Summer 8-2016

Robust Algorithms for Estimating Vehicle Movement from Motion Sensors Within Smartphones

Ilyas Ustun
Old Dominion University, ilyasustun@gmail.com

ROBUST ALGORITHMS FOR ESTIMATING VEHICLE MOVEMENT FROM MOTION SENSORS WITHIN SMARTPHONES

by

Ilyas Ustun
B.S. June 2008, Fatih University, Turkey

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

MODELING AND SIMULATION

OLD DOMINION UNIVERSITY
August 2016

ABSTRACT

ROBUST ALGORITHMS FOR ESTIMATING VEHICLE MOVEMENT FROM MOTION SENSORS WITHIN SMARTPHONES

Ilyas Ustun
Old Dominion University, 2016
Director: Mecit Cetin

Building sustainable traffic control solutions for urban streets (e.g., eco-friendly signal control) and highways requires effective and reliable sensing capabilities for monitoring traffic flow conditions so that both the temporal and spatial extents of congestion are observed. This would enable optimal control strategies to be implemented for maximizing efficiency and for minimizing the environmental impacts of traffic. Various types of traffic detection systems, such as inductive loops, radar, and cameras have been used for these purposes. However, these systems are limited, both in scope and in time. Using GPS as an alternative method is not always viable because of problems such as urban canyons, battery depletion, and precision errors.

In this research, a novel approach has been taken, in which smartphone low energy sensors (such as the accelerometer) are exploited. The ubiquitous use of smartphones in everyday life, coupled with the fact that they can collect, store, compute, and transmit data, makes them a feasible and inexpensive alternative to the mainstream methods. Machine learning techniques have been used to develop models that are able to classify vehicle movement and to detect the stop and start points during a trip. Classifiers such as logistic regression, discriminant analysis, classification trees, support vector machines, neural networks, and Hidden Markov models have been tested. Hidden Markov models substantially outperformed all the other methods. The feature quality plays a key role in the success of a model. It was found that, the features which exploited the variance of the data were the most effective.

In order to assist in quantifying the performance of the machine learning models, a performance metric called Change Point Detection Performance Metric (CPDPM) was developed. CPDPM proved to be very useful in model evaluation in which the goal was to find the change points in time series data with high accuracy and precision.

The integration of accelerometer data, even in the motion direction, yielded an estimated speed with a steady slope, because of factors such as phone sensor bias, vibration, gravity, and other white noise. A calibration method was developed that makes use of the predicted stop and start points and the slope of integrated accelerometer data, which achieves great accuracy in estimating speed.

The developed models can serve as the basis for many applications. One such field is fuel consumption and CO_2 emission estimation, in which speed is the main input. Transportation mode detection can be improved by integrating speed information. By integrating Vehicle (Phone) to Infrastructure systems (V2I), the model outputs, such as the stop and start instances, average speed along a corridor, and queue length at an intersection, can provide useful information for traffic engineers, planners, and decision makers.

This dissertation is dedicated to my family

for their endless support and for their faith in me.

ACKNOWLEDGMENTS

Over the course of my doctoral studies, I have received support and encouragement from a great number of individuals. First, and foremost I would like to thank Dr. Mecit Cetin who has been a mentor and a guide to me.

I would also like to mention the committee members who have guided me with their comments and suggestions: Dr. Roland Mielke, Dr. Tamer Nadeem, and Dr. Jiang Li.

I would like to thank my friends and colleagues who were with me throughout my curricular and extra-curricular activities. I could not forget to mention the Transportation Engineering Students Organization, and I am grateful that I was elected as the founding president.

I would like to give my deepest thanks to my mother, father, and wife. Their prayers, encouragement, and love helped me a lot during this long journey. …and finally, I would like to thank my little son Musab Omer as well, for all the joy he has brought into our family.

TABLE OF CONTENTS

Page

LIST OF TABLES viii
LIST OF FIGURES x

Chapter

1. INTRODUCTION 1
   1.1 Smartphones and Accelerometers 3
   1.2 Objectives 6
   1.3 Dissertation Outline 7

2. LITERATURE REVIEW 8

3. MACHINE LEARNING METHODS 16
   3.1 Support Vector Machines 18
   3.2 Neural Networks 27
   3.3 Hidden Markov Models 31

4. DATA COLLECTION 38
   4.1 Data Analysis and Feature Creation 43
   4.2 Feature Selection 49

5. METHODOLOGY 59
   5.1 Assessing Model Accuracy 60
   5.2 Experiments and Results 73
   5.3 Analysis of the Results 93

6. SPEED ESTIMATION 98
   6.1 Algorithm for Estimating Speed 98

7. CONCLUSIONS 107
   7.1 Potential Applications of Vehicle Motion Detection 110

REFERENCES 114

APPENDICES 118
   A: PREVIOUS METHODOLOGY 118
   B: PROBLEMS RELATED TO SENSORS 134
   C: HMM VALIDATION 138

VITA 142

LIST OF TABLES

Table Page

1. Aggregate summary of datasets used. 39
2. Classification Error and Change Point Detection Performance Metric Results 63
3. Classification Error and Change Point Detection Performance Metric Results 64
4. Classification Error and Change Point Detection Performance Metric Results 65
5. Error function values of time differences up to 20s when the values are divided by 20 67
6. Parameters that were used in the exhaustive search of RBF SVM. 77
7. Validation of RBF SVM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063....78
8. Testing of RBF SVM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063....78
9. Validation of RBF SVM for Driver 2 - Toyota Camry 2012 - Lg G4....78
10. Testing of RBF SVM for Driver 2 - Toyota Camry 2012 - Lg G4....79
11. Validation of RBF SVM Driver 3 - Mazda 3 - Samsung Galaxy S3....79
12. Testing of RBF SVM Driver 3 - Mazda 3 - Samsung Galaxy S3....79
13. The window sizes, and number of delays used in NARX exhaustive search. 82
14. Validation of NN for Driver 1 - Toyota Prius 2007 - Motorola XT 1063. 83
15. Testing of NN for Driver 1 - Toyota Prius 2007 - Motorola XT 1063. 83
16. Validation of NN for Driver 2 - Toyota Camry 2012 - Lg G4. 83
17. Testing of NN for Driver 2 - Toyota Camry 2012 - Lg G4. 84
18. Validation of NN Driver 3 - Mazda 3 - Samsung Galaxy S3. 84
19. Testing of NN Driver 3 - Mazda 3 - Samsung Galaxy S3. 84
20. Binning the continuous features into discrete values....86
21. Validation of HMM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063. 87

Table Page

22. Testing of HMM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063. 87

23. Validation of HMM for Driver 2 - Toyota Camry 2012 - Lg G4. 89

24. Testing of HMM for Driver 2 - Toyota Camry 2012 - Lg G4. 89

25. Validation of HMM Driver 3 - Mazda 3 - Samsung Galaxy S3. 91

26. Testing of HMM Driver 3 - Mazda 3 - Samsung Galaxy S3. 91

27. F1 scores of the validation trips 93

28. F1 scores of the testing trips 93

29. Window sizes of the best models, as found in different ML techniques. 93

30. Prius HMM best model testing output for different threshold values................................96

31. Camry HMM best model testing output for different threshold values. 96

32. Mazda HMM best model testing output for different threshold values. 97

33. The RMSE of estimated speed w.r.t. GPS speed. 106

34. Saturn Ion 2007 Trip Info. 118

35. Features used and their explanation. 120

36. The consolidated testing error values of 7 trips. 121

37. The SVM error results for each feature of test trip 1. 124

38. The transition and emission probability matrices 127

39. The average errors of the 7 test trips using HMM for state estimation............................128

40. The HMM error results for each feature of test trip 1. 130

41. The best linear SVM by using grid search over several C values....................................132

42. HMM Trained State Transition Matrices 138

LIST OF FIGURES

Figure Page

1. The X, Y, and Z axes of a phone.....5
2. SVMs for two-category linearly separable classification [39].....20
3. Linearization of nonlinear and nonseparable data in higher dimension space [39].....24
4. Slack variables allow for not-perfect separation, provide some tolerance [39].....25
5. A feed forward neural network model with 2 hidden layers.....27
6. The model for treating independent data points [41].....31
7. The model for first order Markov chain [41].....32
8. Hidden Markov model, also known as state space model [41].....33
9. The schema for data collection and processing.....39
10. Map of Hampton Roads and trips by three different drivers.....40
11. Map of Hampton Roads and the average speeds of the three drivers.....41
12. The X, Y and Z axes of a phone.....42
13. Three-axis accelerometer data collected by the smartphone for a trip.....43
14. The boxplot of the magnitude of the raw data.....45
15. Center-based features and percentiles extracted from the data with a sliding window.....46
16. Extreme-value features extracted from the data with a sliding window.....46
17. Variation-based features extracted from the data with a sliding window.....47
18. Extracting features and preparing data for analysis.....48
19. Ranked feature scores w.r.t. alpha and the test for Toyota Camry, LG G4.....52
20. Ranked feature scores w.r.t. alpha and the test for Mazda 3, Samsung Galaxy S3.....53
21. Ranked feature scores w.r.t. alpha and the test for Toyota Prius, Moto X1063.....54

