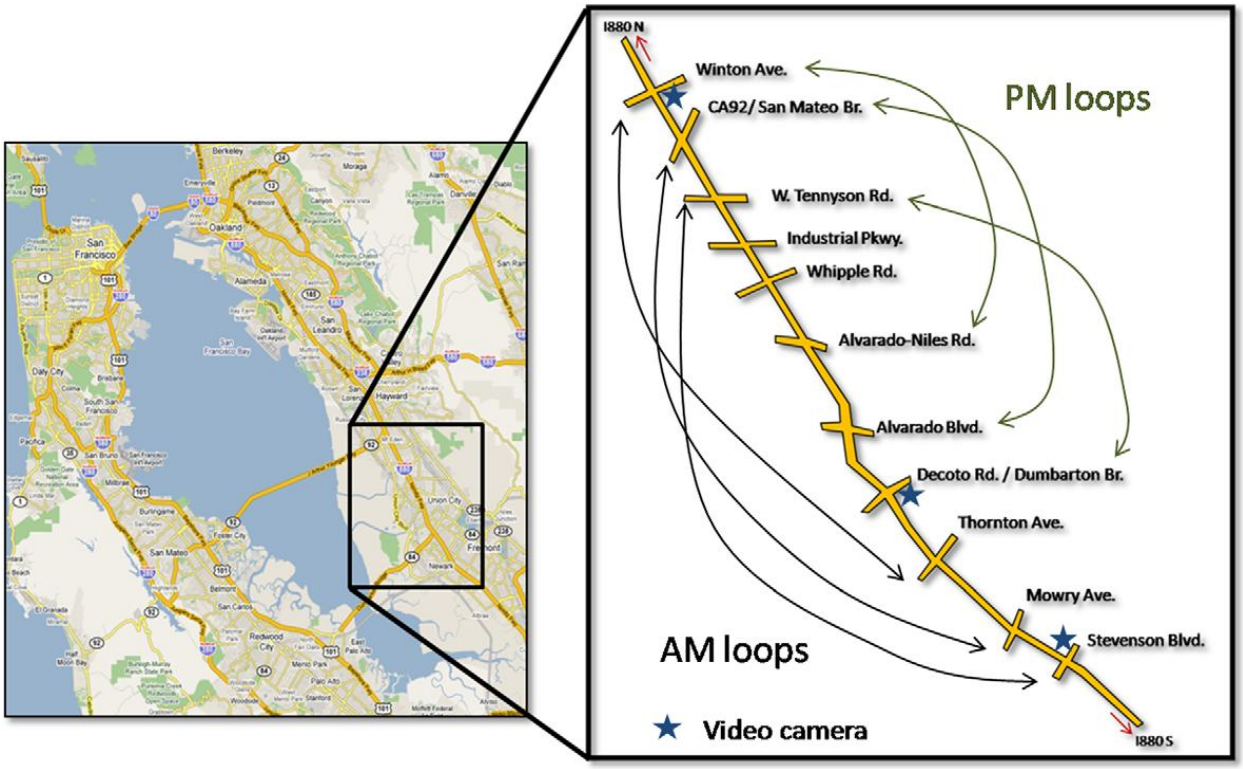


**Figure 1.4: Stretch of freeway I-880 CA, used in the Mobile Century experiment [36]**

The vehicles traversed 6 – 8 miles for each trip on freeway I-880 near Union City in the San Francisco Bay area, California. 45 VTLS were deployed on the freeway and trips were conducted starting from 10AM to 6PM. Their goal is to assess the feasibility of a traffic monitoring system based on GPS enabled mobile phones, evaluate velocity measurements collected by VTLs and loop detectors, as well as the computation of the penetration rate achieved during the day. Their results [36] show that travel times computed with the VTL velocity field are in better agreement with real travel times when compared to the

travel time calculated through loop detector. This suggests that the VTL velocity field is more likely to be closer to the actual velocity experienced by the vehicles, and therefore more accurate, than the loop detector velocity field. That is the project has achieved its goal by showing that low proportion of equipped vehicles can often provide more accurate measurements of velocity than loop detectors.

### 1.4.2 The PUMAS Project

This is another research project conducted to demonstrate the GPS system and travel time estimation. PUMAS [37] stand for urban platform for advanced and sustainable mobility (Plateforme Urbaine de Mobilité Avancée et Soutenable). PUMAS is funded by the state (DGCIS- General Directorate for Competitiveness, Industry and Services), Ile de France and Haute-Normandie Region FEDER (European Union). The project aims to develop mobility platform software, and to evaluate the territory of the Community of Agglomeration of Rouen Elbeuf Austreberthe (CREA). Its main aim is to provide better information on traffic conditions.

The output of the PUMAS system will be a digital map representing the urban road network. Their work collected 4GB of data by driving 1400 Km on city of Rouen, France for 3 days. They used Monte Carlo Method for travel time estimation.

## 1.5 Thesis Structure

This section describes the contents of each chapter. The thesis contains 4 chapters including the introductory chapter.

Chapter2 gives a description and implementation of GPS sensing system on android mobile and the server handling model proposed. The flow of data in server client model is presented.

Chapter 3 gives a clear description of tools used in preprocessing step. We create links for the given road network then deduce the speeds and travel times on the links. The structures of historical speed, crowd-sourced half hour mean speed, and crowd-sourced current mean speed are studied and later use them for estimation and for empirical comparisons. The developed links are trained, validated to create predictor models and further tested on the three available datasets with Support vector regression technique using software platform MATLAB LIBSVM toolbox. We finally compare these results with historic mean of data.

Chapter 4 concludes the thesis and gives some recommendations for future work.

# Chapter 2: Crowdsourcing of Traffic Information

The need traffic information has triggered the deployment of large scale dedicated monitoring infrastructure systems, which mainly consist in the use of inductive loop detectors and video cameras. But maintaining this architecture is cost consuming process. So some alternative traffic sensing infrastructure, such as on-board electronic devices have been proposed which are cost-effective to collect traffic data, leveraging existing communication infrastructure such as the cellular phone network [36]. Many factors such as development in telecommunications in providing accurate position, velocity and rapid supply of cellular phones with GPS-enabled, made the existing system of travel time prediction exploited. These can be used to obtain traffic information. On other hand, the concerns such as communication load, headset energy consumption and privacy should be considered [36]. One approach to do this is by sampling strategy and the two sampling strategies are:

Temporal sampling: Vehicles equipped with the GPS sensor device report their information (position, velocity, etc.) at specific time intervals T, regardless of their positions.

Spatial sampling: Vehicles equipped with GPS sensor device report their information (time, velocity, etc.) as they cross some spatially defined sampling points [VTLS]. This strategy is similar to the one used by

inductive loop detectors, RFID transponders or license plate readers, in which data are obtained at fixed locations.

This sampling technique takes the advantage that the phone is forced to send data from a given location of interest.

For our project we tried to have our own mobile application which makes it easier to add powerful mapping capabilities to your application. For maps we used Google Maps API in our application. Google Maps is a web mapping service application and technology provided by Google, that powers many map-based services, including the Google Maps website, Google Ride Finder, Google Transit, and maps embedded on third-party websites via the Google Maps API. It offers street maps and a route planner for traveling by foot, car, bike (beta), or with public transportation. Google Maps satellite images are not updated in real time, however, Google adds data to their Primary Database on a regular basis and most of the images are no more than 3 years old.

## 2.1 Client Architecture

Data collection can be done in several ways using several devices. With increase in number of android mobile users and GPS sensing technology embedded into android mobile devices enables us to use android mobile device as client. Android software development enables us to create applications that can access different components of android interface. Android software development is the process by which new applications are created for the Android operating system. These applications are usually developed using the Java programming language and the Android Software Development Kit (SDK), but other development tools such as iOS are available.

Android application development can be broken up into three phases which are done concurrently.

i. Interface development

ii. Coding

iii. Testing

Here we are going to show how interface is created, how it looks and how it works. Google maps API can be downloaded from Google developer location [38]. Once you've installed the Google APIs add-on, you can add Maps capabilities to any existing or new Android project. To give our application access to the Maps library, we need to set the project's properties, so that the build tools can locate the Maps library in the Google APIs add-on. We are using Eclipse with Android Development Toolkit for application development.

There three key concepts described by the developers of android application developers about building applications [39] are

- The core framework components that define an application.
- The manifest file in which you declare components and required device features for your application.
- Resources that are separate from the application code and allow your application to gracefully optimize its behavior for a variety of device configurations.

### 2.1.1 Application Components

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter your application. There are four different types of application components [39].

- Activity

An *activity* represents the visual representation of an Android application. An Android application can have several activities.

- Services

A *service* is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface.

- Content providers

A *content provider* manages a shared set of application data. The data can be stored in the file system, an SQLite database, on the web, or any other persistent storage location your application can access. Through the content provider, other applications can query or even modify the data.

- Broadcast receivers

A *broadcast receiver* is a component that responds to system-wide broadcast announcements.

For our application we start by creating main activity which is the user interface for the user to login to the application. The maps should be referenced from the Maps Library through the Application's Manifest File. Application Manifest file is the file where we can set different permissions, settings and components for the application to access the android OS. For further reference please see Appendix A.1. To enable the display of Maps data we should sign in our application with the proper certificate. The certificate you use to sign your application must match the certificate that is associated with the API Key in your MapView objects.

### 2.1.2 MainActivity

In an application, MainActivity refers to users login screen for the first time into the application. It has four states. When the application is opened it is running state and it goes to *pause* state when it loses focus. But it still holds its state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere. The final state is when the activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. If the user needs the display, it must be completely restarted by the user to restore to its previous state. An example of how our layout file looks like is represented in Appendix A.1.

### 2.1.3 Menu Items

Android SDK offers user to create their own menu options list if needed. The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a global impact on the app, such as "Search," "Compose email," and "Settings." For our application we have provided with six different menu items. “Traffic layer”, “Satellite View”, “Street View”, “Location” etc. These services are provided by google server. When the user clicks on the options, it communication is made between Google server and get backs the layer with information requested on to the activity.

### 2.1.4 Android Location Provider

Our Application should be able to determine the current geolocation. Most Android devices allow determining the current geolocation. This can be done via a GPS (Global Positioning System) module, via cell tower triangulation or via wifi networks. Android contains the android.location package which provides the API to determine the current geo position.

### 2.1.4.1 LocationManager

The LocationManager class provides access to the Android location service. This service allows the application to obtain periodic updates of the device geographical location.

### 2.1.4.2. LocationProvider

The LocationProvider class is the superclass of the different location providers which deliver the information about the current location. The Android device might have several LocationProvider available and you can select which one you want to use. In most cases you have the following LocationProvider available.

**Table 2.1:** List of Location providers

| LocationProvider | Description |
|---|---|
| **Network** | Uses the mobile network or Wi-fi |
| **GPS** | Uses GPS receiver from the device |
| **Passive** | This provides coarse location even though GPS is not enabled |

### 2.1.4.3 Getting Location Updates

The location updates can be stored in the GPS device or can be transmitted to be base station. For our project we tried to save the location updates on the device in a text file. Each location update has the

following parameters in it. The Location class in android has capability to return bearing, altitude, latitude, longitude, network provider, speed, time etc. The time is returned in Unix format. We convert the unix timestamp to the local time zone. For our calculations we use latitude, longitude, speed and timestamp. These parameters are returned by the device when it meets certain criteria which we added to save battery life and communication usage. Android has two criteria in which the device can transmit the data. The LocationManager class provides access to the system location services by calling LocationListener callbacks. These services allow applications to obtain periodic updates of the device's geographical location, or to fire an application-specified Intent when the device enters the proximity of a given geographical location. By calling the method requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener) means it gets called when it meets the two criteria as mentioned above.

- minTime : minimum time interval between location updates, in milliseconds
- minDistance : minimum distance between location updates, in meters

The user has ability to specify these two parameters which enable the device to transmit the data when it meets the criteria. When minTime parameter is specified it means the device waits till that time period before it requests for next update and minDistance when specified means the device waits to request for next update before it travels minimum distance. When both parameters are specified, it takes the both the parameters into account. That is, it has to satisfy both conditions to get the next update. Choosing a sensible value for these parameters is important to conserve the battery of the device. For our research we used < 0.1 Hz, 0 m> to receive updates.

## 2.2 Server Architecture

Our research adds one more feature to the application development i.e., communicating to the server. To establish this, we build a local Apache web server [apache] which runs on a local machine. A web server program is software that runs on the web site hosting Server computer. It responds to the requests coming from various clients. Once the web server is build and is running we need to create web applications. This can be done using many scripting languages. We used PHP for server side scripting. For the backend data storage, we installed MySQL which is a database management system.

In the initial stage, when a mobile senses a GPS signal using either network location provider or wireless provider or through GPS sensor, the data is transmitted to the web server using http request in JSON format. On the server end, using PHP scripting we parse the incoming data and store them into LOG data table. When the user requests an update to the map, the floating GPS data is map-matched to stored GPS logs on the phone and traffic layer is plotted. The architecture is shown in Figure 2.1.

## 2.3 Data Preprocessing

In literature there are two kinds of data mentioned. The first is data coming from road sensors and the later is data coming by probing vehicles. We follow the second method in our research vehicle probing by using GPS mobile device as mentioned in previous sections. One dataset is collected on Morgantown Beechurst ave which is a floating car data. Floating data represents the GPS data collected for a particular location doesn’t necessary have the same latitude longitude. This depends on the accuracy of GPS. The data collected in general is a low frequency data usually in a range of 0.1 to 0.5 Hz. So the data coming is a huge set of data, which makes it difficult for computation unless preprocessing is performed. The

preprocessing steps have various subcomponents that need significant research work, development and integration to be done. The steps include representing urban network by building node graph model. This step is important for points of interest which is crucial in travel time estimation and also for result analysis. The other steps include map matching and final path reconstruction.

In our work as stated we tried to work with three different kinds of data. Each data has its own way of preprocessing since they are collected with different frequencies and has different number of nodes and links connecting them. The first data set is from Morgantown Beechurst Ave which we started collected them starting on [03/12/13 to 03/15/13 and 03/18/13 to 03/19/13] and the trips start from 8:00 AM to 11:00 PM. There are total of 34 complete trips collected for 6 days. The other trips information has lost data so we did not use those trips data for our research. First using Google Maps we have created a standard topology of road network in which the trips are conducted.