Figure Page

22. ReliefF feature weights for each K nearest neighbor for Toyota Prius..............................56

23. ReliefF feature weights for each K nearest neighbor for Toyota Camry. 57

24. ReliefF feature weights for each K nearest neighbor for Mazda3. 57

25. Example 1: Importance of using the right performance metric for model selection. 63

26. Example 2: Importance of using the right performance metric for model selection. 64

27. Example 3: Importance of using the right performance metric for model selection. 65

28. The error function values corresponding to time differences. 66

29. The schema for model development and testing...75

30. NARX model representation with delayed feedback loops and delayed inputs [43]. 81

31. Closed and open loop architectures, respectively [43]...81

32. Closed loop NARX with 5 tapped delay lines (MATLAB NN Toolbox). 82

33. HMM Representation. Z is the hidden states, X is the observations vector [41]................85

34. Prius state estimation of test trips. Red: Prediction, Green: Observed. 88

35. Camry state estimation of test trips. Red: Prediction, Green: Observed.............................90

36. Mazda state estimation of test trips. Red: Prediction, Green: Observed.............................92

37. Percentage True, False, and Miss values w.r.t. the threshold and the vehicle type. 97

38. The route and speed of a Toyota Camry driven from Norfolk to Virginia Beach. 99

39. Raw accelerometer, magnitude of acceleration, and speed of GPS and OBD. 100

40. Speed estimations for each axis using the kinematic equation of $Vf = Vi + a\Delta t$. 102

41. Calculating the slope, and finding the beginning and ending speeds of segments. 104

42. Calibrated speed estimation on each axis using the observed change points. 104

43. Calibrated speed estimation on each axis using the predicted change points. 105

Figure Page

44. Acceleration data collected by a smartphone for each transportation type.......................111

45. Ubiquitous use of smartphones could be utilized improve traffic. 113

46. Geographic boundaries of the trip data collected in Norfolk, VA 119

47. The SVM predictions of the same test trip of Saturn Ion using different features. 124

48. Lin. SVM, *C*=1, best models based on classif. error (green), and CPDPM (red). 125

49. The summary of the classification process from start to end. 127

50. The state estimation of the first test trip from Saturn Ion using different features. 129

51. The best HMMs. Classif. error (green), CPDPM (red) agree on the same feature. 131

52. The best SVMs based on Classification error (green), CPDPM (red). 132

53. Best HMM results. Class. error (green), CPDPM (red) agree on the same feature. 133

54. RBF SVM state estimation. 133

55. Loss of GPS signal several times, not reaching zero when the vehicle has stopped. 135

56. Loss of OBD signal, and the erratic behavior of the accelerometer data. 135

57. Some large variation in accelerometer data when the vehicle is stopping. 136

58. GPS signal not going below 10 mph even when the car is stopping. 136

59. GPS signal hanging around 3 mph while the car is in total standstill. 137

60. Change of orientation around 8 min mark, and lack of GPS at the beginning. 137

61. Prius state estimation of validation trips. Red: Prediction, Green: Observed. 139

62. Camry state estimation of validation trips. Red: Prediction, Green: Observed. 140

63. Mazda state estimation of validation trips. Red: Prediction, Green: Observed. 141

CHAPTER 1

INTRODUCTION

Building sustainable traffic control solutions for urban streets (e.g., eco-friendly signal control) and highways requires effective and reliable sensing capabilities for monitoring traffic flow conditions so that both the temporal and spatial extents of congestion are observed and the optimal control strategies are implemented for maximizing efficiency and minimizing the environmental impacts of traffic. In order to do so, agencies responsible for managing and operating transportation networks install various types of traffic detections systems, such as inductive loops, radar, and cameras. These detection systems require a large investment and provide traffic information only for the specific locations where they are installed. An alternative option for collecting data about vehicular movements in traffic is equipping vehicles with tracking technologies, such as GPS. These so-called probe vehicles can provide information about traffic conditions along their travel paths. Since mobile consumer devices, such as tablets and smartphones, are equipped with GPS sensors, such devices may serve as a traffic data collection platform[1] .

Recent studies show that the market share of smartphones continues to grow, with over 60 percent U.S. mobile subscribers owning smartphones as of December 2013. Smartphones have matured as a computing platform and are now equipped with multiple low-energy sensors including a gyroscope, compass, accelerometer, proximity sensor, and ambient light sensor. These low energy sensors are on all the time and perform certain tasks (such as screen rotation). This research aims to exploit the vast amount of data that can be collected by these sensors, which are always on and are collecting data in the background. Even though researchers have

[1] IEEE Transactions and Journals style is used in this thesis for formatting figures, tables, and references.

used data from cell phones before, they have relied on GPS, which has the following disadvantages:

- GPS sensors are energy expensive [1-3]. The GPS sensor, when enabled, consumes significant energy and depletes the battery of the mobile device quickly.
- GPS has problems receiving a signal when surrounded by high structures such as skyscrapers, high mountains, or when in a tunnel [4].
- GPS localization precision is low [5].

This research is focused on developing methods to estimate vehicle movement and operating mode (e.g., being in motion, and stopping) from the raw data collected by the low-energy sensors, such as the accelerometer. For example, speed estimation can be done using the acceleration collected in the motion direction of the vehicle. However, because of the inherent bias in the sensors, the effect of gravity, and the random noise due to environmental factors, the error will accumulate once the acceleration is integrated to estimate speed. This will lead to unrealistic speeds after a very short time period. The estimated speed needs to be adjusted at certain points. The estimated stop or motion start points will serve to calibrate the speed estimation. For example, if the vehicle stopping instances and the standstill durations can be detected, the speed estimation at these intervals can be brought down to zero.

Capturing the stopping events and the speed of vehicles on urban streets can support various applications. If such information is collected from a large number of vehicles and is aggregated at a traffic management center, useful information about the transportation network can be extracted. For example, knowing where vehicles stop more frequently and for longer durations help identify congested segments. In addition, the number of stops a vehicle makes and its speed are directly related to the fuel consumption and emissions. If such congested segments are along signalized arterials, signal timing adjustments can be made to improve vehicle stops, delays, and emissions. The route information based on average speed and gas emission levels can be used to calculate and inform people about speed-based shortest paths and fuel efficiency-based shortest paths from their origins to destinations.

1.1 Smartphones and Accelerometers

Recent studies [6-8] show that smartphone market penetration has continued to grow throughout the years. As of December 2013, more than three out of five Americans (65%) are carrying smartphones, up from 44 percent in 2011 and from just 19 percent in 2009 [9]. The September 2012 and June 2013 surveys conducted by Pew Research Center show that more than 45 percent, and 56 percent of American adults own a smartphone, respectively [10, 11]. It is estimated that there will be more than 192 million smartphone users in the U.S. by the year 2016 [12]. As of 2012, phones operating on the Android platform are the most prevalent type (51.8%) of smartphone, followed by iOS (34.3%) [7]. This ratio did not change much during 2013, when the Android share stayed the same and the iOS share increased to 40% [13].

Mobile devices have become more sophisticated over the years and are now equipped with several sensors which make them useful data collection devices. The sensors that are present in the smartphones are:

1- Global Positioning System (GPS)

2- Vision (Front camera, back camera)

3- Audio (Microphone)

4- Light (Proximity)

5- Acceleration (Accelerometer, Gyroscope)

6- Temperature

7- Direction (Magnetic compass or magnetometer)

Since the smartphone penetration is already at high levels and continues to grow rapidly, making use of smartphones for scientific purposes will provide new and cheap ways for collecting, processing, and sending data. The ubiquitous use of smartphones will provide the ability to collect data in large amounts, over longer periods, and in a cheap and passive way without causing any trouble or expense to the user, since they already own it and are carrying it.

Smartphones are capable of many tasks such as:

1- Providing sensor data such as GPS, accelerometer, gyroscope, magnetometer etc.

2- Storing data

3- Using data and making computations

4- Sending and receiving data via Bluetooth, Wi-Fi, and/ or 3G or 4G connection

The reasons for using Android based operating system can be listed as:

1- It is free

2- It is open source

3- It is the leading operating system among smartphone operating systems, with approximately 50% market share.

There are many studies that use devices with embedded accelerometers and other sensors that track people to collect data and estimate daily activities. However, sensors are not practical to use in everyday life since they involve physical accelerometers or other equipment being worn on different parts of the body. While this can be done in a laboratory setting or in specialized conditions, and for a short term (such as using a heartbeat sensor to test a patient for its heart conditions at a hospital or at his home for a day or two), it definitely cannot be used in everyday life for long durations. Smartphones provide a ubiquitous and cheap way to achieve long term data collection.

1.1.1 Smartphone Accelerometers

An accelerometer is a sensor that returns a real valued estimate of acceleration along the X, Y, and Z axes which are shown Fig. 1. These values can be used to estimate velocity and displacement. The accelerometer used in smartphones are DC accelerometers and are capable of measuring “static” acceleration. This means that the value reported by the accelerometer is not the classic definition of time rate change of velocity, but is a function of the force exerted on it [14]. For example, even when the phone is at rest on a table, one of its axes will report +g (positive gravity) as acceleration resulting from the opposing force. The fact that the accelerometer sensor can detect the effect of Earth’s gravity coupled with the gyroscope can be very useful. By making use of these sensors phones can be used as: a steering wheel in car racing games, a racket in tennis games, a sleeping behavior tracker, a pedometer that counts the number of steps taken, a slider which can slide through pages without touching the phone by just tilting it upward or downward, and in many more ways [15, 16].

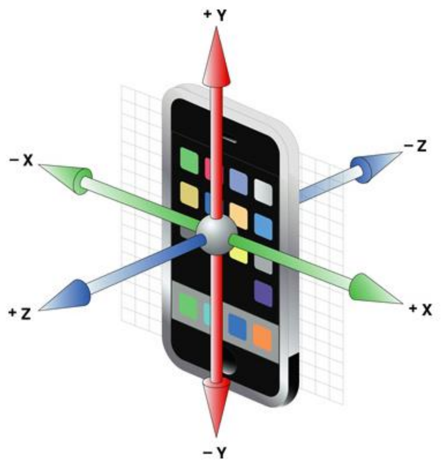


Fig. 1. The X, Y, and Z axes of a phone.

Accelerometers offer a number of desirable features in monitoring the human movement. First, they respond to both frequency and intensity of movement, and so they are superior to actometers or pedometers, which are attenuated by impact or tilt. Second, some types of accelerometers can be used to measure tilt as well as body movement, making them superior to motion sensors that have no ability to measure static characteristics. Third, enhancements in microelectromechanical systems (MEMS) technology have made possible the manufacture of miniaturized, low cost accelerometers [17]. Another feature of the accelerometer is its high degree of reliability in measurement, with little variation over time [18].

Recently, physical activity detection has become another use of smartphones with the purpose of health monitoring. Because of their light weight, small size, and low energy consumption, the accelerometer sensor embedded wearable systems or smartphones represent one of the emerging fields that is becoming more and more frequently used, especially in patient activity monitoring and in the classification of the ambulatory environment. Detecting physical activities correctly can also help people in their daily lives, since the phone can trigger or block certain apps during certain activities. For example, when it is detected that a person is jogging, a pre-programmed type of music can start playing, and any incoming call can be directed to voicemail [19].