We reduced this to a model with 10 links with 11 nodes. It is technically represented as directed graph $G = (V, E)$ where $V$ is vertices or nodes and $E$ is the edges connecting them. Each link thereby has several points but the points of interest comes from the starting node and ending node for each link which can tell the travel time taken to traverse the link. Now the other trips data has to be matched to these set of nodes for travel time estimation and this process is referred to as Map matching. Our problem of travel time estimation can be further reduced into a simple problem from the above strategy. Suppose a car is moving from link 1 at time $t_1$ with initial velocity $v_1$, we would like to know how much time it takes to cross link 7. In that case we use the prediction models for every link and estimate the values. Finally summing the estimates for all the links gives the time taken to reach the destination.

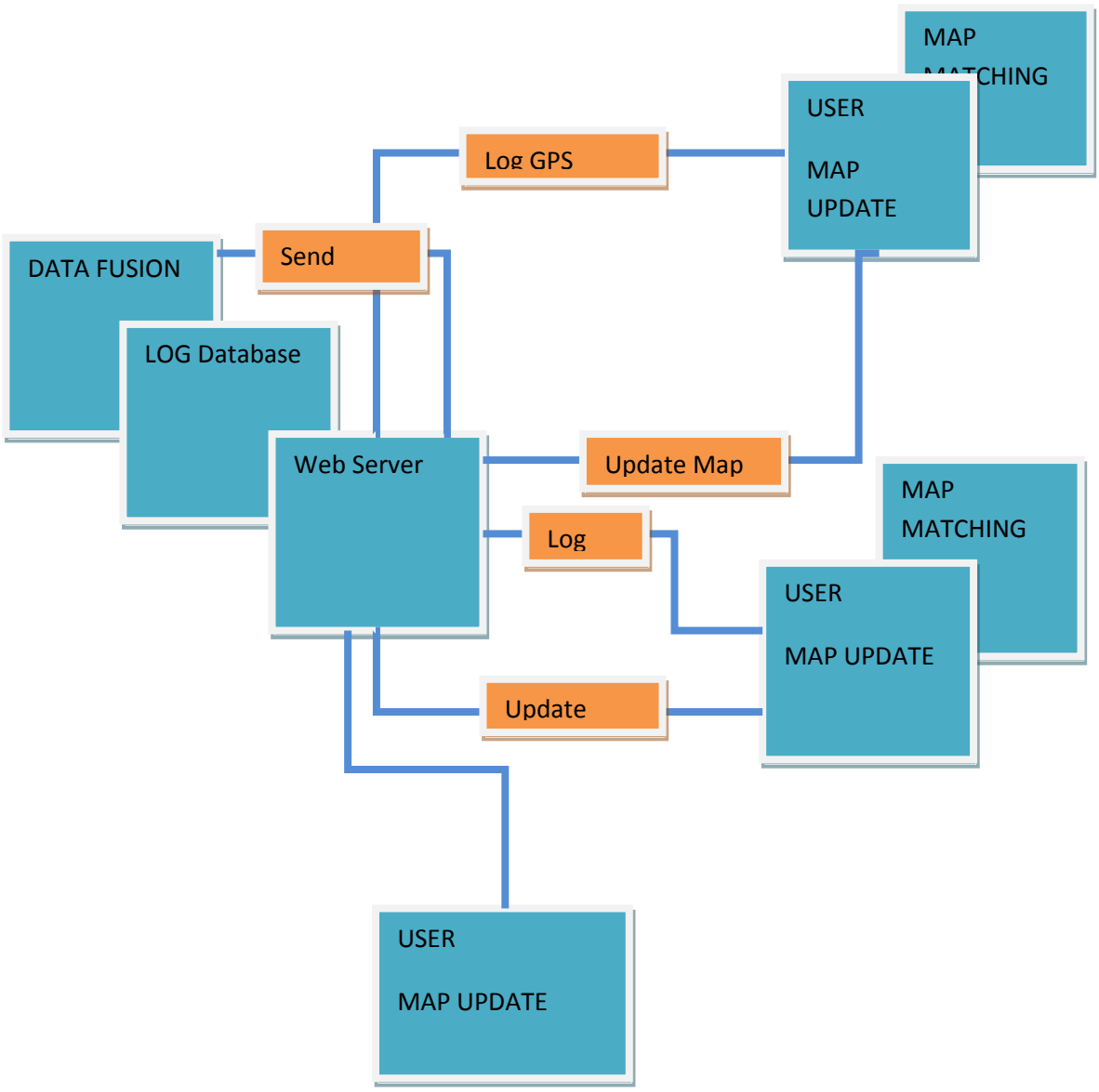


**Figure 2.1: Proposed Server client architecture**

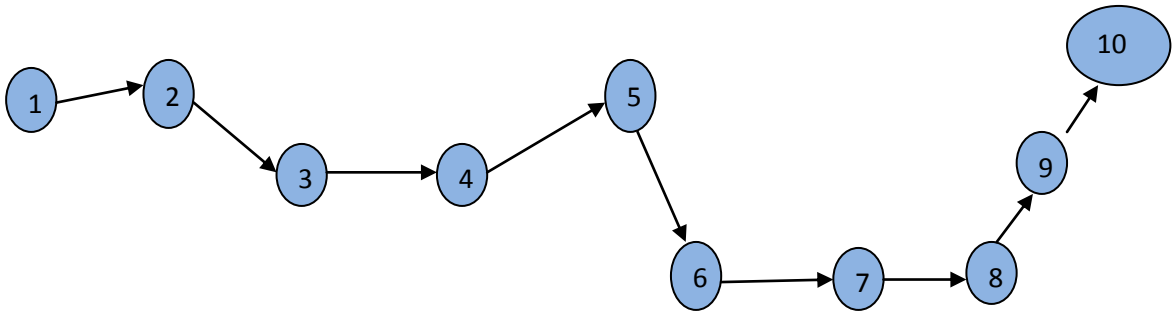


**Figure 2.2: Simple road network representation using directed graph**

### 2.3.1 Map Matching

Map Matching is a very important process and there is a lot of research in this field alone [40]. Different researchers implement this using different ways but in general this can be coined into three classes namely local or incremental method, global method and statistical method. The first method tells that it tries to match local geometries of roads. Global method tries to match entire trajectory with the road network. Some Statistical methods use Bayesian classifier which incorporates a Hidden Markov Model to model the topological constraints of the road network. Every method has its own time complexity, which is how much time it takes to perform a match. Our GPS log, which consists of lots of GPS traces, looks like as shown the table below.

**Table 2.2:** Sample GPS data coming from android device application

| Trace ID | Device ID | Latitude | Longitude | TIMESTAMP |
|---|---|---|---|---|
| **1210** | 3589xxxxxxxxx | 39.647017 | -79.957865 | 2012-10-27 23:55:17 |

| | | | | |
|---|---|---|---|---|
| **1211** | 3589xxxxxxxxx | 39.647026 | -79.957890 | 2012-10-27 23:55:17 |
| **1212** | 3589xxxxxxxxx | 39.646987 | -79.957874 | 2012-10-28 00:09:25 |
| **1213** | 3589xxxxxxxxx | 39.646992 | -79.957844 | 2012-10-28 00:09:25 |
| **1214** | 3589xxxxxxxxx | 39.646992 | -79.957844 | 2012-10-28 00:48:59 |

A trip can be called a trajectory T with points/nodes $e_1$, $e_2$ …..$e_n$ with a sampling rate 0.1 Hz. This trajectory has to be matched to road segment we have created above to find the list of nodes which is the problem statement for map matching. The data transmitted or stored by the device is in raw format. Using MATLAB import functionality we first import the data into a matrix. Our approach has three major components namely Distance Computation, Spatial Analysis and Fitting Missing data. In the first segment we try to find the distance from all the points on the trajectory to all the points to the road segment. The results are stored in a matrix. In the second segment we try to find the point $e_i$ on the trajectory which is closest to point $v_i$ on the road network which also should satisfy the minimum distance criteria. The minimum distance criteria we assumed is, a circle of radius 0.01 mile around the point $v_i$. If there are multiple points falling in that region the closest point is chosen.

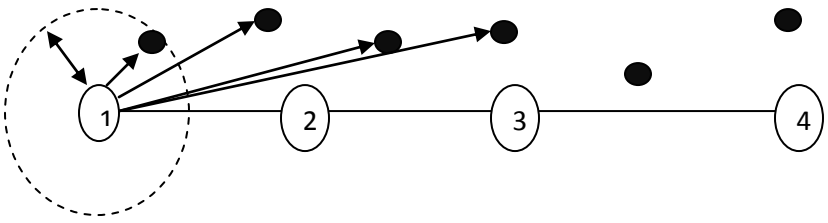


**Figure 2.3: Simple Matching Process**

In the final component the data is fitted with missing values. The trajectory reconstruction process losses some matching points on the road network G. These points are needed so as to construct a complete trajectory similar to road network. For doing this we followed a moving window technique.

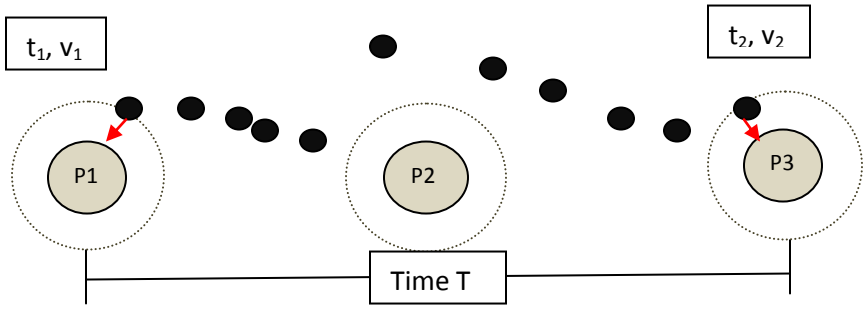


**Figure 2.4: Systematic representation of map matching for road network**

Assume that n points are missing between $P_i$ and $P_j$. These vertices can be gathered from road network but the information such as time and speed at these vertices are missing. These data can be assumed null or can be assigned some random value. In our research we proposed moving window technique which is finding the missing vertices/nodes get assigned time with the total time taken between $P_i$ and $P_j$ divided by n+1 and the speed at each vertex is assumed remain constant from the previous known vertex. For the Figure 2.4, t at point P2 is calculated using

$$t_n = \frac{(t_j - t_i)}{n+1} * n + t_i \qquad (2.1)$$

Where n = 1, 2, 3……. Number of missing points

The performance of the application to communicate to server depends on the network characteristics. One of such important characteristic is channel load. To study how our application affects the channel, we can calculate the channel load as shown below.

Suppose the Sampling rate R for packet transmission = 1Hz

The size of the message L = 5*4*8 + overload = 200 Bytes

Number of cars in a cell = N

So the required throughput S = N*L*R

If N = 1000, S= 200 KB/s or 1.6 Mbps, which shows that most of the 4G networks supports the chosen sampling rate but the worst case could be when there are more than 10000 cars on a link. Our research approach is good enough to build a road network from a trajectory when time complexity is not considered and the road networks are not crowded.

# Chapter 3: Travel Time Estimation using crowdsourced data

## 3.1 Overview

SVMs have been successfully applied [41] to a number of applications ranging from particle identification, face identification, text classification, bioinformatics and database marketing. The approach is systematic and is motivated by statistical learning theory. Due to the sparse distribution of data, traditional Neural Networks suffered from generalization which thereby produced models that overfit the data [30]. This is caused due to the optimization algorithms used for parameter selection and the statistical measures used to select the best model. The foundations of Support Vector Machines (SVM) have been developed by Vapnik [26] and are gaining popularity due to many attractive features, and promising empirical performance [30]. The author in [30] showed that basic neural network structure which consists of Empirical Risk Minimization (ERM) principle is inferior to Structural Risk Minimization (SRM) by which SVM is formulated. SRM minimizes the upper bound on the expected risk which is opposite to ERM which minimizes the error on the training data. This motivation, considering theoretical bounds on the generalization error led SVMs with a greater ability to generalize. SVMs are applied both in classification and regression problems and the terms Support Vector Classification (SVC) and support vector regression (SVR) will be used.

## 3.2 SVM Architecture

To introduce SVMs lets start by taking simple binary classification in which the goal is to form a function from the dataset which can separate the two classes. That is the classifier should generalize well. SVM attempts to place the margin in such a way that it maximizes the margin which means maximizing the distance between it and the nearest data point of each class [30]. This maximal margin or the linear classifier is termed as the optimal separating hyper plane as shown in figure [3.1]. The generalization bounds have two important features.

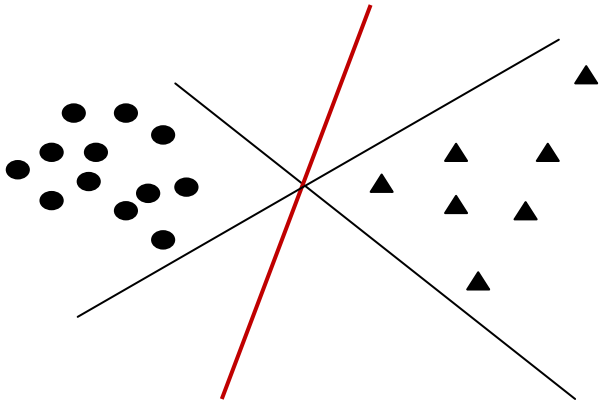

**Figure 3.1: Optimal separating plane for data**

Let data points for binary classification be,

$$D = \{(x^1, y^1) \dots\dots\dots (x^l, y^l)\}, x \in \mathbb{R}^n, y \in \{-1,1\} \qquad (3.1)$$

As explained previously, there are a number of hyperplanes that can separate these two sets of data and the problem is to find out the one with the largest margin. The SV classifiers are based on the class of hyperplanes called boundary lines,

$$(w.x) + b = 0, w \in \mathbb{R}^n, y \in \{-1,1\} \qquad (3.2)$$

where,

w = the boundary,

x = the input vector, and

b = the scalar threshold.

The data will be correctly classified if the data satisfies the condition $y_i * (w.x_i + b) > 0 \; \forall i$. To remove redundancy, the hyperplane is considered in canonical form defined by a unique pair of values $(w, b)$ at the margins satisfying the condition for points on one side by $(w.x + b) = 1$ (3.3), and the points on other side by $(w.x + b) = -1$ (3.4).

The quantities w and b will be scaled for this to be true, and therefore the support vectors correspond to the extremities of the data. Thus, the decision function that can be used to classify the data is:

$$y = sign\big((w.x) + b\big) \qquad (3.5)$$

There can be many possible hyperplanes that can separate the training data into the two classes. To find the optimal separating hyperplane we need to maximize the margin [30]. This margin, ρ is the sum of the absolute distance between the hyperplane and the closest training data points in each class.