1.2 Objectives

The specific objectives of this dissertation are:

- to develop robust algorithms that detect when the vehicle stops or starts to move based on the smartphone accelerometer data,

- to investigate which feature(s) provide the most useful information for detecting stops more accurately,
- to evaluate the performance of the algorithms based on field data collected by different drivers, vehicles and phones,
- to develop a model that can estimate speed based purely on accelerometer data, without the help of GPS,
- to develop a novel performance metric which will assist in model evaluation. The error metric should be able to quantify performance based on accuracy - whether true points are found, and precision- about how close the estimations are to the true points. The error metric should be able to penalize any missed true points, and also be able to penalize any false positives heavily.

1.3 Dissertation Outline

In Chapter 2, a literature survey is presented. It mainly focuses on previous studies that have used smartphone, accelerometer, and other sensor data in their research. Chapter 3 is a concise overview of some of the machine learning methods that are used in the model development. Information about the dataset and how it is collected can be found in Chapter 4. Chapter 5 is about the methodology applied and the experimental results obtained in detecting the vehicle movement. The novel change point detection performance metric is also introduced in Chapter 5. Chapter 6 presents a method for estimating speed from acceleration obtained with accelerometer data of the smartphone. Chapter 7 concludes the dissertation with the summary of findings, results, contributions, and possible applications of the developed models.

CHAPTER 2

LITERATURE REVIEW

Activity recognition and transportation mode identification are two major fields in which accelerometer data have been widely used. Many of the early studies included wearable accelerometers in order to detect activity and transportation mode. In the past, very few studies have used mobile phones to collect data. This situation is rapidly changing, and the presence of smartphone as a data collection tool can be seen more frequently in recent literature.

In some applications, GPS is also used to enhance the classification accuracy for transportation mode detection. GPS is used mainly to provide speed information. The added information of speed can help distinguish between certain modes such as bicycle, motorcycle, and a passenger vehicle. Some researchers have also tried to integrate the Geographical Information Systems (GIS) to the model in order to further increase accuracy and/or analyze human behavior when subject to different conditions, such as being close to a metro station, being close to shops, etc.

Accelerometers have been used to monitor a range of different movements, including gait, sit-to-stand transfers, postural sway, and falls. They have also been used to measure physical activity levels and to identify and classify movements performed by subjects [17]. The physical activities of people such as walking, running, sitting, watching TV, scrubbing, brushing teeth, and climbing are some that can be mentioned [20, 21]. Accelerometers have also been used as motion detectors for body positioning and posture sensing [21]. GPS in combination with an accelerometer has been used in recent research to detect physical activities [22-25].

Bao et al. used five biaxial accelerometers worn on different parts of the body [20]. Mean, energy, frequency domain entropy, and correlation of acceleration data were calculated to be used in several classifier algorithms. Decision tree classifiers were the best performing among the algorithms. A key finding here is that some activities are recognized well with subject independent training data, while some others are subject dependent, i.e., they perform better when both training and testing is done only on the same subject’s data. When only two out of five accelerometers were used, the performance dropped only slightly. Accelerometers worn on the thighs were found to be the most useful in detecting activity.

Automated recognition of human daily activities from wearable sensor signals has many applications in health care, sports, and aged care. Wang et al. proposed a Hidden Markov model (HMM)-based recognition method to recognize six human daily activities from sensor signals collected from a single waist-worn tri-axial accelerometer [26]. The HMM performed very well in recognizing the activities.

Patient care, chronic disease management, and the well-being of aged people are some areas in which automatic recognition and classification of a person’s activity are of critical importance [27]. Ermes et al. used signal features calculated for each second of the data collection. Time-domain features calculated were mean, variance, median, skew, kurtosis, 25% percentile, and 75% percentile. Frequency-domain features included the estimation of power of the frequency peak and signal power in different frequency bands. Speed was calculated from GPS location data. Spectral entropy was also used. The authors used custom decision trees, automated decision trees, and artificial neural networks as methods to classify different activities. In their proposed hybrid method, they combined the best qualities of the custom decision tree model and neural networks.

Lester et al. used acceleration data together with GPS to detect the mode of transport used by the passenger and also to extract trip information and dwell locations [28]. Activity inference provides information on what the user is doing, and the localization (such as cell-tower/WIFI localization and/or GPS) provides location. They also make use of GIS overlay, so that, together with GPS data, they can more accurately classify whether the passenger is in a car or a bus by checking the stop and go behavior together with the bus stops along the route. The research was done in 2008 when mobile phones were just beginning to evolve into small computers, i.e. smartphones. iPhones and Nokia N95 are mentioned in the paper; however, since the technology was very new, the authors preferred to use a device called Mobile Sensing Platform (MSP) instead. MSP combines an Intel XScale processor with an accelerometer, barometric pressure sensor, light sensors, humidity sensors, microphone (not used in this experiment), GPS, and storage capacity. The participants of the experiment wore the device on the waist and entered information about the current environment (in a bus, car, at a café, etc.) into a cellphone. This type of research can be useful in the analysis of how the behavior of a person is affected in the proximity of a built environment. For example, how does the proximity or lack thereof of metro station, bus stop, shops etc. affect the behavior of an individual? Based on the options available, do people tend to use public transit or to drive to work? This kind of knowledge can provide valuable information to city management, transit planners, road network planners, and the like.

Jennifer et al. have used smartphone accelerometer sensors to detect walking, jogging, sitting, standing, climbing upstairs, and climbing down stairs [19]. Their study is part of research project called Wireless Sensor Data Mining, which aims to collect the sensor data from smart phones and other mobile devices (e.g., tablet computers, music players, etc.) and mine this sensor data for useful knowledge [29]. Also, their intention is to bring attention to mining wireless sensor data in the area of activity recognition. A window size of 10 seconds is used for data extraction. This window size quite large and is not suitable for a real-time or near-real-time recognition process. 29 people volunteered for data collection, and they carried their phone in their pants leg pocket. They did nothing regarding orientation correction; the x y and z axis readings from the phone were directly used in feature extraction. A total of 43 features were extracted from the data, all from six basic features. In many cases, the X, Y, and Z directions had their own feature calculated separately. The following features were calculated for each window: average, standard deviation, average absolute difference, average resultant acceleration, time between peaks, and binned distribution. Here, the noteworthy feature is the average resultant acceleration, which is defined as the average of the square root of the sum of the values of each axis squared over the example duration. Example duration refers to the data points that are within the window size. So, in essence, they have used the orientation invariant total acceleration.

$$a_{Tot} = \sqrt{a_x^2 + a_y^2 + a_z^2} \tag{1}$$

Different machine learning techniques such as decision trees (J48), logistic regression, and multilayer neural networks present in the WEKA data mining suite were applied. The results show that climbing up and down stairs are the most difficult to recognize, while sitting and standing are easily recognized, thanks to the effect of earth's dramatically gravity changing the values of acceleration in each axis. The results also agree with other research, where it was found that an accelerometer attached to the thighs provided the best results [20].

Chen et al. used smartphone accelerometer data to recognize the states of the users (whether they were in-vehicle or pedestrian) [30]. The aim of the study was to trigger the WiFi of the smartphone once it sensed that the user was in vehicle, so that the smartphone could connect to other devices in the car and send and receive data. Participants in the research were required to carry the smartphones in their pockets while travelling. Fast Fourier Transform (FFT) components were calculated for each window, and energy and frequency domain entropy features were extracted by using these FFT components. When the energy and entropy results are plotted onto a scatter graph, the in-vehicle and pedestrian states form distinct clusters. The proposed method is able to detect the states with high accuracy and low power consumption. The authors do not indicate exactly which pocket the phone was carried in [30]. Also, they do not give information about how they dealt with the three-axis data while obtaining the features.

The conventional methods of collecting information about how people go to work, go shopping, or how much exercise they do per day include paper-based, phone-based, and internet-based surveys. However, these methods are a burden, both on the surveyor and the surveyed, and they might involve errors, or surveys might not be filled out in time or might be filled in a rush, so results might be unreliable.

During the past years, GPS based technology has been used successfully in collecting activity travel diary data. Professional organizations are now discussing the possible replacement of traditional travel survey methods by GPS data collection, since this will have the added benefit of reducing respondent and researcher burden. However, GPS-collected data is not exact and may have errors in it. When traveling underground or in urban canyons, GPS signals may not be received. Now, if you add the errors resulting from using these GPS traces in algorithms to extract travel information, the outcome may not be very accurate.

GPS-based detection methods rely on speed and time information. Use of speed may be misleading if the feasible range of different modes have similar speed distributions. For example, fast walking and slow biking, bus or car on congested road, and light rail might be misclassified, based solely on speed data. When speed is not sufficient enough to discriminate between transportation modes, or when there are problems with the GPS signal, other relevant data would be useful. Smartphone sensor data provide such an opportunity by introducing additional and more reliable information. Accelerometers, gyroscopes, magnetometers, etc. provide additional information that can be utilized to detect the mode of transportation. Accelerometer data is one of the most used sensors, together with GPS, in the field of activity and mode recognition. The advantage of accelerometer over GPS is that it is not susceptible to the problems of GPS mentioned above. It can capture data independent of the exterior situation [31].

The studies in this field are partially successful in detecting the transportation modes. All of these approaches rely basically on speed and time information, which are extracted from the GPS traces. As stated above, GPS signals may not be received accurately and may sometimes be lost. As a result, this will affect accuracy.

Feng et al. have used Bayesian Belief Network in order to detect different transportation modes [31]. A Bayesian Belief Network is a graphical representation of the conditional probability and causality relationships between variables. This is a dynamic structure that has the ability of improving over time, as more samples are collected. The researchers have also used location-based variables such as latitude and longitude obtained from the GPS to calculate the speed and distance. They have developed an algorithm based on the Haversine formula, which is used to calculate great circle distances between two points on a sphere. This is achieved by using the latitude and longitude of the points and relating the sides and angles of spherical triangles. The researchers have found that accelerometer-only detection rates were better than GPS-only detection, and that the results were best when both accelerometer and GPS were used. Statistical properties of acceleration such as mean, variation, energy, and correlation among three axes may help to differentiate between different transport modes and activity types. When different mode types or activities are analyzed, it is easy to see that they have different fluctuations and different ranges for the acceleration values. However, relying only on representative statistics is not very effective, as shown by different studies. Although these statistics provide a good amount of information, more advanced techniques are needed to fully differentiate between modes.