In Euclidean geometry distance $d(w, b; x)$ of a point $x$ , from the hyperplane $(w, b)$ is:

$$d(w, b; x) = \frac{|(w.x_i) + b|}{\|w\|} \qquad (3.6)$$

Thus, the sum of the absolute distance between the hyperplane and the closest training data points in each class i and j, ρ is calculated as given in Equation 3.7.

$$\rho = min\frac{|(w.x_i) + b|}{\|w\|} + min\frac{\left|(w.x_j) + b\right|}{\|w\|} = \frac{2}{\|w\|} \qquad (3.7)$$

Thus the optimal canonical hyperplane is the one that maximizes the above margin and to do that we need to minimize [26]

$$\phi(w) = \frac{1}{2}\|w\|^2 \qquad (3.8)$$

subject to the constraints $y_i * (w.x_i + b) \geq 1 \; \forall i$

These training patterns, called support vectors, carry all relevant information about the classification problem. The learning task can be reduced to minimization of the primary lagrangian [26]. More details on solving this optimization problem are provided in Appendix A.2.

## 3.3 Support Vector Regression

### 3.3.1 Linear Regression

Consider the problem of approximating the set of data,

$$D = \{(x^1, y^1) \dots\dots\dots (x^l, y^l)\}, x \in \mathbb{R}^n, y \in \mathbb{R} \quad (3.9)$$

with a linear function,

$$f(x) = \langle w, x \rangle + b \quad (3.10)$$

the optimal regression function is given by the minimum of the functional,

$$\phi(w, \xi) = \frac{1}{2}\|w\|^2 + C\sum_i (\xi_i^- + \xi_i^+) \quad (3.11)$$

where C is a pre-specified value, and −, + are slack variables representing upper and lower constraints on the outputs of the system. SVMs can be applied to regression problems introducing a loss function [1]. The loss function should be modified to have a distance measure. The author represented loss functions described below in his work [30] and they are a) Quadratic b) Laplace c) Huber d) ε-insensitive.

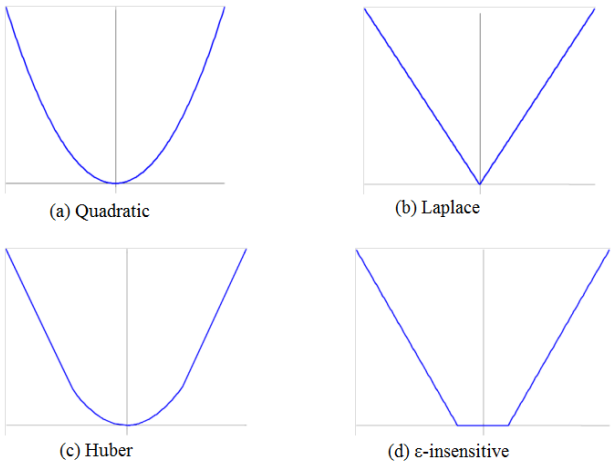


**Figure 3.2: Loss Functions**

The first three lost functions do not produce any sparseness in the support vectors. Vapnik proposed the loss function as shown in Figure 3.2 that enables to obtain a sparse set of support vectors [31].

For non-linear regression, the loss function employed as stated is $\varepsilon$-insensitive loss function that has form

$$L_\epsilon(y) = \begin{cases} 0, for\ \ |f(x) - y| x < \varepsilon \\ |f(x) - y| - \varepsilon,\ otherwise \end{cases} \qquad (3.12)$$

Intuitively speaking, No penalty should be imposed if the absolute residual is off target by epsilon or less. However, if the opposite is true, that is $|y_i - f(x)| - \varepsilon > 0$, then a certain amount of loss should be associated with the estimate. This loss rises linearly with the absolute difference between y and f(x) above ε-[SVR Basics].

The solution to above is given by

$$\max_{\alpha,\alpha^*} W(\alpha, \alpha^*) = \max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)\langle x_i, x_j\rangle + \sum_{i=1}^{l}\alpha_i(y_i - \varepsilon) - \alpha_i^*(y_i + \varepsilon) \qquad (3.13)$$

$$\bar{\alpha}, \bar{\alpha}^* = arg\min_{\alpha,\alpha^*}\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)\langle x_i, x_j\rangle - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)y_i + \sum_{i=1}^{l}(\alpha_i + \alpha_i^*)\varepsilon \qquad (3.14)$$

with constraints

$$0 \le \alpha_i, \alpha_j{}^* \le C, i = 1, \dots\dots, l$$

$$\sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \qquad (3.15)$$

Solving Equation 3.13 with constraints Equation 3.15 determines the Lagrange multipliers, $\alpha, \alpha^*$ and the regression function is given by Equation 3.10, where

$$\bar{w} = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) x_i$$

$$\bar{b} = -\frac{1}{2} \langle \bar{w}, (x_r, x_s) \rangle \qquad (3.16)$$

The Karush-Kuhn-Tucker (KKT) [30] conditions that are satisfied by the solution are,

$$\bar{\alpha}_i \bar{\alpha}_i^* = 0, i = 1, \ldots \ldots, l \qquad (3.17)$$

Therefore the support vectors are points where exactly one of the Lagrange multipliers is greater than zero. When $\varepsilon = 0$, we get the $L_1$ loss function and the optimization problem is simplified,

$$-C \leq \beta_i \leq C, i = 1, \ldots \ldots, l$$

$$\sum_{i=1}^{l} \beta_i = 0 \qquad (3.18)$$

and the regression function is given by Equation 3.10, where

$$\bar{w} = \sum_{i=1}^{l} \beta_i x_i$$

$$\bar{b} = -\frac{1}{2} \langle \bar{w}, (x_r + x_s) \rangle \qquad (3.19)$$

### 3.3.2 Non-Linear Regression

A non-linear mapping can be used to perform mapping the data from original space into a high dimensional feature space where linear regression is performed. The kernel approach, which is dot

product of two vectors, is again employed to address the curse of dimensionality. The non-linear SVR solution, using an ε-insensitive loss function, Figure 3.2(d), is given by,

$$\max_{\alpha,\alpha^*} W(\alpha,\alpha^*) = \max_{\alpha,\alpha^*} \sum_{i=1}^{l} \alpha_i^*(y_i-\varepsilon) - \alpha_i(y_i+\varepsilon) - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\alpha_i^*-\alpha_i)\left(\alpha_j^*-\alpha_j\right)K\left(x_i,x_j\right) \quad (3.20)$$

with constraints,

$$0 \le \alpha_i, \alpha_j{}^* \le C, i = 1, \ldots\ldots, l \quad (3.21)$$

Solving Equation 3.20 with constraints Equation 3.21 determines the Lagrange multipliers,

$\alpha_i, \alpha_i{}^*$ , and the regression function is given by,

$$f(x) = \sum_{SVs} (\bar{\alpha}_i - \bar{\alpha}_i^*) K(x_i, x) + \bar{b} \quad (3.22)$$

where

$$\langle \bar{w}, x \rangle = \sum_{i=1}^{l} (\alpha_i - \alpha_i^*)\, K\left(x_i, x_j\right)$$

$$\bar{b} = -\frac{1}{2}\sum_{i=1}^{l} (\alpha_i - \alpha_i^*)\left(K\left(x_i, x_j\right) + K(x_i, x_s)\right) \quad (3.23)$$

The insensitive loss function is attractive because unlike the quadratic and Huber cost functions, where all the data points will be support vectors, the SV solution can be sparse.

## 3.4 SVR Development using LIBSVM

LIBSVM is a library for Support Vector Machines (SVMs) [42] which has developed starting in the year 2000. The authors' goal is to help users to easily apply SVM to their applications. LIBSVM has gained wide popularity in machine learning and many other areas especially in biometrics, computer vision, natural language processing, traffic analysis, finance.

Using LIBSVM is done in two steps. Training a data set to obtain a model and second, using the model to predict information of a testing data set. The LIBSVM package is structured as follows [42].

1. Main directory: core C/C++ programs and sample data. In particular, the file svm.cpp implements training and testing algorithms, where details are described in this article.
2. The tool sub-directory: this sub-directory includes tools for checking data format and for selecting SVM parameters.
3. Other sub-directories contain pre-built binary les and interfaces to other languages/software.

LIBSVM supports various SVM formulations for classification, regression and distribution estimation. C-support vector classification (C-SVC), $\nu$-support vector classification ($\nu$-SVC), distribution estimation (one-class SVM), ε-support vector regression (ε -SVR), and $\nu$ -support vector regression ($\nu$ -SVR) are some of the formulations of LIBSVM.

## 3.5 SVR Training, Testing and Validation Datasets

The available dataset is divided into three parts. A fraction of 60% data is used for training the SVR, 20% is used for validation and the rest is used for testing. Since the goal of cross validation is to test the model in the training phase, we have chosen hold-out cross validation, which means all validation dataset is used

for testing and this helps in the case when the data size is small to avoid over fitting. For our experiments three datasets are available, out of which two datasets are collected using GPS mobile device probing and the other is generated using micro Simulator VISSIM.

As described in section 2.3, GPS data is collected using an android mobile device with a frequency of 0.1Hz on Morgantown Beechurst Avenue for a week. This data set has 34 trips in total with single trip duration of 20 minutes. For the sake of analysis we have chosen 10 links from each trip. Due to less data availability in this dataset, we used 32 trips for training, 1 trip data to formulate the optimum <C, $\gamma$> and the last trip for testing. One more important step in preprocessing is to form time slots which involve formation of half hour time slots. The data coming from the source is fine-grained which makes training for each link more complex. So all the values falling under 8AM -8:30 AM fall under time slot 8, all the values under 8.30AM – 9.00 AM fall under 8.5 and so on. This gives more samples for single time slot to run SVR.

The second dataset is taken from a standard database from Project Mobile Century. The Mobile Century data [36] which was collected on February 8, 2008 from 10:00am and 18:00pm (PST) on Interstate 880, CA, as part of a joint UC Berkeley - Nokia project, funded by the California Department of Transportation, to support the exploration of uses of GPS enabled phones to monitor traffic. The authors used Nokia N95, which has embedded GPS chip-set and is capable of producing a time-stamped geo-position (latitude, longitude, altitude) every 3 seconds. From this time-stamp and position data, the instantaneous velocity is produced by the phone at the same frequency. The data has includes both northbound and southbound with each file containing the following five columns: "unixtime", "latitude", "longitude", "postmile" and "speed". We used northbound data which has 1388 trips and divided into

60%, 20%, 20% for training, validation and testing after preprocessing as described above. The entire day is divided into 16 time slots. So, each link is characterized by time of the day which is a time slot, speed on the link and travel time taken on the link. The preprocessing steps are followed by normalization of data which avoids overfitting and outliers.

The third dataset used is generated using VISSIM simulator. VISSIM is a microscopic, time step and behavior based simulation model developed to model urban traffic and public transit operations. The program can analyze traffic and transit operations under constraints such as lane configuration, traffic composition, traffic signals, transit stops, etc., thus making it a useful tool for the evaluation of various alternatives based on transportation engineering and planning measures of effectiveness. Using VISSIM, we have generated 143 cars and simulated for 600secs on 3 different road networks. This particular data set has no time slots since the duration is too small; 10 minutes.

## 3.6 Kernels Functions

There are many kernels proposed by many researchers [30]. But the basic four kernels are:

linear: $K(x_i, x_j) = x_i^T x_j$ (3.24) .

The linear kernel is the simplest kernel.

polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$ (3.25)

This mapping is popular method for non-linear modeling and so called non-stationary kernel

Radial basis function (RBF): $K(x_i, x_j) = exp\left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0$ (3.26)

Normally a Gaussian is used as RBF. The adjustable parameter sigma plays a key role in the performance of the kernel.

$$\text{sigmoid: } K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r) \tag{3.27}$$

Sigmoid kernel function is equivalent to a two-layer, perceptron neural network.

## 3.7 Feature Scaling

Standardizing the data before feeding it to a trainer is very necessary in machine learning techniques or it leads to create a model which gives more weights to few features. This sometime leads to over fitting or losing some features. So, Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data-preprocessing step. Distance measurement is one technique which is widely used in classifiers, which when performed to un-standardized data with a feature having a broad range of values; the distance will be governed by this particular feature. Ensuring standardized feature values implicitly weights all features equally in their representation.

Rescaling of data has to be done before training and predicting using a predictor model. Scaling helps the features to be independent of each other and they can be scaled by domain as stated in [43].

$$x_{scaled} = \frac{x - \min(x(:))}{\max(x(:)) - \min(x(:))} \tag{3.28}$$

Where $x$ is original value and $x_{scaled}$ is the normalized value.

Some other methods are scaling using variance of the data or scaling using domain [43].

## 3.8 Model and Parameter Selection

### 3.8.1 Model Selection

In this section we are going to explain how model formation is done for travel time estimation. We concentrated more on urban area road networks. Consider an urban network with a total of N nodes and they are connected by N-1 links. Each link is characterized by some particular road features such as an intersection with or without a traffic lights, pedestrian walkways, stop/slow signs [9] etc. These characteristics can be static if there exists a stop signal or dynamic if there is traffic with respect to time. The travel time experienced by a vehicle traveling through a node pair depends on the characteristics of the road features as well as the demand-capacity restrictions imposed by the dynamics of traffic flow. We also assume that each node pair is associated with unidirectional traffic flow. This road network can be looked as directed graph $G = (V, E)$, where $V$ is set of all nodes and $E$ is the set of all edges. We assume that any node pair$< i, i+1 > \in V$, the travel time data is available at times$0 < t_1 < t_2 < \cdots t_n$. The travel times available at those values can be labeled as $TT_i \ \forall \ i \ \in V - 1$. The inputs to these models in this thesis are travel time of the day $(ToD)$ and speed of entry on to the link $(V_{st})$.

## 3.8.2 Parameter Selection

LIBSVM provides many kernels and we have to choose one among them which gives better results. In general the parameter selection tool assumes that the RBF (Gaussian) kernel because it nonlinearly maps samples into higher dimensional space. The RBF kernel takes the form [42]

$$K(x_i, x_j) = exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma > 0 \qquad (3.29)$$

As stated by the authors in [44] the polynomial kernel has more hyperparameters than the RBF kernel and the k$^{th}$ value of RBF kernel states in between 0 and 1 whereas of other kernels it may go it infinity. When the number of parameters is high it is suggested to use linear kernel instead of RBF. For RBF, there are two parameters $C; \gamma$ which needed to be decided. It is required to pick best possible interval of $C$ and $\gamma$

with the grid space search which can give better accuracy. The grid search approach follows the following: One parameter is fixed and the other parameter is varied then the parameter that is fixed is incremented to next value and continued. We varied $C$ and $\gamma$ in the range [0, 1000] and presented the results for best possible range. Then, a search is made in all grids to find which pair gives highest CV accuracy. Users can then use the best parameters to train the whole training set and generate the final model. LIBSVM can run jobs in a parallel environment.