The CO2GO project involves the development of a smartphone application that can detect the transport mode and estimate CO2 emissions based on the mode the passenger is in [32, 33]. The purpose of CO2 estimation is to create environmental awareness, i.e. the passenger will be able to see his or her carbon footprint and then act accordingly. The acceleration sensor and GPS are used. An algorithm uses a Decision Tree classification method to determine the user's transportation mode using the accelerometer. A second algorithm computes the distance travelled, using the GPS and internet map services. The computations are made on the smartphone, which is based on an android operating system. The output is presented to the user in the amount of CO2 emissions, thus making it understandable and useful. The main novelty of this project is that the orientation problem is overcome by using the square root of the sum of each axis squared, which makes the orientation of the phone irrelevant. Automatic mode recognition, no manual entry need from the user regarding position, orientation, or transport mode make the application very useful. Accelerometer, GPS, and online maps are queried sparsely, decreasing the amount of battery used.

Cooper et al. have used accelerometer and GPS data to investigate the level and location of the physical activities of children walking to school [23]. The study found that the mean accelerometer per minute before school was 43% higher for children walking to school than for those coming to school by car. The study provided evidence that walking contributes to higher total physical activity in children. The reasons behind the choice of a particular mode of transportation are important for both city design and health of people, as walking is found to be a significant contributor to daily physical activity.

Troped et al. have done a pilot study activity mode detection of walking, jogging/ running, bicycling, inline skating, and driving by the use of both GPS and accelerometer [24]. The experiments were carried out by ten adults who wore a GPS unit and accelerometer simultaneously during the aforementioned activities. Discriminant function analysis was used to identify a combination of variables derived from the accelerometer and GPS speed that best classified the mode. Walking and bicycling minutes were correctly classified most frequently (96%). The study has proven that the combination of GPS and accelerometer has improved the detection rates.

In order to understand the activities and transportation modes that people prefer within a built environment alongside GPS and sensor data such as accelerometer, geographic information systems (GIS) can be very useful. Oliver et al. have studied the feasibility of the combination of GPS, GIS, and accelerometer to understand the transport-related physical activity of adults in a built environment [22]. The accelerometer and GPS data extracted from forty adults were integrated into GIS database so that physical activity intensity, GPS speeds, and routes traveled could be analyzed. The results show that integrating GPS and accelerometer data into a GIS database can be very promising in understanding the transport-related physical activity of individuals in built environments.

CHAPTER 3

MACHINE LEARNING METHODS

Machine learning (ML) is defined in various ways by different scholars. Arthur Samuel gives one of the first definitions of ML as: "Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed." This can be understood as the automation of learning by computers. The key part missing in this definition would be data. The way ML learns and improves on the predictions is through data. Tom Mitchell gives a nice definition which summarizes the whole process of machine learning [34]. In his words, machine learning is described in this way: "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

ML is a subfield of artificial intelligence that is concerned with the development of algorithms which allow computer programs to learn from data or experience [35]. These algorithms are useful in situations where analytic or explicit model development is not possible, and where there are data available. These datasets are used to train a specific model using the algorithm where it learns from the data to estimate the underlying true model. Once the training is finished and the model is ready, new unused datasets are used to test the accuracy of the model, also referred to as generalization error.

Classification is the process of learning from data and separating them into groups. A dataset consists of rows (examples, observations) $\boldsymbol{x_i}$ each with p dimensions (the columns, variables, features), and the corresponding classes y_i which are also called labels. Subscript i stands for each example in the data set. A pair ($\boldsymbol{x_i}$, y_i) is called a training example where $i = \{1,2,...N\}$ for a set with N observations. In the context of the movement of a vehicle, each y_i

would represent whether the vehicle at instance *i* is at a standstill or is in motion. An instance can represent a single time point at which a measurement is taken or a time interval over which a feature is calculated. In the first case, the corresponding x_i is an array of independent variables such as the recording obtained from the sensors and the GPS at each instance. In the latter case, however, each x_i would be an array of the features calculated such as range, and standard deviation over a window that spans several neighboring points. What the classification algorithm is trying to do, is to learn a function or hypothesis $h_w(\mathbf{x})$ by using the training set, such that this hypothesis is able to predict *y* of new input data with highest accuracy possible. The *w* represents the parameters or weights that are found by the algorithm.

There are different methods to come up with a hypothesis. If both features (input) and labels (output) are used during the learning process, it is called supervised learning. If only the features are used and the labels are unknown, it is called unsupervised learning. In unsupervised learning, the algorithm tries to group data into clusters. The algorithms used in this research are all supervised learning methods since the ground truth of the outcome states are known with the help of the GPS speed.

Classifiers such as logistic regression, discriminant analysis, classification trees, support vector machines, neural networks, hidden Markov models, and more have been tested. Most of these algorithms did not perform well. The three best performing algorithms were found to be support vector machines, recurrent neural networks, and hidden Markov models. These three techniques have been used in detecting the change points during a trip of a vehicle, and are explained in the remainder of this chapter.

3.1 Support Vector Machines

Developed from statistical learning theory of Vapnik and Chervonenkis, the support vector machine is a generalization to the *Generalized Portrait Algorithm* developed in Russia in the 1960s [36]. Support vector machines have been based upon the Vapnik Chervonenkis (VC) theory, which is a framework that characterizes the properties of learning machines that enables them to generalize well to unseen data [37]. The generalization of SVM is achieved through controlling the sum of the training data error rate and the capacity or complexity of the learning machine, which is measured by its VC dimension. Details about the VC dimension can be found in Vapnik's book [36].A smaller VC dimension leads to a smaller confidence interval, but causes the training error to increase. The Structural Risk Minimization principle is proposed to solve this trade-off, in which the sum of the right hand side of Eq. (2) is being minimized. The trade-off between the complexity of the decision rule and the generalization error can be controlled by changing the parameter C. A large value of C will allow for larger weights, a more complex learning machine, and a low error training error. But the system will likely overfit the training data, hence causing a large testing error. A lower value of C will force the network to have smaller weights and will increase the training error, but will generalize well to unseen data.

$$Test\ Error \leq Training\ Error + Complexity\ of\ Model \tag{2}$$

Support vector machines have a strong ability to approximate linear and nonlinear relationships. SVMs work by mapping the input vectors into some high dimensional feature space using nonlinear functions. The linear separation between two classes is done in this high dimensional space. The properties of the linear decision surface ensure that the network is highly generalizable. One very important idea behind SVMs is that not all of the data points influence optimality as in linear regression, or Naive Bayes, but the points close to decision boundary. These points are referred to as Support Vectors; hence the name Support Vector Machine.

SVMs behave similarly to multilayer feedforward neural network models. One difference is that SVMs are based on structural risk minimization. VC depends on the chosen class of functions, whereas the empirical risk and actual risk depend on the one particular function chosen by the training procedure. Structural Risk Minimization (SRM) consists of finding this subset of functions which minimizes the bound on the actual risk. SRM enables the support vector machine to better generalize, compared to neural networks. Another important difference is that SVMs can always guarantee a globally optimum solution [38]. This is because the objective function given in Eq. (3) of SVM is strictly convex, which means that the Hessian of the objective function is positive definite, which leads to a single global solution.

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\, \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{3}$$

SVMs have applications both in classification and regression. C-SVM variant is the one used for classification which is also used as one of the methods in the motion detection algorithm in this dissertation. The C-SVM will be referred to as SVM from now on for brevity, and will be explained next with a two category linearly separable classification problem. This two variable problem can be generalized to more variables. The training data consists of pairs $\{x_i, y_i\}$, where $i = \{1, 2, ..., N\}$, $y_i \in \{-1, 1\}$, $x_i \in R^2$.

We want to separate these two categories by a decision boundary, as shown in Fig. 2. While doing this, we want to maximize the margin to the closest positive and negative training examples (shown as m/2 in the figure) which gives two parallel separating boundaries, shown as dashed lines.

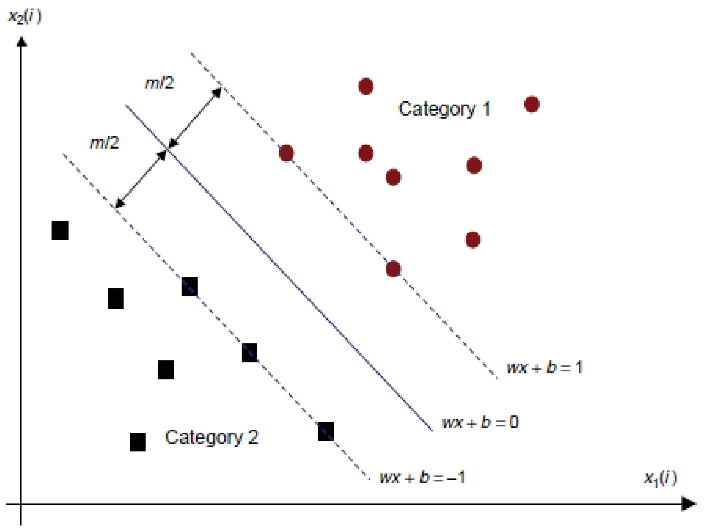


Fig. 2. SVMs for two-category linearly separable classification [39].