## 3.9 TTE using SVR technique

The experiment was conceived as a proof of concept of the system described in the earlier sections. It was designed with two fundamental goals.

1) More accurate estimation in velocity gives more accurate travel time.
2) Link starting velocity is sufficient for the estimation of travel time.

Link travel times ($TT_i$) are calculated using estimated models. The estimated/predictor models are created by training each link with corresponding training data then validated for optimum values of C and γ. Once the optimum values are obtained we form a structural array of predictor models for each link. The following algorithm gives the travel time taken to traverse the entire length of the road by summing all TTs on all the links traversed and pictorially given in Figure 3.3.

**Variables**:
*n*: Number of Links on the road network
*inputs*: Time of the day (ToD), Speed at the beginning of the link ($V_{st}$)
*output*: Travel time taken to traverse the entire road (TT)
*PredictedModels*: Generate predicted models using training data
*CrowdsourcedHalfHourMeanSpeedModels*: Generate average speed models using cloud sourced data

```
      CurrentMeanSpeedModels: Generate last 5 minutes mean speed for link at ToD
Algorithm:
      Set ToD(1) = Trip starting time on the link
      Set V_st(1)  =  Link Starting speed on the link
      for i = 1:n
            tt(i) = PredictedModels(ToD(i),V_st(i))

   if  i != n
       ToD(i+1) = ToD(i)
       Case 1:
         V_st(i+1) = CrowdsourcedHalfHourMeanSpeedModels (ToD(i+1))
       Case 2:
         V_st(i+1) = CurrentSpeedModels(ToD(i+1))
       Case 3:
         V_st(i+1) = trueSpeed(ToD(i+1));
   end
  end
```

### 3.9.1 Crowdsourced half hour mean speed and Current mean speed models

For the proposed model, the input is travel time of the day and the speed on the link. To predict the travel time for a link with unknown inputs, the true time of the day is used as one input since for our data modulation time of the day remains constant for a trip duration. The travel duration is less than the frequency modulation of the data; therefore, even if the predicted time is added to the true time as input, the prediction doesn't change. The other input (i.e., speed) is fed with the crowd-sourced half hour mean for the current link at particular time of the day. The crowd-sourced half hour mean is created for each link at each time slot of the day for half hour duration either from testing and validation set. For current mean speed, it is computed by taking last five minutes mean of speed for each link at each time slot of a day.

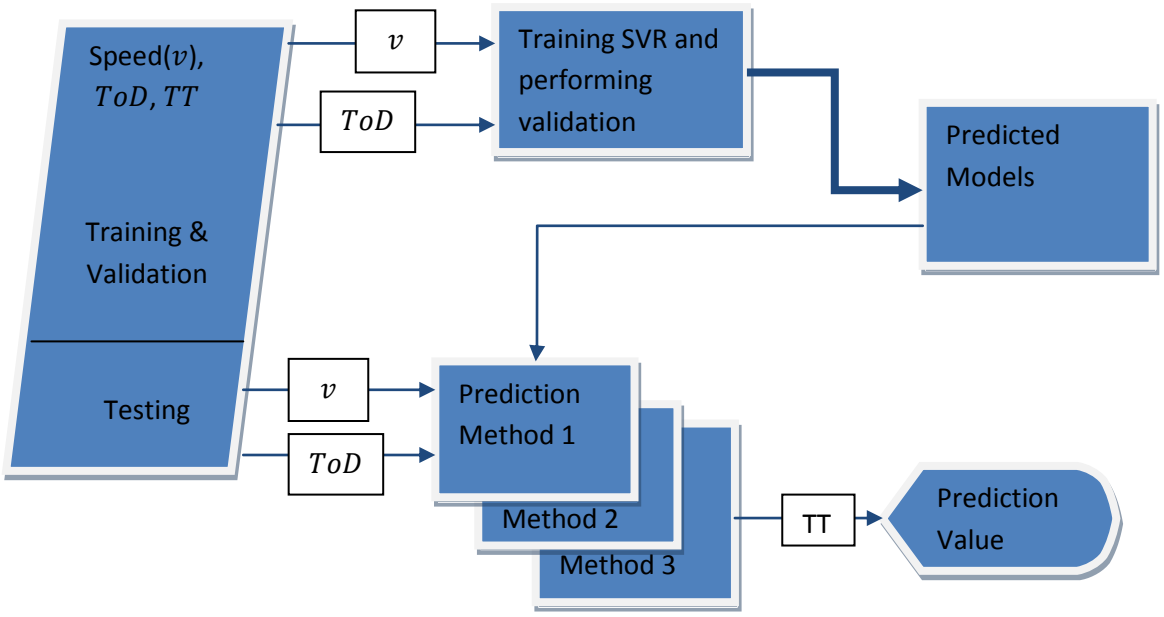


**Figure 3.3: Systematic illustration of how SVR predictor works**

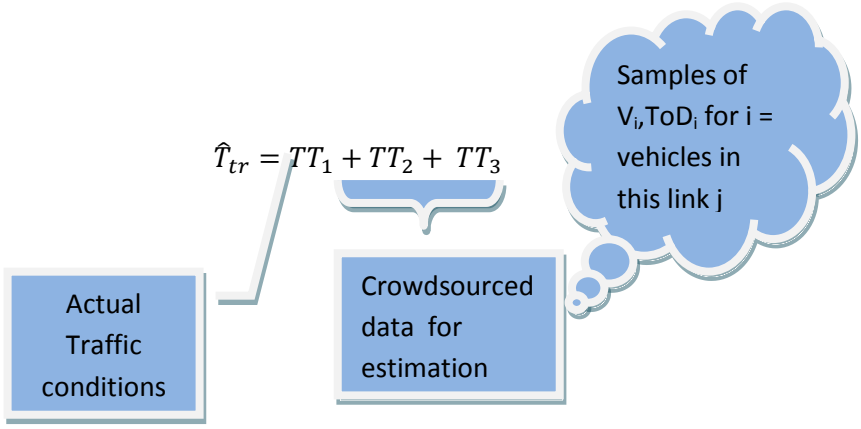


**Figure 3.4: Travel time estimation system representation**

Where $\hat{T}_{tr}$ is the total travel time taken to reach destination and $TT_i$ represents travel time taken on link i.

## 3.10 Experiments

In this section we compare the results of using SVR with those from statistical methods such as taking the historic mean of the recorded values. As described in section 1.3.4, we present the results obtained by taking the historic mean. We believe that historic mean represents the future travel time.

$$T(t, \Delta) = \overline{T}(t) \qquad (3.30)$$

where $\overline{T}(t)$ is the average of past travel time at time t and $T(t, \Delta)$ is travel time at time step $\Delta$.

As described in Section 3.7, three datasets are evaluated in our study. In the first case, we evaluate the data from Mobile Century Project as described in Section 3.7. The NOKIA N95, which was used during the data collection, did not estimate the velocity of the phone directly [36]. Hence, the speed is calculated by applying a finite-difference approximation of the position estimates provided by GPS chipset. Few outliers are still left behind, which can be observed in the following figures (i.e., Fig. 3.5 and 3.6). Support Vector Regression has the capability to handle outliers; therefore, using them poses less risk. However, the outliers drastically affect the calculation of the historic mean time. The input of the SVR predictor model is the pair of time of the day and speed of the vehicle on the link. SVR's output is predictor models for all links, which thereafter are fed with testing input pairs to get travel time taken on the corresponding link. As stated earlier, our model is link dependent. Therefore, in the entire road network, the dataset I880 is divided into 16 links. Each link can be described in terms of time of the day and speed of the vehicle on that link. Training each individual link with the training data and validating

the model predicted using hold-out validation dataset lead to the best model outcome. Fine tuning of ε-SVR can be done by choosing the best values for the parameters ε, $C$, and γ. Kernel trick is performed for the transformation of non-linear data into linear. We tested both Radial Basis Kernel (RBF) and Linear kernel. First, the predictor performance is studied for RBF kernel and then, the effect of applying linear kernels on the prediction results is investigated. The value of ε is fixed to 0.001, which gave the best performance results. The other two parameters (i.e., C and γ) are optimized for each link predictor model using a grid search. The model with the minimum mean squared error (MSE) is chosen as the best model. As shown in table below, the values of C and γ define the SVR boundary. As observed from the Table 3.1, the values of γ=8 and C=2 give the best result. The pair $(\log(2), 3\log(2))$ gives the best result, and the predictor model corresponding to this pair is chosen for the prediction of the trained link.

**Table 3.1:** MSE variation with different values of C and γ on one link

| Log(C)/log(γ) | 2 | 4 | 8 |
|---|---|---|---|
| 2 | 7.62849258731645e-07 | 4.19353968548827e-07 | 3.31772067615358e-07 |
| 4 | 7.36400947642571e-07 | 3.65578729897751e-07 | 6.32033319060932e-07 |
| 8 | 5.89970540140955e-07 | 3.76914903824149e-07 | 6.68916505675308e-07 |
| 16 | 4.90870282611699e-07 | 4.74188985362503e-07 | 8.51156425754004e-07 |
| 32 | 4.03065060463852e-07 | 5.58965447004445e-07 | 1.00169565550865e-06 |
| 64 | 3.98413185410396e-07 | 5.55466675111199e-07 | 8.26259254108130e-07 |

The predicted model for link 16 is shown in the following table. Using this predicted model, 20% of the data is used for testing. Testing phase is carried out in three different methods as shown in Fig. 3.3. Method 1 involves crowd-sourced half hour mean of the speed on a link on particular time. Method 2 involves true values of speed on the link. Method 3 involves the crowd-sourced mean calculated based on the last five minutes for the particular link. If a link has no historic data, this parameter is set to the median of the different time slots on the same link.

**Table 3.2:** Sample representation of Prediction model from LIBSVM SVR

| FIELD | VALUE | MAX | MIN |
|---|---|---|---|
| predicted_model_rbf(16).Parameters | [3;2;3;2.079400000000000;0] | 3 | 0 |
| predicted_model_rbf(16).nr_class | 2 | 2 | 2 |
| predicted_model_rbf(16).totalSV | 26 | 26 | 26 |
| predicted_model_rbf(16).rho | -0.0812 | -0.0812 | -0.0812 |
| predicted_model_rbf(16).Label | [] | | |
| predicted_model_rbf(16).sv_indices | <26x1 double> | 316 | |
| predicted_model_rbf(16).ProbA | [] | | |
| predicted_model_rbf(16).ProbB | [] | | |
| predicted_model_rbf(16).nSV | [] | | |
| predicted_model_rbf(16).sv_coef | <26x1 double> | 0.6932 | -0.6932 |
| predicted_model_rbf(16).SVs | <26x3 double> | | |

For a link, the speeds on different timeslots vary in a similar fashion as observed from the Table 3.3.

**Table 3.3:** Current mean speed for the data with missing values

| LinkID/TOD | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 69.02 | 68.24 | 64.60 | 67.29 | 67.51 | 65.26 | 71.19 | 0 | 0 | 69.01 |
| **2** | 69.70 | 69.60 | 65.21 | 67.11 | 64.49 | 66.13 | 67.62 | 0 | 0 | 69.70 |
| **3** | 71.11 | 72.42 | 62.81 | 68.61 | 68.33 | 67.15 | 69.48 | 0 | 0 | 71.11 |
| **4** | 65.51 | 80.60 | 64.70 | 63.32 | 62.15 | 69.30 | 67.36 | 0 | 0 | 65.51 |
| **5** | 66.74 | 71.43 | 63.24 | 67.03 | 64.11 | 68.20 | 66.46 | 63.79 | 60.84 | 66.74 |
| **6** | 68.78 | 75.01 | 63.53 | 66.58 | 67.14 | 67.67 | 64.31 | 67.51 | 61.42 | 68.78 |
| **7** | 67.58 | 71.03 | 60.95 | 66.22 | 66.20 | 67.08 | 61.12 | 66.05 | 62.72 | 67.58 |
| **8** | 63.45 | 72.88 | 61.33 | 64.10 | 67.99 | 68.16 | 60.97 | 64.40 | 55.51 | 63.45 |

The same method of filling missing values is followed in crowd-sourced historic half hour mean table. If a missing value is found, it is replaced with the median of the crowd-sourced historic mean speed values on different timeslots for the same link. The missing historic mean time for this dataset is set to zero(s).

The following Fig. 3.5 shows the estimated values for all trips with true inputs, travel time of the day, and speed on that link and chosen rbf kernel.

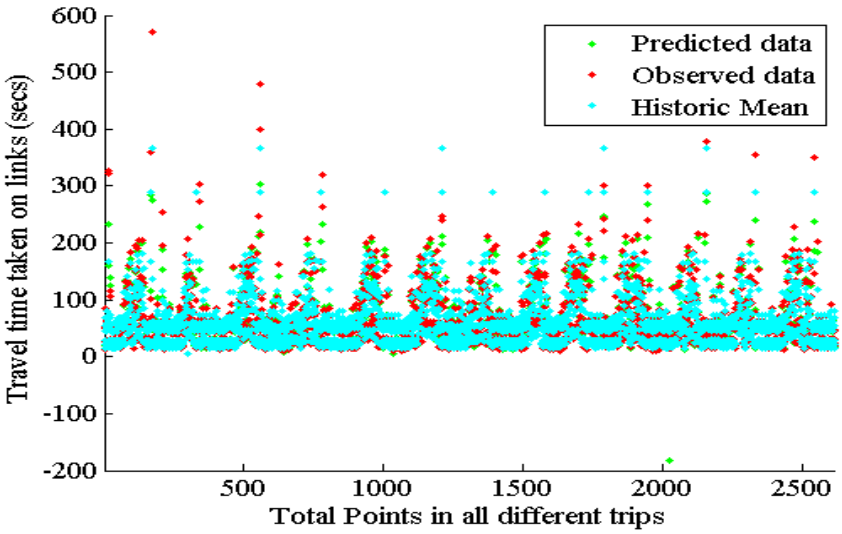


**Figure 3.5: Validation of SVR Estimator performance on Mobile Century dataset with true inputs**

It can be observed that there is a missing value in Figures 3.6 and 3.7, which is set to zero. There is no crowd-sourced data for that time zone for that particular link. For prediction, we used median of crowd-sourced historic half hour speeds as mentioned above. The root mean squared error (RMSE) of the trips is used as the performance measure and defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2} \qquad (3.31)$$