These boundaries are in the shape of a line for two-dimensional data, plane for three-dimensional data, and a hyperplane for data with dimensions more than three. In this example, since the data is two dimensional, it will be a line. The separating hyperplane that is at equidistance to the two boundaries is defined as:

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0 \tag{4}$$

Where,

$\boldsymbol{w}$	is the weights vector
$\boldsymbol{x}$	is the p dimensional input vector
b	is a constant

The SVM tries to establish this hyperplane such that the closest points in both categories are separated as much as possible. This is pictured by dashed lines in Fig. 2 and is formulized as follows:

$$\boldsymbol{w^T}\boldsymbol{x} + b \geq +1 \text{ for } y = +1 \ (H_1) \tag{5}$$

$$\boldsymbol{w^T x} + b \leq -1 \text{ for } y = -1\ (H_2) \tag{6}$$

These formulas can be combined into:

$$y_i(\boldsymbol{w^T x_i} + b) \geq 1 \text{ for } \ i = 1,2, \dots, N \tag{7}$$

The minimum distance between two parallel lines is the distance of a line drawn from a point on one line to the point on the other line that is perpendicular to both. Let the lines be:

$$\begin{gathered} Ax + By + C_1 = 0 \\ Ax + By + C_2 = 0 \end{gathered} \tag{8}$$

Then, the distance between them can be expressed as:

$$d = \frac{|C_2 - C_1|}{\sqrt{(A^2 + B^2)}} \tag{9}$$

To find the maximum perpendicular distance between the parallel hyperplanes shown as the lines from Eq. (5) and Eq. (6) are converted to the form in Eq. (10):

$$\begin{gathered} \boldsymbol{w^T x} + (b - 1) = 0 \\ \boldsymbol{w^T x} + (b + 1) = 0 \end{gathered} \tag{10}$$

Thus, the distance between *H₁* and *H₂* is:

$$d = \frac{|b + 1 - (b - 1)|}{\sqrt{(\boldsymbol{w^T w})}} = \frac{2}{||\boldsymbol{w}||} \tag{11}$$

In order to achieve the largest margin between the boundaries, SVM tries to maximize this distance. Maximizing this distance is equivalent to minimizing $||\boldsymbol{w}||^2$ which can also be written $\boldsymbol{w^T w}$ in matrix notation form. Thus, the objective of SVM can be formulated as:

Minimize:

$$\frac{1}{2}\boldsymbol{w^T w} \tag{12}$$

Subject to:

$$y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}} + b) \geq 1 \text{ for } i = 1,2,\dots,m \tag{13}$$

Eq. (12) and (13) form a constrained optimization problem. It is found that, instead of the above optimization problem, Lagrangian formulation of the problem is more convenient to solve. Lagrangian identity can be written as dot products. This feature will come in very handy when generalizing to nonlinear cases [38].

By applying Lagrangian formulation, we get:

$$L_P = \frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} - \sum_{i=1}^{N}\alpha_i\, y_i(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}} + b) + \sum_{i=1}^{N}\alpha_i \tag{14}$$

The L_p stands for Lagrangian-Primal. Since this problem is a convex optimization problem where both the objective function and the constraint set is convex, it allows us to solve it in the dual form. The dual model used is called Wolfe dual and it will solve the problem to yield the same results for $\boldsymbol{w}$, b, and α as the primal function.

It is required that the gradient of the primal form L_P w.r.t. to $\boldsymbol{w}$ and b vanish:

$$\boldsymbol{w} = \sum_{i=1}^{N}\alpha_i y_i \boldsymbol{x}_{\boldsymbol{i}} \tag{15}$$

$$\sum_{i=1}^{N}\alpha_i y_i = 0 \tag{16}$$

Where,

α_i are positive Lagrange multipliers one for each of the inequality constraint in Eq. (13) i = {1,2, …, N}.

Because these are equality constraints in dual formulation, they can be substituted into Eq. (14) which yields the dual form of the Lagrangian problem L_D:

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\, \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \tag{17}$$

We have two different Lagrangian problems that arise from the same objective function. The solution can be found by minimizing the primal Lagrangian L_P or by maximizing the dual Lagrangian L_D. The dual Lagrangian constraints are:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{18}$$

$$\alpha_i \geq 0 \text{ for } i = \{1,2,\dots,N\} \tag{19}$$

The solution for $\boldsymbol{w}$ is obtained by Eq. (15). In the solution, those points for which $\alpha_i > 0$ are called **support vectors** and lie on one of the hyperplanes H_1, H_2. All other points have $\alpha_i = 0$ and lie on the side of hyperplanes H_1, H_2 such that the inequalities of Eq. (5) and (6) hold. Thus, the support vectors play the critical role in forming the decision boundary and in classifying the dataset. All other points do not have any influence on the classifier. Even if the other points were moved around without crossing H_1, H_2 and if training would have been repeated, the same boundaries would have been found [38].

Until now, we have solved the problem for linearly separable case. In many situations, however, problems which are linearly nonseparable or nonlinear need to be solved. To address linear nonseparable cases, slack variables are introduced. To address a nonlinear case, a transformation function ϕ is introduced. Going back to Eq. (12), the nonlinear and nonseparable problem is then formulated as:

Minimize:

$$\frac{1}{2}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} + C\sum_{i=1}^{m} \xi_i \tag{20}$$

Subject to:

$$y_i(\boldsymbol{w}^{\boldsymbol{T}}\varphi(\boldsymbol{x}_{\boldsymbol{i}}) + b) \geq 1\text{-}\xi_i \;\; \forall i \;\in\; m \tag{21}$$

$$\xi_i \geq 0 \; \forall i \;\in\; m \tag{22}$$

Where,

ξ_i is the slack variable, and

$\phi(\mathbf{x_i})$ maps the input features to a new higher dimensional space.

Mapping of the input vector into high dimensional space will also change the dimension of $\boldsymbol{w}$. By doing this, the nonlinear problem is linearized in the new feature space, as can be seen in Fig. 3. The exact form of this mapping function is not needed to be known explicitly, since Kernel functions can be used instead. The slack variable ξ_i is defined as the distance to which point *i* goes beyond the boundary of its category, as shown in Fig. 4. Slack variables allow some variables not to be classified correctly, to some extent. This is again related to maximal margin notion. By sacrificing some accuracy, the SVM achieves a large margin, which benefits the overall problem. However, there is a tradeoff between achieving a large margin and accuracy, and the balance is found by penalizing the tolerance to errors in the cost function (Eq. (20)).

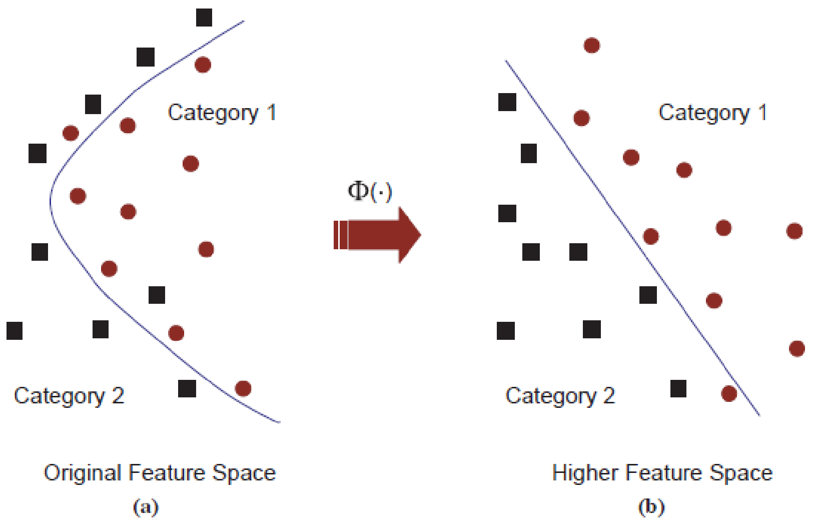


Fig. 3. Linearization of nonlinear and nonseparable data in higher dimension space [39].

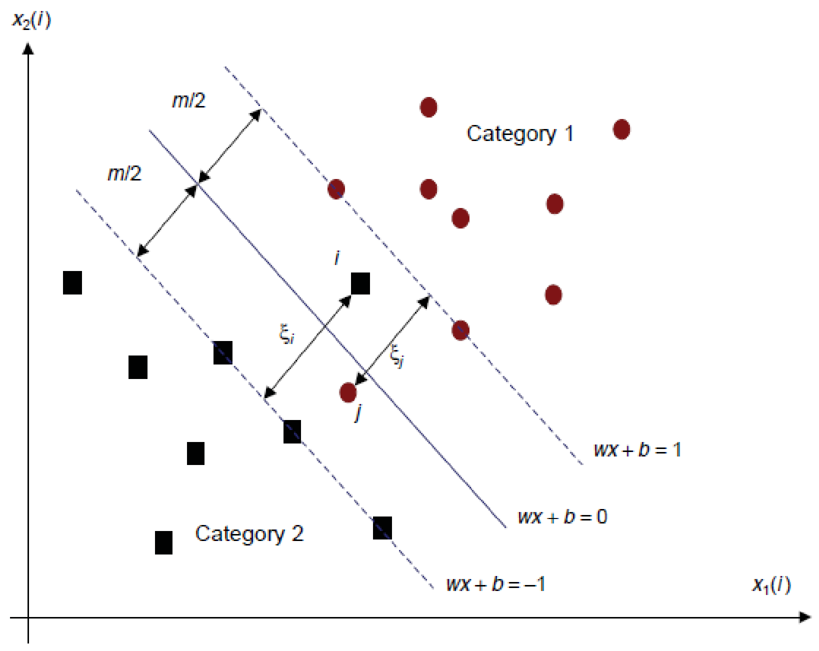


Fig. 4. Slack variables allow for not-perfect separation, provide some tolerance [39].

Given the slack variables, and by mapping the input vectors into higher space, we can now use the dual form with these concepts.

Minimize:

$$L_D = \sum_{i}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \, \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{m} \alpha_i \tag{23}$$

Subject to:

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{24}$$

$$0 \leq \alpha_i \leq C \;\; i = 1, \dots, m \tag{25}$$

Where $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i)\varphi(\boldsymbol{x}_j)$ is the kernel function.

The reason that the slack variables disappeared lies in the fact that we have chosen the penalty function to be:

$$C\left(\sum_{i=1}^{m} \xi_i\right)^k \tag{26}$$

When k=1, neither the slack variable ξ_i, nor their Lagrange multipliers appear in the Wolfe dual problem [38].

After solving for α_i, $\boldsymbol{w}$ can be calculated by:

$$\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \varphi(\boldsymbol{x_i}) \tag{27}$$

In the testing phase, to predict the output $\hat{y}$ given new inputs **x,** Eq. (28) is applied:

$$\hat{y} = sgn(\boldsymbol{w^T}\varphi(\boldsymbol{x}) + b) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b\right) \tag{28}$$

As can be seen in Eq. (28), the mapping function is replaced by kernel function. Thus, the explicit form of the mapping function is not needed. There are many different types of kernel functions, some of which are:

$$K(x, y) = (x^T y + 1)^p \tag{29}$$

$$K(x, y) = \exp(-\gamma \|x - y\|^2) \tag{30}$$

$$K(x, y) = \tanh(\kappa x^T y - \delta) \tag{31}$$

Where polynomial, radial basis, and hyperbolic tangent kernel functions are given respectively. In this dissertation the preferred kernel is the one most used in the literature, the radial basis kernel function.