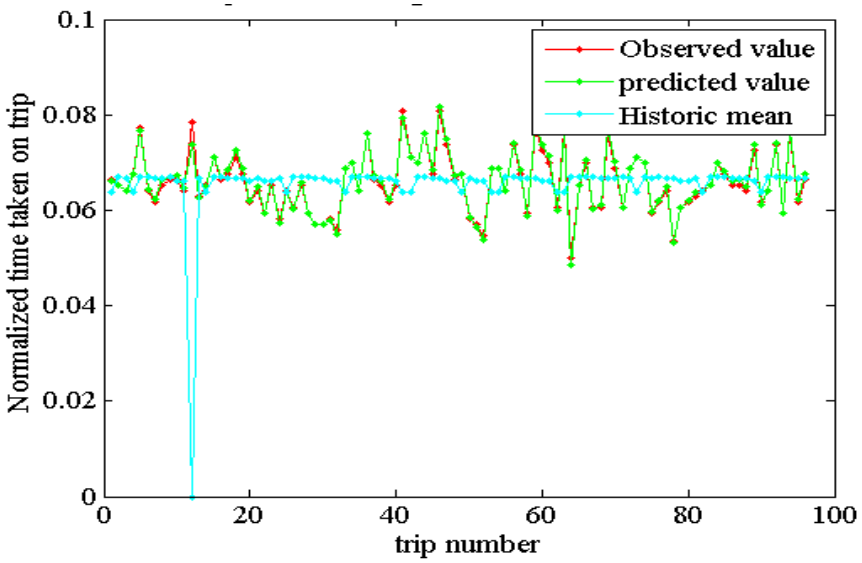


**Figure 3.6: Comparison of estimated travel time with historic mean on Mobile Century dataset for link 1**

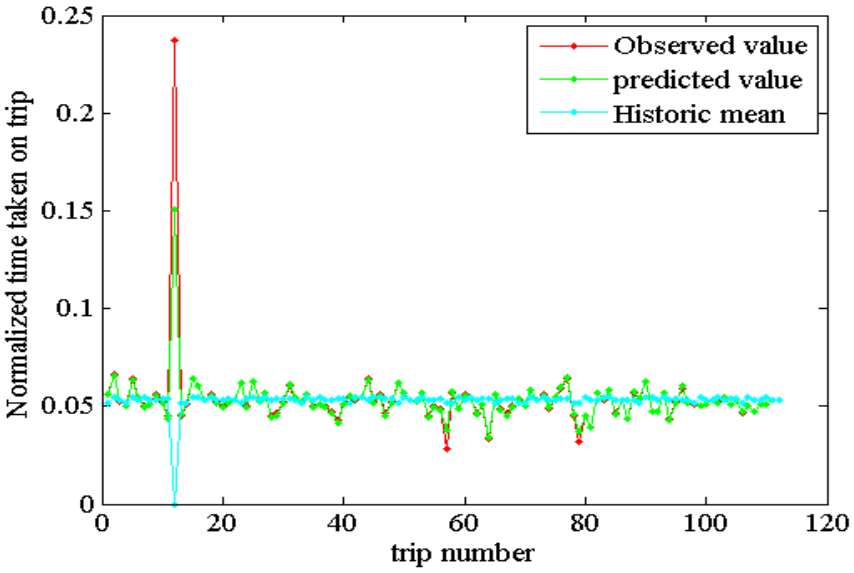


**Figure 3.7: Comparison of estimated travel time with historic mean on Mobile century dataset for link 3**

The preceding figures 3.6 and 3.7 show the prediction is close to the observed value with the RMSE of 15.6985. Observing Figure 3.8 gives a better view of how the error varies for all points in all testing trips. The error shown in this figure is the absolute error with respect to the observed value defined as

$$err_{predicted} = |\text{predictedvalue} - \text{Observedvalue}| \quad (3.32)$$

$$err_{hist\_mean} = |\text{historicmean} - \text{Observedvalue}| \quad (3.33)$$

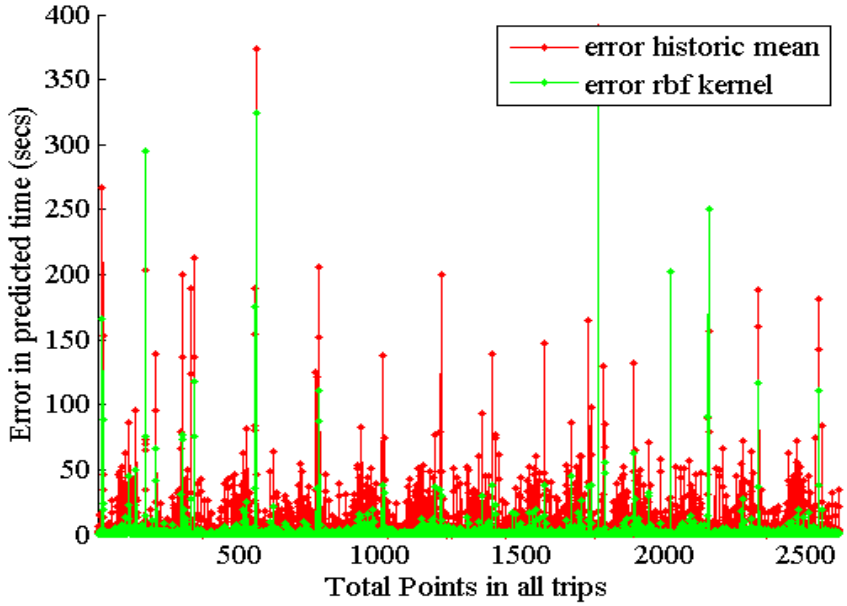


**Figure 3.8: Estimation error for all the trips points using true values of speed and ToD as inputs**

Figure 3.9 shows the cdf error plot for the Mobile Century dataset, when true values are used for prediction. As observed from this figure, in 95% of instances the prediction error is less than 7.754 sec for travel time in each trip if the true values of speed and time are used. For comparison, it should be noted that the 95% error is around 47 sec if historic mean is used for prediction.

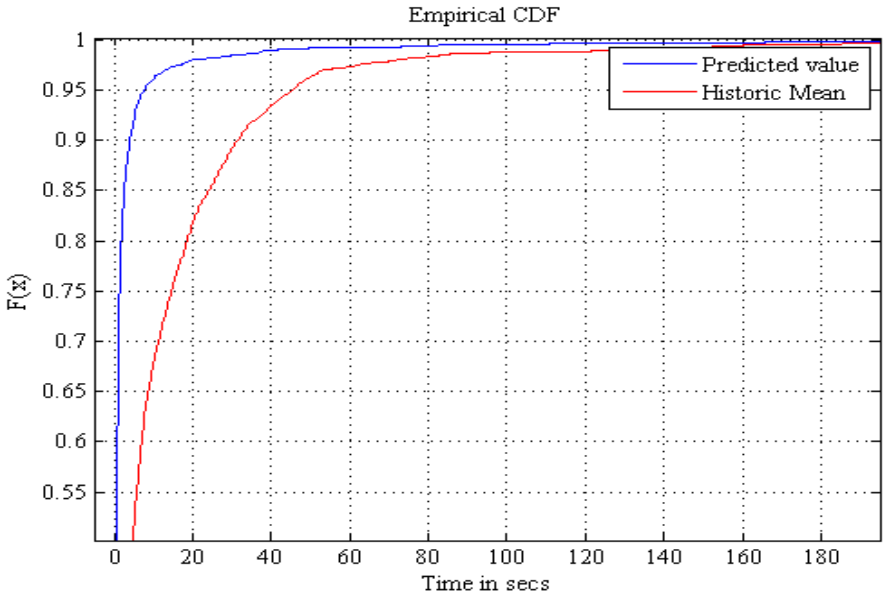


**Figure 3.9: CDF plot for errors in Mobile Century data with true values of predictor**

Consider the case with the prediction happening when no true values are available for the trips. There are two options from which speed values can be chosen:

1) Crowdsourced half hour mean speed.
2) Crowdsourced current mean speed.

Using crowd-sourced historic mean speed takes the mean of speed over entire 30 min interval. Suppose a trip starts at 11.30 AM, with the training data available we average over 30 min intervals to form a crowd-sourced historic mean table for speed on different links on different time of a day. When the predictor wants to know what the crowd-sourced speed value at time 11.30 on link 10 is, it checks at the table for the link and timeslot and picks the corresponding value. In our case we pick the value available for time interval 11.30 AM on time slot 10. The total calculated RMSE value in this case is 28.5131 and the prediction outputs for all the trips data is as shown in the plot below (i.e., Figure 3.10).

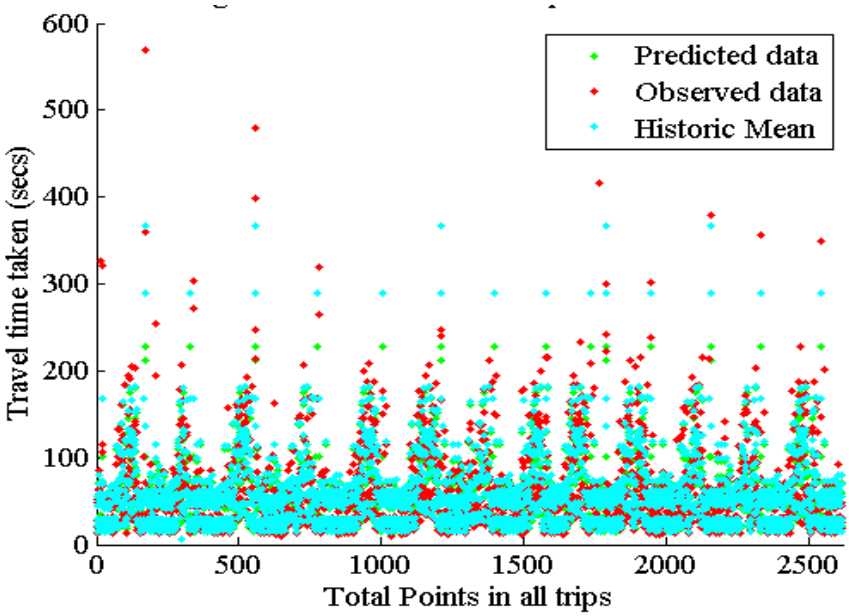


**Figure 3.10: Estimation using Crowd-sourced historic half hour mean speed vs historic mean**

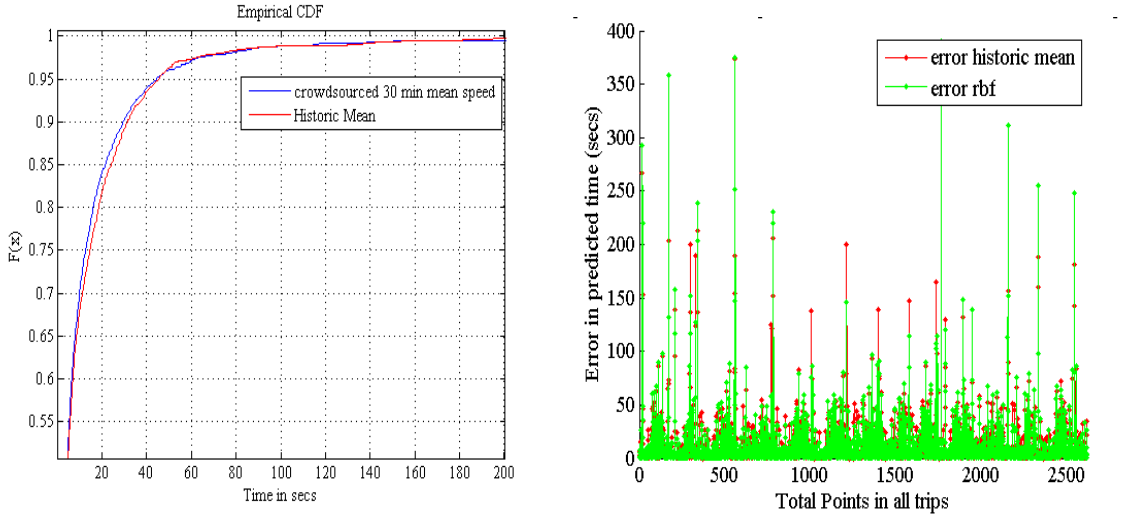


**Figure 3.11: (left) Estimation error for all trip points using crowdsourced data and rbf kernel and (right) cdf plots for errors**

Figure 3.11 shows that using historic mean gives the even distribution of errors in prediction when performed using crowd-sourced half hour mean speeds. i.e., both the predictions using historic mean and prediction using crowd-sourced historic half hour mean speeds as input give almost same amount of error. The RMSE values are shown in table below. The assumption travel time depends on speed of the day is shown in the Figure 3.12. Travel time varies with speed of the day in inverse relation. So any little deviation in the value of speed has affect on the predictor performance.

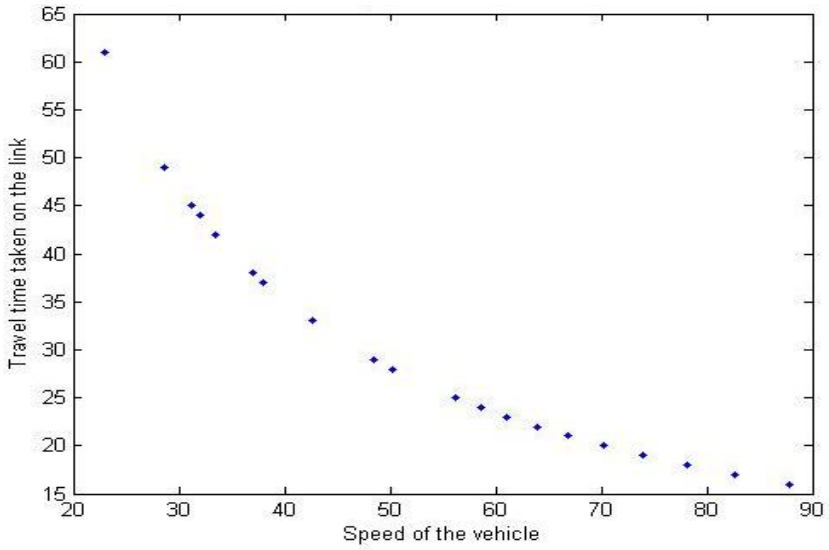


**Figure 3.12: Travel time (sec) distribution on a link 8 at a give time 11 AM**

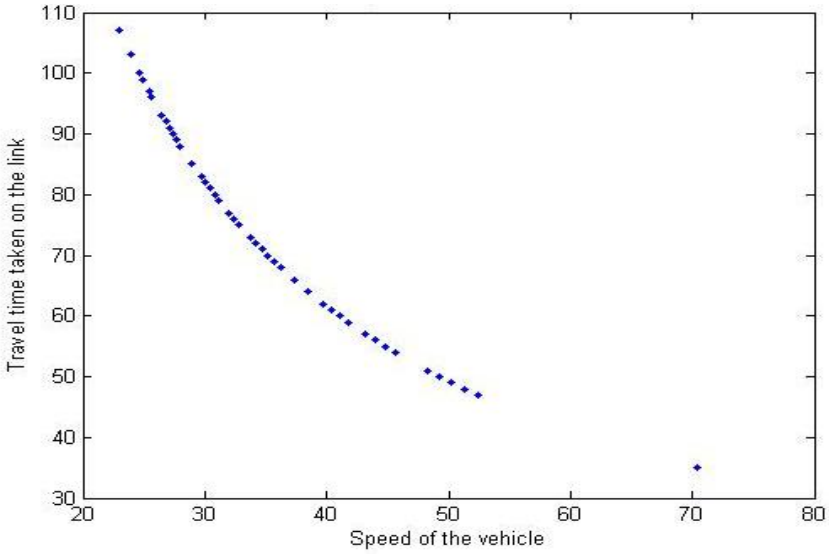


**Figure 3.13: Travel time (sec) distribution on a link 12 at a give time 11AM**

The other independent variable is time of the day and the plots below show how speed varies with time of the day. The intuition is, in peak hours of the day there is heavy traffic and so the speed is low with high travel time to reach the destination.