The model described above is for binary classification. This model can be extended for multi-category cases by adopting a "one-against-one" approach [40]. In this method:

1. K SVM classifiers are trained to distinguish $y = k$ from the rest, for $k = 1,2,\dots,K$.
2. The weight vectors for each classifier is computed $\boldsymbol{w_1}, \boldsymbol{w_2}, \dots, \boldsymbol{w_k}$.
3. Point $\boldsymbol{x}$ is assigned to class k that yields the largest $\boldsymbol{w_k}^T\boldsymbol{x}$ value.

3.2 Neural Networks

The Neural Network (NN) is a black box mathematical model which consists of artificial neurons connected with each other to form a network. Nodes are grouped into layers and form a directed graph where the transition goes from input layer to the output layer, as depicted in Fig. 5. A neuron is the basic element of NN, where computations take place. The term Artificial Neural Network (ANN) is also used to distinguish the model from biological neural network models.

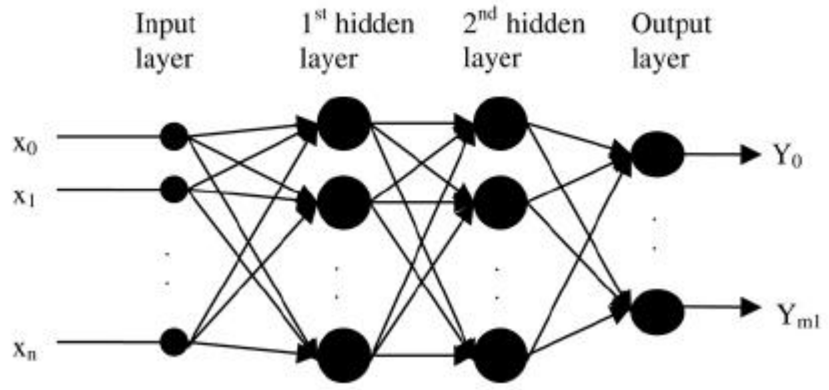


Fig. 5. A feed forward neural network model with 2 hidden layers.

If there are only the input and output layers, this is called a perceptron, developed by Rosenblatt, and it can be regarded as the simplest feed forward neural network. Here, the sum of weighted inputs is transferred to a function which calculates the output. If there is another layer, called the hidden layer, in the network, then our model becomes a multilayer perceptron neural network.

The term NN generally refers to feed forward neural networks unless otherwise stated. There are other NN models such as: Probabilistic Neural Networks, General Regression Neural Networks, Radial Basis Function Networks, Cascade Correlation, Functional Link Networks, Kohonen networks, Gram-Charlier networks, Learning Vector Quantization, Hebb networks, Adaline networks, Heteroassociative networks, Recurrent Networks, and Hybrid Networks. The Feedforward Back Propagation Neural Network (FFBPNN) will be analyzed further, as it is the backbone of the overall NN literature. Also, the dynamic recurrent neural network used in this dissertation for finding the change points during a trip is based on the feed forward neural network.

3.2.1 Feed Forward Back Propagation Neural Network (FFBPNN)

FFBPNN is a neural network architecture which has an input layer, one or more hidden layers, and an output layer. The input layer has as many neurons as the number of features of the input vector $\boldsymbol{x}$. The hidden layer can be a single layer or multiple layers, each with a single neuron or with several neurons. The hidden layer neurons apply a nonlinear transfer function to the sum of weighted input features. The result is multiplied by the weight of the connection between the current layer and the next, and is transferred to the next layer. If the next layer is a hidden layer, the same procedure is repeated. If it is the output layer, then the incoming weighted outputs from hidden neurons are summed and the final output is found.

The more hidden layers, and the more hidden neurons in each hidden layer of a neural network, the more complex the system gets. More complicated systems can fit more difficult problems; however, special attention should be paid not to overfit the training data, which will lower the generalization capability of the network.

The NN with feedforward and backward propagation algorithm can be summarized as follows:

1. The parameters (weights) are initialized randomly.

2. For each $\boldsymbol{x_i}$,

    {Implement forward propagation and get $h_w(\boldsymbol{x_i})$

    Compute cost $J(\boldsymbol{w})$

    Implement back propagation to compute partial derivatives}

3. Implement gradient descent, or Newton's method, or another advanced optimization algorithm to minimize $J(\boldsymbol{w})$ as a function of $\boldsymbol{w}$.

First, the weights are initialized randomly. Then, for each input vector, first the FF algorithm then the BP algorithm is applied. The FF algorithm takes the input and, by applying the transfer functions in the hidden layer neurons, computes the output. The activation function chosen is generally the sigmoid function:

$$\sigma(\boldsymbol{x_j}) = \frac{1}{1 + exp(-\boldsymbol{x_j})} \tag{32}$$

Where

$\boldsymbol{x_j}$ is one feature (dimension) of the input vector x

This value is passed on to all of the units in the next layer by multiplying it with the corresponding weight in that connection. At the end, an estimated output is found for the input. Then the current cost is computed. At this stage, forward propagation is completed and the back propagation algorithm starts.

The back propagation computes the error starting from output node and, by going backwards, computes all of the errors in each node until the input layer. There is no error associated with input layer, so there, nothing is done.

The errors found are used to get the partial derivatives with respect to each weight. Finally, these partial derivatives are used to update the weights. To do this, several methods are possible. The most popular one is the Least Mean Squares (LMS) method also known as Gradient Descent Algorithm. The name Widrow Hoff Algorithm is commonly used too, since it was first introduced by these people. The Gradient Descent Algorithm makes corrections to the weight vector in the direction of the negative of the gradient vector, which eventually leads to the minimum mean square error. Once the weights are updated, the procedure restarts from beginning. This is done until the algorithm converges. Convergence can be met by certain number of iterations, or by checking the change in the error function in the validation set. If the validation set error stops decreasing, and starts increasing for several iterations, the algorithm can be stopped and the weights where the validation set error was lowest can be selected as the model to be used for testing.

By adding feedback loops with tapped delay lines from the predicted output to the input layer, the feedforward neural network architecture can be turned into a recurrent dynamic neural network. Feedback loops provides memory to the NN, which allows for better results in time series problems, where data points are related to each other. This model is explained more in the Methodology chapter, and is used as one of the algorithms for change point detection.

3.3 Hidden Markov Models

Traditional statistical methods, and many machine learning models use restrictive assumptions such as normality, linearity, and independence among predictor variables. The models based on such assumption can be shown as in Fig. 6. For some applications, this assumption will not be satisfactory. Sequential data, for example, is such an area, where treating each point as an individual, independent member of the population will fail to accurately come up with a classification because of the correlation between consecutive pairs of data points. There will be much larger relation between near points in the sequence than between the farther points. However, none of this is captured once the data is treated as i.i.d [41].

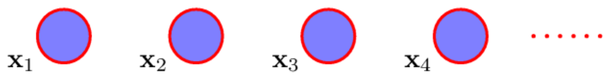


Fig. 6. The model for treating independent data points [41].

The main area where sequence is of concern is time series data. Weather data, the daily values of stock exchange, currency exchange, and acoustic data in speech recognition are a few examples to mention. The nucleotide base pairs in a DNA, and the character and word sequence of a sentence in a particular language are some examples of sequential data where time series is irrelevant [41].

3.3.1 Markov Models

An easy way to capture the effect of correlation between data in a probabilistic model is Markov Chain Model. The beauty of the Markov model lies in its famous characteristic called memory-less property. Basically, the current point is dependent only on the most recent point and is independent of all previous points. This is called the first-order Markov Chain, and it is depicted in Fig. 7. This model allows us to simplify Eq. (33) which is the product rule to express joint probability distribution for a sequence of observations, and to obtain Eq. (34).

$$p(x_1, \dots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \dots, x_{t-1}) \tag{33}$$

$$p(x_1, \dots, x_T) = p(x_1) \prod_{t=2}^{T} p(x_t | x_{t-1}) \tag{34}$$

Also, by using the above property and by applying d-separation, it can be found that the conditional distribution of x_t given all the previous points, is:

$$p(x_t | x_1, \dots, x_{t-1}) = p(x_t | x_{t-1}) \tag{35}$$

The first-order Markov chain can be further relaxed to include several previous points, which yield the higher order Markov chains. For example, in a second order Markov chain, the current point is dependent on the two previous points and is independent of all the rest.

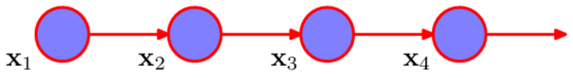


Fig. 7. The model for first order Markov chain [41].

3.3.2 Hidden Markov Models

A richer and more useful model can be reached by introducing latent or hidden variables. For each observation x_t a hidden variable z_t is introduced. The z_t's may or may not be of the same type and dimensionality to the observed variable x_t's. Now, it is assumed that the hidden variables compose the Markov chain, and that the observed variables serve as an output at each point in the sequence, hinting at the underlying hidden variable, which cannot be observed directly. This graphical structure is called the state space model, and is shown in Fig. 8. Speech recognition, natural language modeling, online handwriting recognition, and protein and DNA sequence modeling are some of fields where Hidden Markov models are predominantly used.

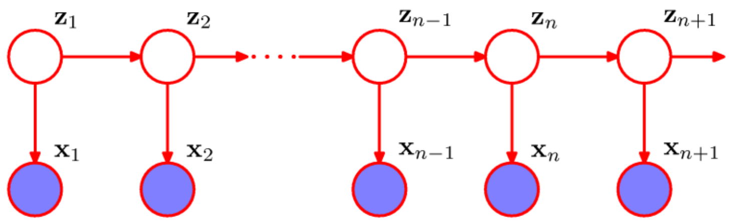


Fig. 8. Hidden Markov model, also known as state space model [41].