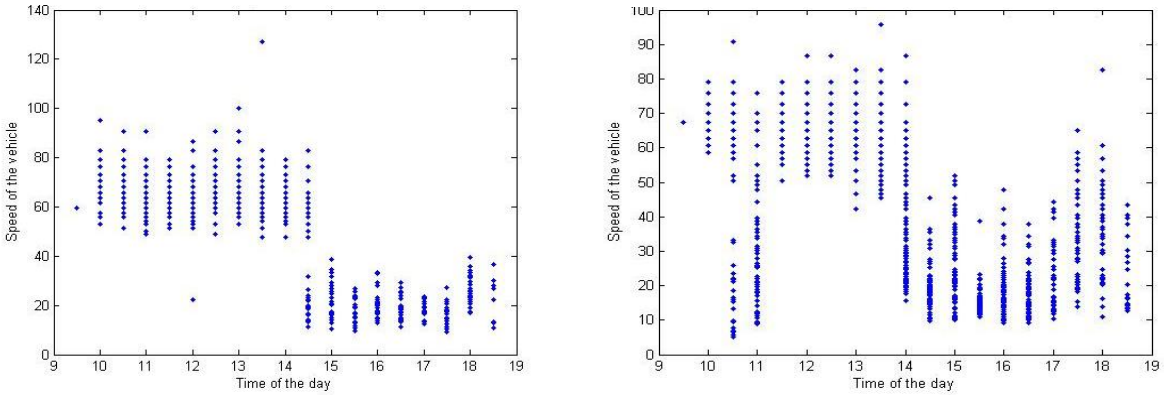


**Figure 3.14: Distribution of Speed on link 6 and link 10**

It can be observed that there is a pattern in travel time with respect to time of the day. During afternoon times because of high traffic, speed values are very low, i.e., the vehicles are in congested state and it takes longer duration than the morning times.

In the later case, when crowd-sourced current mean speed is used for testing i.e., by computing the mean speeds for the last 5 minutes and using that value as testing input, the RMSE for total trips is 37.1959. The plots below (Figure 3.15) show the predictions for all the trips.

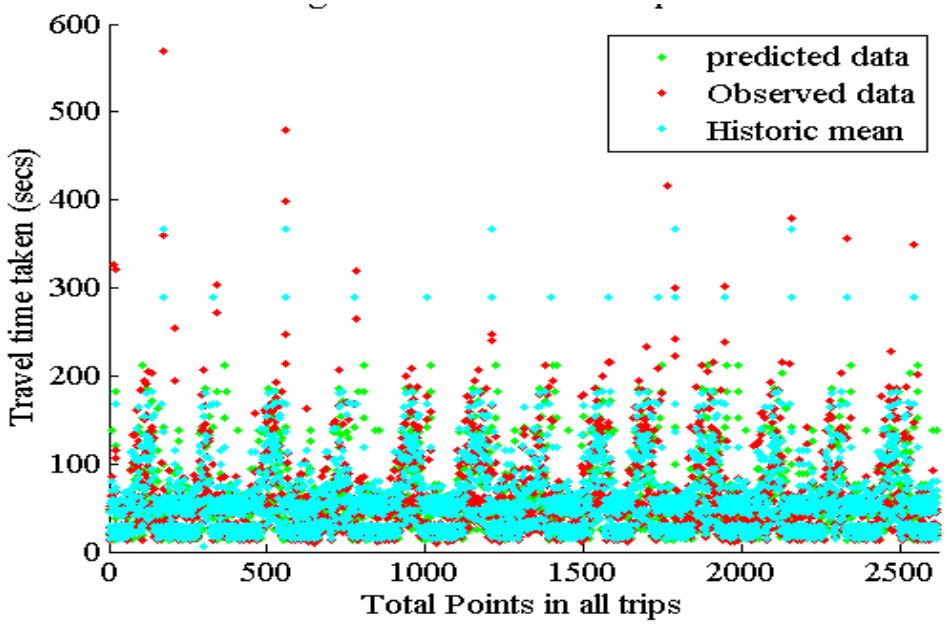


**Figure 3.15: Estimation for all trip points using crowdsourced current mean using rbf kernel**

The error plot gives the performance of the predictor. The figure shows that the error values are evenly distributed for both the estimations using historic mean data and using crowd-sourced current mean speed. This implies the predictor is performing to the mean value and can be used in case when there is no true data coming from the user to predict the travel time. From the 95% error plot it can be observed that the error in prediction with historic mean and prediction using crowd-sourced current mean speed perform

same till 85% of time. The error in the trip is around 25 seconds but later on the predictor using crowd-sourced current mean has bad performance when compared to historic mean. This is because the crowd-sourced current mean has around 10% deviation from true values.

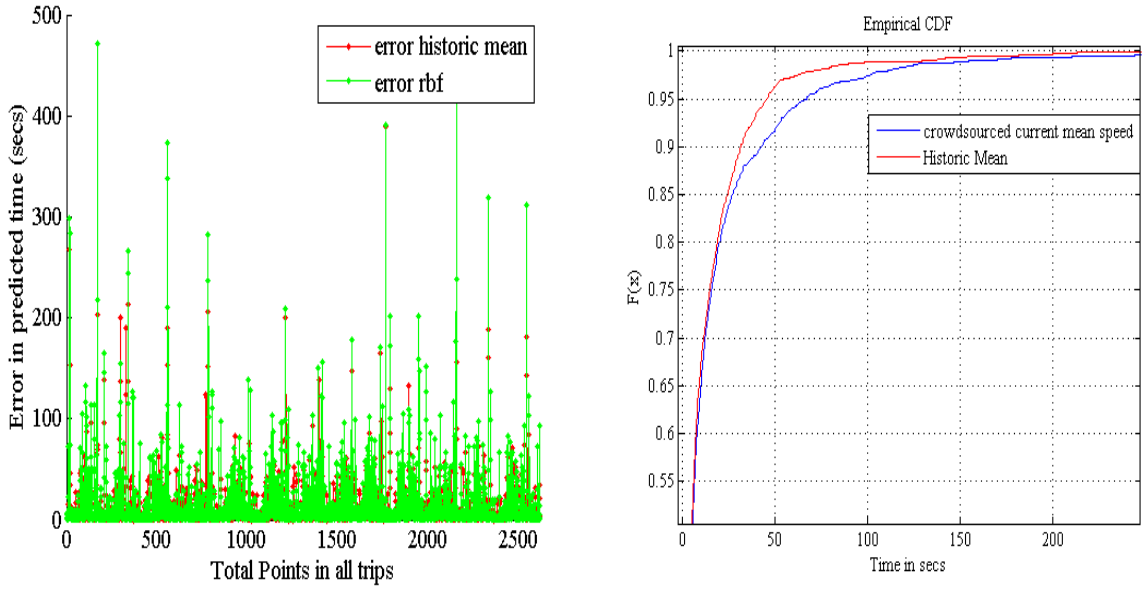


**Figure 3.16: Estimation error for all trips with crowdsourced current mean as input**

Using Linear kernel to the predictor gives moderate results. The RMSE for total trips is as shown in Table 3.4.

**Table 3.4:** RMSE representation for different scenarios with Mobile century dataset

| RMSE Kernel/Speed | Crowdsourced Historic half hour mean Speed | Crowdsourced Current mean speed | True speed | Historic mean |
|---|---|---|---|---|
| RBF | 28.51 | 37.20 | 15.70 | 26.97 |
| Linear | 31.40 | 37.22 | 26.18 | 26.97 |

On observation, increase in deviation from the true values produced large errors in the prediction. When the true values of speed and time of the day are used as predictor input the RMSE is 15.70 but with increase in noise which is introduced using Matlab 'rand' function to the results produced the RMSE values as shown in table below. It can be observed that with increase in noise percentage, that a predictor performance approaches historic mean performance as shown in Table 3.5. This is other supporting feature for this work to show that more accurate estimation in speed values is required for good predictor. The results when we use the true values of speed and time of day are satisfactory and helps in achieving our goal stating starting velocity of the links are sufficient to predict travel time taken on that link. From the Figure 3.17, it is observed that 7% noise in data accounts to same error produced for crowdsourced half hour mean.

**Table 3.5:** Effect of noise on predictor performance

| **Noise** | **1%** | **5%** | **10%** |
|---|---|---|---|
| **RMSE** | 16.00 | 20.93 | 33.66 |

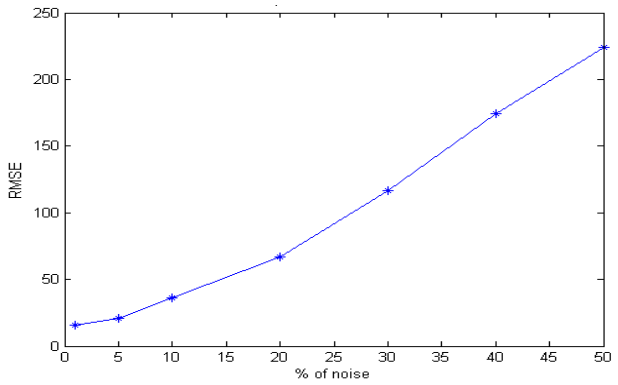


**Figure 3.17: Effect of noise on input speed**

In the second case, consider the VISSIM dataset mentioned in section 3.4. This dataset is formulated with three road networks which has 9 sub-networks i.e., 9 links. As shown from the Table 3.6, the links are formed by considering the ordered pairs (0,1), (0,2), (0,3), (0,10000), (0,10001), (1,1), (1,10000), (2,1), (2,10001). The simulated 143 vehicles enter a section of this road network and completes a trip.

**Table 3.6:** Link node representation of VISSIM generated road network

| **RouteId** | **0** | **1** | **2** |
|---|---|---|---|
| **LinkIds** | 1,2,3,10000,10001 | 1,10000 | 1,10001 |

The SVR predictor is trained for this 9 different links using speed and time of the day as inputs to form 9 independent models. When a test vehicle enters this link, using the vehicles time of entry and speed at which it started, the travel time can be estimated. The two independent variables are fed to the predictor by selecting the corresponding link predictor model and the output is travel time taken on the link. The dataset generated is for 10 minutes and therefore is not lots of data for training or testing for each link. Most fraction of data is used for training and only one trip is used for validation which gives the optimum values of $<C,\gamma>$. Prediction is carried out for the last trip which is a road network formed by car 143 traversing 4 links. The prediction gave better results using true values of inputs with linear kernel as kernel technique. The results are shown in Figures 3.18 and 3.19.

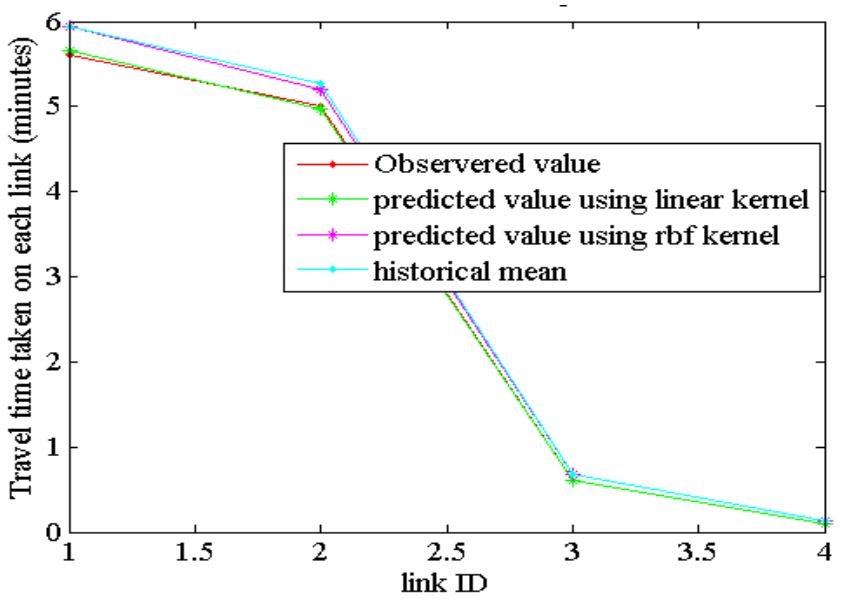


**Figure 3.18: Estimation for last trip on VISSIM generated data**

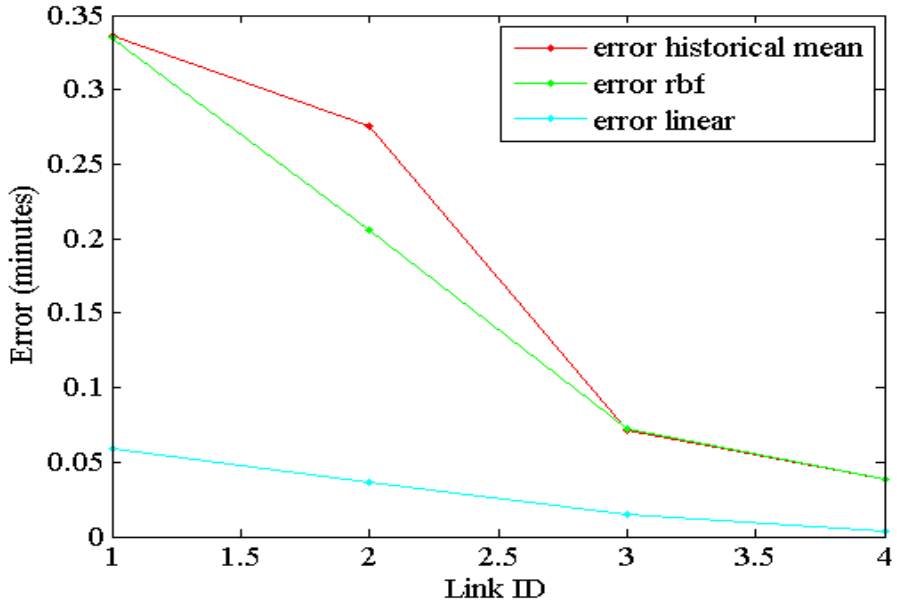


**Figure 3.19: Estimation error for last trip on VISSIM generated data**

The experiment is carried out using average speed of the link and true value of time. The values of C and γ are chosen from a window size of [log (2^.[1:20]), log (2^.[1:10])]. The prediction is performed on the last vehicles data using the best predictor models formed with RBF and Linear kernel. The average time taken for a trip is 11.3 seconds. Table 3.7 shows the corresponding trip errors for different cases.