The Markov properties also hold for the hidden Markov model. For a hidden Markov model, three parameters need to be found: The initial probability distribution (π), the transition probabilities between latent variables (A), and the probabilities of observing certain outcomes at certain hidden states (B). Thus, the set of parameters governing the model for HMM can be defined by the set given in Eq. (36).

$$\lambda = (A, B, \pi) \tag{36}$$

S being the set of all possible states, and O being the set of all possible observations, we have:

$$S = \{s_1, s_2, \dots, s_N\}, \qquad O = \{o_1, o_2, \dots, o_M\} \tag{37}$$

For a fixed state sequence Z, and the corresponding observation sequence X, we will have:

$$Z = \{z_1, z_2, \dots, z_T\}, \qquad X = \{x_1, x_2, \dots, x_T\} \tag{38}$$

The transition array A is a matrix that stores the probabilities of state j following state i, independent of time:

$$A = [a_{ij}], \qquad a_{z_{t-1},z_t} = P(z_t = s_j | z_{t-1} = s_i) \tag{39}$$

The observation array *B* is a matrix that stores the probability of observation *k* being emitted at state *i*, independent of time:

$$B = [b_{ik}], \qquad b_{z_t,x_t} = P(x_t = o_k | z_t = s_i) \tag{40}$$

Π is the array for initial state probability distribution:

$$\pi = [\pi_i], \qquad \pi_i = P(z_1 = s_i) \tag{41}$$

Two assumptions are present in the HMM. The first property is the Markov property, which says that the current state is dependent only on the previous state:

$$P(z_t | z_1 : z_{t-1}) = P(z_t | z_{t-1}) \tag{42}$$

The second property states that the current observation is dependent only on the current hidden state and is independent of the rest:

$$P(x_t | x_1 : x_{t-1}, z_1 : z_t\ ) = P(x_t | z_t) \tag{43}$$

The joint distribution of hidden Markov model is given in Eq. (44).

$$p(x_1, \dots, x_T, z_1, \dots, z_T) = \ p(z_1) \prod_{t=2}^{T} p(z_t | z_{t-1}) \prod_{t=1}^{T} p(x_t | z_t) \tag{44}$$

There are three main fields that need to be addressed in a hidden Markov model. These will be explained briefly.

3.3.2.1 Evaluation

Given the model λ for HMM, and the observation sequence, the probability of the observations given the model *P(X|λ)* is calculated. This can be viewed as the "evaluation" of the model in how well it predicts the given sequence of observations.

The probability of observations for a specific state sequence *Z* is:

$$P(X|Z,\lambda) = \prod_{t=1}^{T} P(x_t|z_t,\lambda) = b_{z_1,x_1} b_{z_2,x_2} \dots b_{z_T,x_T} \tag{45}$$

The probability of state sequence is:

$$P(Z|\lambda) = \prod_{t=1}^{T} P(x_t|z_t,\lambda) = \pi_{z_1} a_{z_1,z_2} a_{z_2,z_3} \dots a_{z_{T-1},z_T} \tag{46}$$

Thus, the probability of the observations given the model parameters is:

$$P(X|\lambda) = \sum_{Z} P(X|Z,\lambda)P(Z|\lambda) = \sum_{Z} \pi_{z_1} b_{z_1,x_1} a_{z_1,z_2} b_{z_2,x_2} \dots a_{z_{T-1},z_T} b_{z_T,x_T} \tag{47}$$

An easier approach to find the probability of the observation sequence is defined by the forward algorithm. The forward probability variable *α* which is the probability of the partial observation sequence $x_1, x_2, \dots, x_t$ and state s_i at time *t*, defined as:

$$\alpha_t(i) = P(x_1 x_2 \dots x_t, z_t = s_i|\lambda) \tag{48}$$

Thus if the trellis diagram is filled from begin to end with *α* values obtained in each step, the sum of α’s in the final column will be equal to the probability of the observation sequence.

The forward algorithm is as follows:

1. Initialization

$$\alpha_1(i) = \pi_i b_{z_i,x_1}, \qquad 1 \le i \le N \tag{49}$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_{z_j,x_{t+1}}, \qquad 1 \le t \le T, 1 \le j \le N \tag{50}$$

3. Termination

$$P(X|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{51}$$

The forward algorithm reduces the complexity of computing the probability of an observation sequence given the model from $2TN^T$ to N^2T, which is huge.

3.3.2.2 Decoding

Decoding is the process to get the hidden state sequence that was most likely responsible for generating the observed sequence. The Viterbi Algorithm is used to get a single best state sequence. The Viterbi Algorithm is very similar to the forward algorithm, but instead of summing at each step at each state, the transition probabilities are maximized. The variable δ is the probability of the most probable state path for the partial observation sequence.

$$\delta_t(i) = \max_{Z} P(z_1 z_2 \dots z_t = s_i, x_1 x_2 \dots x_t | \lambda) \tag{52}$$

The Viterbi Algorithm is as follows:

1. Initialization

$$\delta_1(i) = \pi_i b_{z_i, x_1}, \qquad 1 \le i \le N, \psi_1(i) = 0 \tag{53}$$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] b_{z_j, x_t}, \qquad 2 \le t \le T, 1 \le j \le N \tag{54}$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right], \qquad 2 \le t \le T, 1 \le j \le N \tag{55}$$

3. Termination

$$P^* = \max_{1 \le i \le N} [\delta_T(i)] \tag{56}$$

$$z_T^* = \arg\max_{1 \le i \le N} [\delta_t(i)] \tag{57}$$

4. Optimal state sequence backtracking

$$z_t^* = \psi_{t+1}, (z_{t+1}^*), \qquad t = T-1, T-2, \dots, 1 \tag{58}$$

The variable ψ stores the state that maximizes the current partial probability and acts as a backpointer in Step 4. After finalizing the algorithm by finding the maximum probability, the state sequence is backtracked and the whole state sequence is revealed.

3.3.2.3 Learning

This is the process for learning the model parameters: initial state probabilities array π, state transition array A, and observation probability array B. Two basic approaches can be applied to compute the parameters: the supervised and the unsupervised learning. If we know the state sequence and the observation sequence, the state sequence can be used as labels to the observation sequence, making it a supervised learning. The parameters can be found by maximum likelihood estimation. If the state sequence is not known, or if it is not wanted to be used, then all that is left is the observation sequence; thus, the unsupervised learning approach should be taken. In this case, the Baum-Welch algorithm is a well-known algorithm to estimate the model parameters.

CHAPTER 4

DATA COLLECTION

The field data are collected with two devices: a smartphone and an on-board diagnostics (OBD) device. The on-board diagnostics second-generation (OBD II) device is connected to the vehicle's CAN bus. The OBD device used is capable of transmitting data via Bluetooth. The smartphone and the OBD are paired via a Bluetooth connection. An Android application is developed which records the data from the smartphone sensors, and also logs GPS readings and the data transmitted from the OBD, such as speed. The overall schematic of the data collection and processing can be seen in Fig. 9.

Each sensor has its own data sampling rate, and the app is set to log data from each sensor at the highest rate allowed. From the field data collected, it is found that the sampling rate for accelerometer sensor can be as low as 1 per second and as high as 238 per second, with the majority at 15 per second. In general, the more the forces exerted on the phone, the higher the sensor activity is. The GPS and OBD data collection rate is at 1 per second. These separate datasets from acceleration, GPS, and OBD are first interpolated with a common start and end time and then combined together, which yields the raw data. The OBD or GPS speed data are used to create ground truth data of vehicle standstill and motion states which will be used in model training and testing.

The data collection process involved normal driving on streets, arterials with signalized intersections, and the highways in the Hampton Roads area in Virginia. The vehicle and device combinations that were used to collect data are: Toyota Prius 2007 – Moto XT1063; Mazda 3 2007 – Samsung Galaxy S3; Toyota Camry 2012 – Lg G4. The summary of the trips taken by the drivers are given in Table 1. Some of the trips of each driver are plotted on the map in Fig. 10, and their average speeds are shown in Fig. 11.

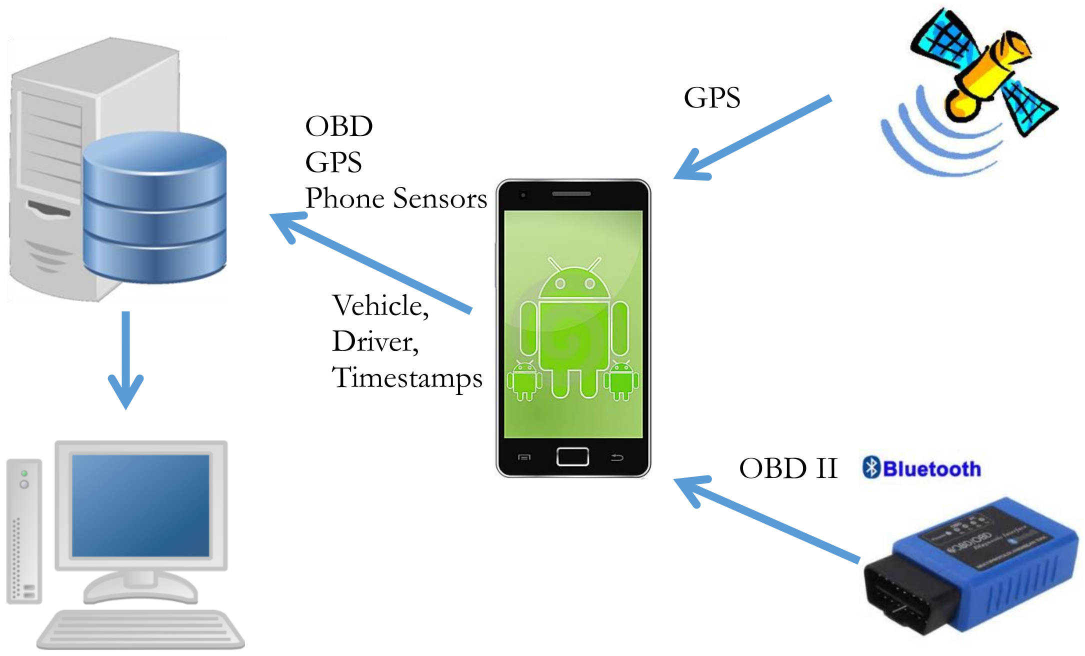


Fig. 9. The schema for data collection and processing.

Table 1. Aggregate summary of datasets used.