**Table 3.7:** Performance measurement table for VISSIM generated data

| **Absolute error Kernel/Input values** | **True values of speed and new time in seconds** | **True values of time and crowd-sourced historic mean speed in seconds** |
|---|---|---|
| **LINEAR** | 0.11 | 0.44 |
| **RBF** | 0.65 | 0.65 |
| **AVERAGE** | 0.72 | 0.72 |

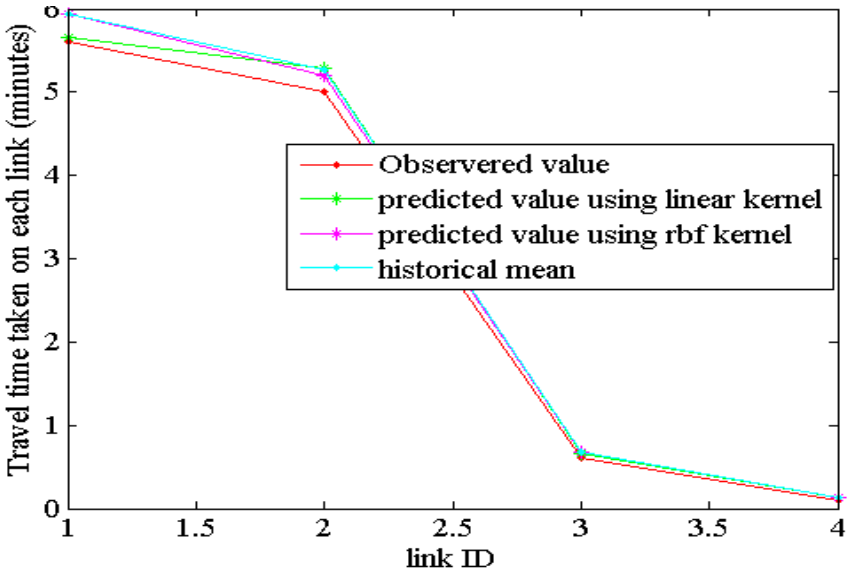


**Figure 3.20: Estimation using historic mean speed for last trip data from VISSIM**

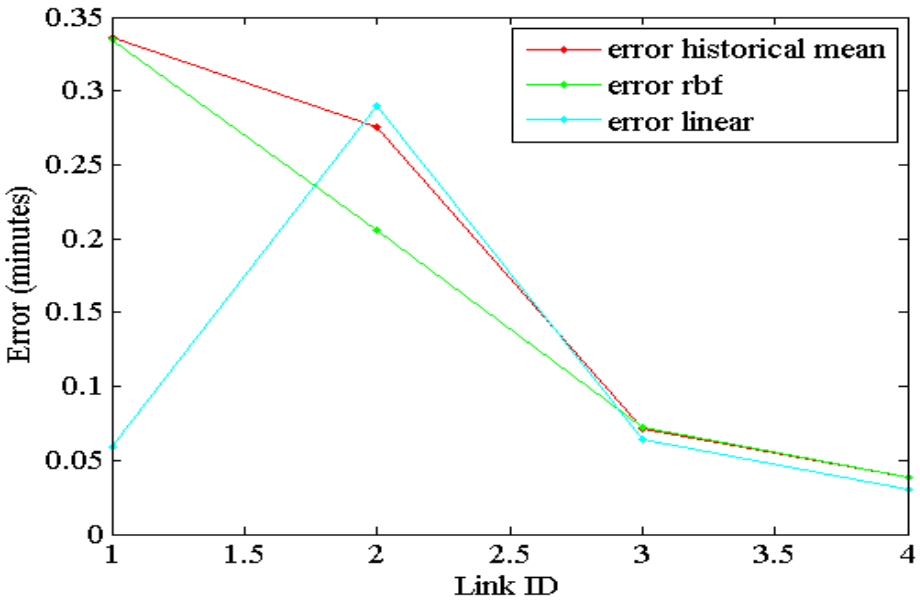


**Figure 3.21: Estimation error for last trip with historic mean of speed for VISSIM data**

Third dataset is the one which is collected in Morgantown, WV for a week duration starting from 8:00AM to 11:00 AM as stated in section 2.3. The dataset was collected with frequency of 0.1Hz with trip duration of 20 min. For our purpose of link model creation, each entire trip is divided 40 links; 10 links data is used for prediction. This experimental system is not fully stabilized and therefore the dataset has lot of discrepancies like missing values and floating data. We performed moving average technique to fill the missing values and performed map matching to match the floating GPS points. The plots shows below gives the predictor outputs obtained after passing the dataset through the SVR training and validation and finally through testing process. Prediction input methods are same as described in Mobile century dataset, which we use true values of speed and time of the day as inputs in first case. The later is with crowd-sourced historic half hour mean of speed and travel time of the day as input. Thereafter the results are compared with historic mean of time.

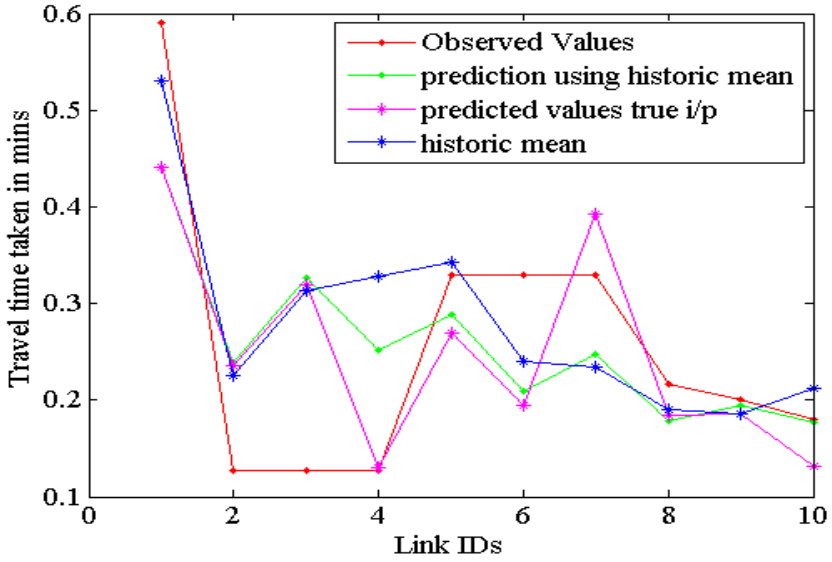


**Figure 3.22: Estimation for trip 34 using rbf kernel and true inputs from Morgantown dataset**

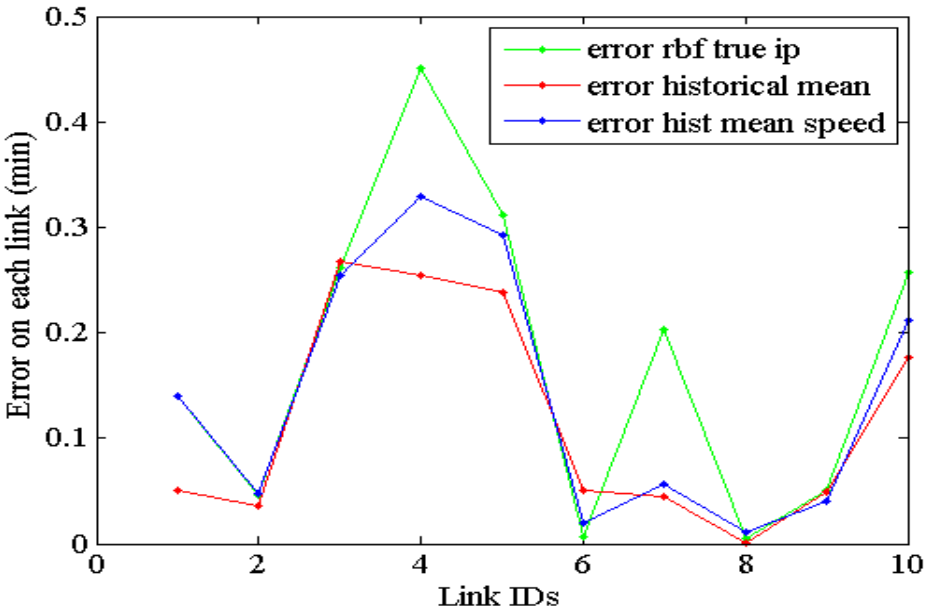


**Figure 3.23: Estimation error for trip 34 from Morgantown**

Error analysis is done using the RMSE value of the predictor in two different cases and is shown the table below. The mean trip time for one trip is 3.71 min.

**Table 3.8:** Performance measure for dataset from Morgantown

| Kernel and speed Used | RMSE | Total error for trip (minutes) |
|---|---|---|
| **RBF True speed** | 0.2236 | 1.73068 |
| **RBF crowdsourced mean** | 0.1820 | 1.40556 |
| **Linear True speed** | 0.1945 | 1.46646 |
| **Linear crowdsourced mean** | 0.1814 | 1.34781 |
| **Historic mean** | 0.1531 | 1.16640 |

Since the data is so erroneous the prediction rate is very low for this case. The moving average technique followed in preprocessing to fill the missing entries didn't seem to be a good method. This dataset performance can be improved by collecting larger amounts of data with more accuracy.

From the results presented in this chapter, we claim that time of the day and speed on the link are two important parameters for travel time prediction/Estimation. We also proceed stating better the true nature of inputs the better the predictor performance with support vector regression technique.

# Chapter 4: Conclusions

This thesis mainly concentrates on two research problems: The collection of traffic-related data using mobile phones and studying the requirements and methods for estimation of travel time using the collected GPS data. The main requirements for achieving these two objectives are building a real-time data collection service and developing advanced methodologies for quick estimation of travel time. Our research includes building a GPS sensing system, applying a map matching algorithm to match the collected data with a standard dataset, and estimating the travel time per road section in an urban context. We use the proposed system to evaluate the requirements on accuracy of crowdsourcing data for travel time prediction.

The first step was to define a link model and travel time taken on the link. Next, the parameters affecting a link model were explored. By conducting different experiments, we showed that on an urban road network, the speed of the vehicle and the time of entry on the link are two important parameters that can be used to predict the travel time on each link. In the next step, we studied methods to deal with missing values in the data and corrected them by adding moving average or median of the link data.

For our research, we used three datasets, where two of them are real-world data and one is generated using VISSIM traffic simulator. For the data coming from the simulator, there is no need for correction; but for the data coming from the real world, there are missing entries that are due to loss of network GPS signal, phone application crash, and transmission loss. This situation has to be carefully handled when collecting data. The three datasets are used for prediction, and the predicted output is compared with statistical methods such as historic mean, and the observed data.

The first two data sets, i.e., the dataset coming from Mobile Century and the other one coming from micro-simulator VISSIM, prove the reliability of the data. Dataset 3 collected in Morgantown is erroneous in nature and has to be carefully studied for further improvements. The results lead to large deviation in prediction when the predictor is run using crowd-sourced historic mean of speed, which has around 7% error. The conclusion from this observation is that the data coming from the source has to be accurate. This conclusion is further substantiated considering the results when the predictor runs based on true values of speed. Further analysis of the predictor performance using the crowd-sourced current mean, which is the mean of the speed data over the last 5 minutes, resulted in RMSE of 37.20. The results are above the RMSE value when we use historic mean of time on the link. The approach based on the mean of the speed data over the last 5 minutes produces 10% deviation compared to the true data, which results in the degradation of predictor performance.

In summary, the dynamics of travel time estimation using machine learning techniques, namely SVR, is an alternative technique that can be used for the prediction of link travel time when we have a large set of input values that are good estimates of actual values on the road. In the absence of high-quality and large data inputs, historic mean of data can be used for the travel time estimation. In general, using GPS data and SVR-based machine learning techniques require large data sets that are good representative of the actual case.

# Bibliography


[1] Lelitha Devi.V (2004), "ESTIMATION AND PREDICTION OF TRAVEL TIME FROM LOOP DETECTOR DATA FOR INTELLIGENT TRANSPORTATION SYSTEMS APPLICATIONS", A Ph.D Dissertation at Texas A&M University

[2] Turner, Shawn M., William L. Eisele, Robert J. Benz, and Douglas J. Holdener.Travel time data collection handbook. No. FHWA-PL-98-035,. 1998.

[3] Byeong-Seok, Y. O. O., and Chang-Ho Park. "Travel time estimation using mobile data." In *Proceedings of the Eastern Asia Society for transportation studies*, vol. 5, pp. 1533-1547. 2005.

[4] Klein, Lawrence A., Milton K. Mills, and David RP Gibson. *Traffic Detector Handbook*: - Volume I. No. FHWA-HRT-06-108. 2006.

[5] Richard W.Lyles, John H.Wyman, "An Operational Review of Traffic Data Collection Systems", *ITE Journal*, December 1983.

[6] RNCOS Business Consultancy Services, "GPS Market Forcast to 2015", accessed on 04/10/2014, http://www.rncos.com/Report/IM443.htm.

[7] GPS.Gov, "Space Segment and Constellation Arrangement", last modified on April 1, 2014, http://www.gps.gov/systems/gps/space/.

[8] Quiroga, Cesar A., and Darcy Bullock. "Travel time information using global positioning system and dynamic segmentation techniques." *Transportation Research Record: Journal of the Transportation Research Board* 1660, no. 1 (1999): 48-57.

[9] Herring, Ryan, et al. "Using mobile phones to forecast arterial traffic through statistical learning." In *89th Transportation Research Board Annual Meeting, Washington DC.* 2010.

[10] Hunter, Timothy, Ryan Herring, Pieter Abbeel, and Alexandre Bayen. "Path and travel time inference from GPS probe vehicle data." *NIPS Analyzing Networks and Learning with Graphs (2009)*.

[11] Li, Yanying, and Mike McDonald. "Link travel time estimation using single GPS equipped probe vehicle." In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference* on, pp. 932-937. IEEE, 2002.

[12] Cayford, Randall, and Tigran Johnson. "Operational parameters affecting the use of anonymous cell phone tracking for generating traffic information." In *Institute of transportation studies for the 82th TRB Annual Meeting*, vol. 1, no. 3, pp. 03-3865. 2003.

[13] Bradford S.Westgate, "VEHICLE TRAVEL TIME DISTRIBUTION ESTIMATION AND MAP-MATCHING VIA MARKOV CHAIN MONTE CARLO METHODS", A Ph.D Dissertation at Cornell University

[14] Box, George EP, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time series analysis: forecasting and control*. John Wiley & Sons, 2013.

[15] Yang, Jiann-Shiou. "Travel time prediction using the GPS test vehicle and Kalman filtering techniques." *In American Control Conference, 2005. Proceedings of the 2005*, pp. 2128-2133. IEEE, 2005.

[16] Westgate, Bradford S., Dawn B. Woodard, David S. Matteson, and Shane G. Henderson. "Travel time estimation for ambulances using Bayesian data augmentation." *The Annals of Applied Statistics* 7, no. 2 (2013): 1139-1161

[17] Yeon, Jiyoun, and Byungkon Ko. "Comparison of travel time estimation using analysis and queuing theory to field data along freeways." In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference* on, pp. 530-538. IEEE, 2007

[18] Petty, Karl F., Peter Bickel, Michael Ostland, John Rice, Frederic Schoenberg, Jiming Jiang, and Ya'acov Ritov. "Accurate estimation of travel times from single-loop detectors." *Transportation Research Part A: Policy and Practice* 32, no. 1 (1998): 1-17.

[19] Coifman, Benjamin. "Estimating travel times and vehicle trajectories on freeways using dual loop detectors." *Transportation Research Part A: Policy and Practice* 36, no. 4 (2002): 351-364.

[20] Christopher C. Schrader, Alain L. Kornhauser, Laura M. Friese, "Using Historical Information in Forecasting Travel Times", *Transportation Research Board Annual Meeting Washington, DC* January 2004.

[21] Lin, Wei-Hua, and Jian Zeng. "Experimental study of real-time bus arrival time prediction with GPS data." *Transportation Research Record: Journal of the Transportation Research Board* 1666, no. 1 (1999): 101-109.

[22] Rice, John, and Erik Van Zwet. "A simple and effective method for predicting travel times on freeways." *Intelligent Transportation Systems, IEEE Transactions on* 5.3 (2004): 200-207.

[23] Chien, Steven I-Jy, and Chandra Mouly Kuchipudi. "Dynamic travel time prediction with real-time and historic data." *Journal of transportation engineering129*, no. 6 (2003): 608-616.

[24] Zegeye Kebede Gurmu,"A Dynamic Prediction of Travel Time for Transit Vehicles in Brazil Using GPS Data", A Ph.D Dissertation at University of Twente

[25] Kwon, Jaimyoung, Benjamin Coifman, and Peter 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, no. 1 (2000): 120-129.

[26] Yang, Jiann-Shiou. "A study of travel time modeling via time series analysis." In *Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference* on, pp. 855-860. IEEE, 2005.

[27] Bontempi, Gianluca, Souhaib Ben Taieb, and Yann-Aël Le Borgne. "Machine Learning Strategies for Time Series Forecasting." In *Business Intelligence*. Springer Berlin Heidelberg, 2013. 62-77.

[28] Karunasinghe, Dulakshi SK, and Shie-Yui Liong. "Chaotic time series prediction with a global model: Artificial neural network." *Journal of Hydrology* 323, no. 1 (2006): 92-105.

[29] Andras, Peter. "The equivalence of support vector machine and regularization neural networks." *Neural Processing Letters* 15.2 (2002): 97-104.

[30] Gunn, Steve R. "Support vector machines for classification and regression."*ISIS technical report* 14 (1998).

[31] Muller Smola , A. J. Smola , G. Ratsch , B. Scholkopf , J. Kohlmorgen , V. Vapnik, "Predicting Time Series with Support Vector Machines", *Artificial Neural Networks — ICANN'97*, pp 999-1004, 1997

[32] Ho, Chia-Hua, and Chih-Jen Lin. "Large-scale linear support vector regression." *The Journal of Machine Learning Research* 13.1 (2012): 3323-3348.

[33] Takeda, Akiko, Jun-ya Gotoh, and Masashi Sugiama. "Support vector regression as conditional value-at-risk minimization with application to financial time-series analysis." *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop* on. IEEE, 2010.

[34] Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." *Journal of basic Engineering* 82, no. 1 (1960): 35-45

[35] Chu, Lianyu, S. Oh, and Will Recker. "Adaptive Kalman filter based freeway travel time estimation." In *84th TRB Annual Meeting, Washington DC*. 2005.

[36] Herrera, Juan C., et al. "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment." *Transportation Research Part C: Emerging Technologies* 18.4 (2010): 568-583.

[37] Amnir Hadachi, "Travel Time Estimation Using Sparsely Sampled Probe GPS Data in Urban Road Networks Context", A Ph.D Dissertation at Normandie University

[38] Developers Google Maps API, accessed on 4/10/14, https://developers.google.com/maps.

[39] Android Open Source project, "Application Fundamentals", accessed on 3/31/14, http://developer.android.com/guide/components/fundamentals.html.

[40] Lou, Yin, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. "Map-matching for low-sampling-rate GPS trajectories." In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 352-361. ACM, 2009.

[41] Campbell, Colin. "Kernel methods: a survey of current techniques". *Neurocomputing* 48.1 (2002): 63-84.

[42] Chang, Chih-Chung, and Chih-Jen Lin. "LIBSVM: a library for support vector machines." *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, no. 3 (2011): 27.

[43] Juszczak, P., D. Tax, and R. P. W. Duin. "Feature scaling in support vector data description." In *Proc. ASCI*, pp. 95-102. 2002.

[44] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A practical guide to support vector classification." (2003).

# Appendix

## A.1 Android Manifest

Every Android application should have an android manifest file which basically presents all the essential information about the application developed. One important feature is it declares all the permissions required for the application in order to access the protected parts of android API. Given below is our manifest file shows that there is a main process TrafficMapActivity which will be launched on the start of the application. Access_fine_location and Access_coarse_location are the two permissions used by the GPS sensor to gain access to the GPS signals in fine and coarse location strategies.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cyber.trafficmap"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <uses-library android:name="com.google.android.maps" />
        <activity
             android:name=".TrafficMapActivity"
             android:label="@string/app_name"
             android:configChanges="keyboardHidden|orientation">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
```

```
            </intent-filter>
        </activity>
        <service android:name=".UpdaterService"></service>
    </application>
</manifest>
```

## Android Layout

Layout file describes the components in user interface. It can be done in two ways: One way is to do it using XML file and other method is to initiate the elements during runtime. In our application we used XML representation and given below is a layout.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

<com.cyber.trafficmap.OnDoubleTap
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                          android:apiKey="0vLpyotXXXXXXXXXXXXXXXXXXXXXXXX"
                android:id="@+id/mvMain"
                android:enabled="true"
                android:clickable="true"/>
</LinearLayout>
```

## Activity Lifecycle

A simple representation of activity of an application is shown in figure below. The application is launched everything declared in onCreate( ) method acts like a global state and everything is destroyed when it reaches onDistroy( ) method. More information about the lifecycle of an activity can be found in android developers website.

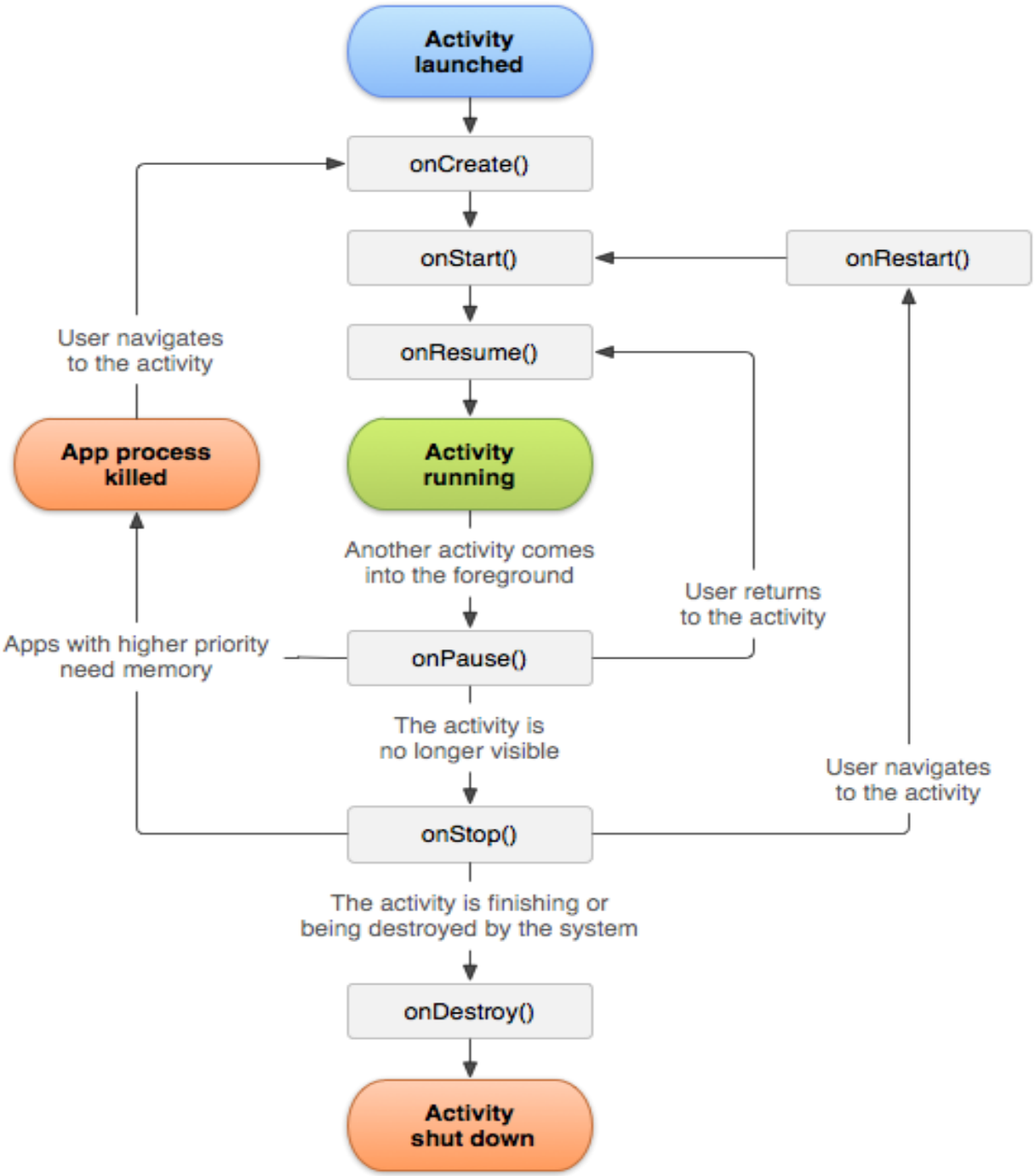


**Figure A.1: Systematic diagram for representing state paths of an activity [39]**

# A.2 Solution for Optimal plane separating classes

$$\phi(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \left(y^i\left[\langle w, x^i\rangle + b\right] - 1\right) \qquad (A.1)$$

where α are the Lagrange multipliers.

The Lagrangian has to be minimized with respect to w, b and maximized with respect to $\alpha \geq 0$. Classical Lagrangian duality [30] enables the primal problem, Equation 3.9, to be transformed to its dual problem, which is easier to solve. The dual problem is given by,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} \left(\min_{w,b} \phi(w,b,\alpha)\right) \qquad (A.2)$$

The minimum with respect to w and b of the Lagrangian, , is given by,

$$\frac{\partial \phi}{\partial b} = 0 \Rightarrow \sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (A.3)$$

$$\frac{\partial \phi}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{l} \alpha_i y_i x_i \qquad (A.4)$$

hence from equations A.1, A. 2, A.3, A.4 the dual problem is

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i\, y_j \langle x_i, x_j\rangle + \sum_{k=1}^{l} \alpha_k \qquad (A.5)$$

and the solution is

$$\alpha^* = \arg\min \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i \alpha_j y_i\, y_j \langle x_i, x_j \rangle - \sum_{k=1}^{l} \alpha_k \qquad (A.6)$$

with constraints

$$\alpha_i \geq 0, i = 1, \ldots., l$$

$$\sum_{j=1}^{l} \alpha_j y_j = 0 \qquad (A.7)$$

Solving Equation A.6 with constraints Equation A.7 determines the Lagrange multipliers,

and the optimal separating hyperplane is given by,

$$w^* = \sum_{i=1}^{l} \alpha_i y_i x_i$$

$$b^* = -\frac{1}{2}\langle w^*, x_r + x_s \rangle \qquad (A.8)$$

where $x_r$ and $x_s$ are any support vector from each class satisfying,

$$\alpha_r, \alpha_s > 0, y_r = -1, y_s = 1 \qquad (A.9)$$

The hard classifier is then,

$$f(x) = sgn(\langle w^*, x \rangle + b) \qquad (A.10)$$

This may be more appropriate than the hard classifier of Equation A.10, because it produces a real valued output between −1 and 1 when the classifier is queried within the margin, where no training data resides. From the Kuhn-Tucker conditions,

$$\alpha_i \left( y^i \left[ \langle w, x^i \rangle + b \right] - 1 \right) = 0, i = 1, \ldots\ldots, l \qquad (A.11)$$

and hence only the points $x^i$ which satisfy,

$$y^i\left[\langle w, x^i\rangle + b\right] = 1 \qquad (A.12)$$

will have non-zero Lagrange multipliers. These points are termed Support Vectors (SV). If the data is linearly separable all the SV will lie on the margin and hence the number of SV can be very small. Consequently the hyperplane is determined by a small subset of the training set; the other points could be removed from the training set and recalculating the hyperplane would produce the same answer. Hence SVM can be used to summarize the information contained in a data set by the SV produced. If the data is linearly separable the following equality will hold,

$$\|w\|^2 = \sum_{i=1}^{l} \alpha_i = \sum_{i \in SVs} \sum_{j \in SVs} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \qquad (A.13)$$

Hence from Equation (A.13), the VC dimension of the classifier is bounded by,

$$h \leq \min\left[\mathrm{R}^2 \sum_{\mathrm{i} \in \mathrm{SVs}} , \mathrm{n}\right] + 1 \qquad (\mathrm{A}.14)$$