	Dataset	# of Trips	Total Length(min)	Standstill Duration(min)	% Stop Duration	# of Stops	MaxSpeed (mph)	AvgSpeed (mph)
Prius	Train	10	181	25	14	64	69	33
	Val	10	241	56	23	125	68	24
	Test	20	496	91	18	251	68	25
Camry	Train	10	204	34	17	74	74	30
	Val	10	182	18	10	36	82	40
	Test	20	488	77	16	137	78	33
Mazda	Train	10	289	69	24	135	70	24
	Val	10	207	50	24	74	72	31
	Test	20	462	140	30	160	80	24

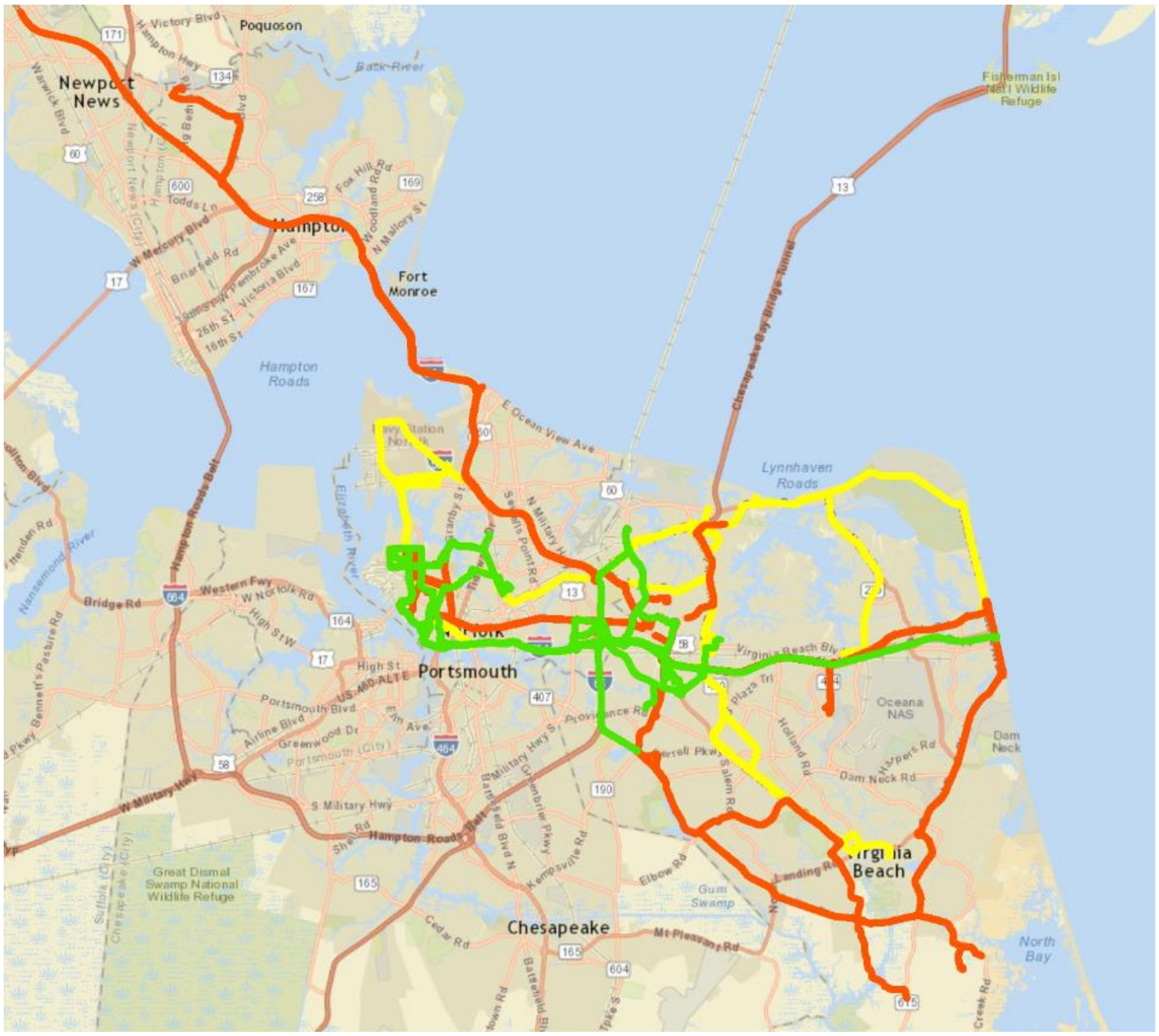


Fig. 10. Map of Hampton Roads and trips by three different drivers.

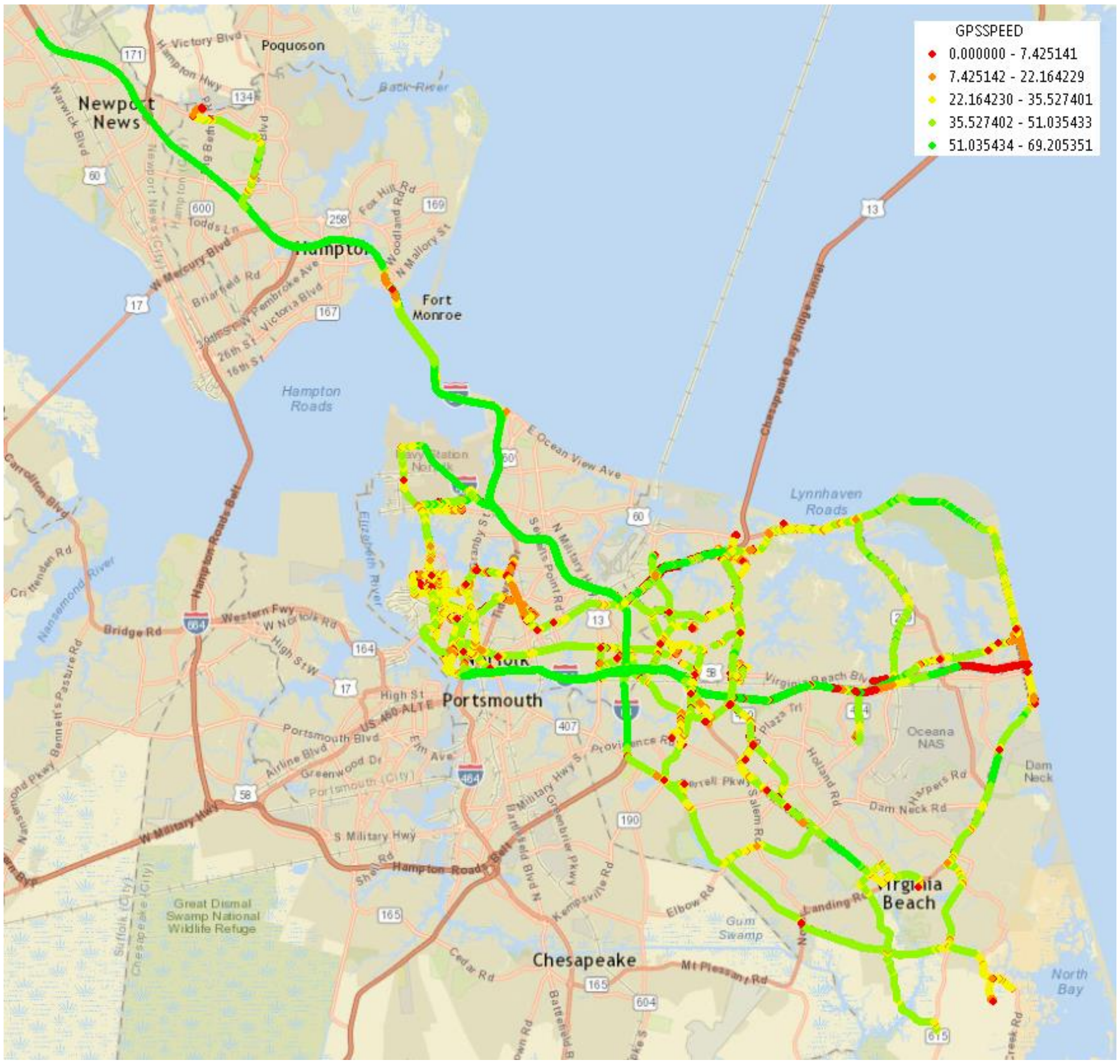


Fig. 11. Map of Hampton Roads and the average speeds of the three drivers.

The application obtains data from the phone’s three-axis accelerometer which measures the acceleration in three directions, its three-axis gyroscope which measures the angular velocity, and its 3-axis magnetometer which measures the angle by which the device is rotated, relative to the Earth’s magnetic north pole. Thus, we have, in total, nine independent measurements from three low-energy sensors. These three axes, X, Y and Z, of a phone are shown in Fig. 12.

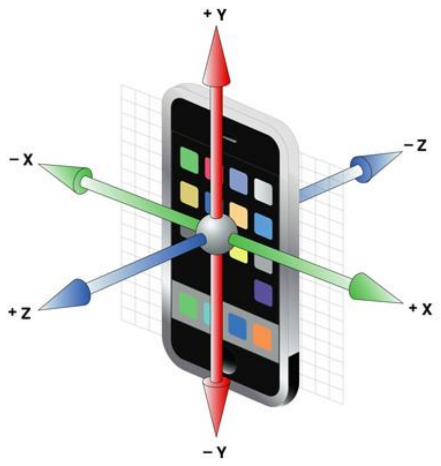


Fig. 12. The X, Y and Z axes of a phone.

The triaxial accelerometer data, together with the vehicle speed obtained from the OBD or GPS, are used for model training and testing, where the purpose is to develop an algorithm which can detect the state of the vehicle (i.e., whether it is in motion or stationary/stopping) by only using the low-energy sensor data collected. Different combinations of sensor data were experimented with, such as acceleration data alone or acceleration together with gyroscope data. It was found that, in some phones, gyroscope data exhibits a significant drift which is inconsistent with the state of the vehicle. Thus, only acceleration data, which is found to behave reliably, and distinctly according to the state of the vehicle, are used for detecting vehicle standstill or motion. The distinctive behavior of acceleration data can be seen when checked together with the speed of the vehicle as shown in Fig. 13.

In this figure, the dashed blue line at the top is the OBD speed, yellow line is the GPS speed, followed by the magnitude of the accelerometer data or the total acceleration as calculated in Eq. (1). The next three subplots present the accelerometer data in X, Y, and Z axes which are depicted in red, green, and blue solid lines, respectively. In this figure, the X, and Z are each under the effect of gravity since the phone was not positioned correctly in a straight line in the vehicle. The Y axis represents the motion direction (with the phone situated next to the driver with its longitudinal direction towards the front of the vehicle). The changes in the speed of the vehicle are reflected mostly in the changes of the acceleration value in the Y axis. It can be observed that there exists a pattern between the accelerations as measured by the three directions X, Y, and Z and the speed. It is this behavior of the sensors that will be exploited in order to determine the state of the vehicle.

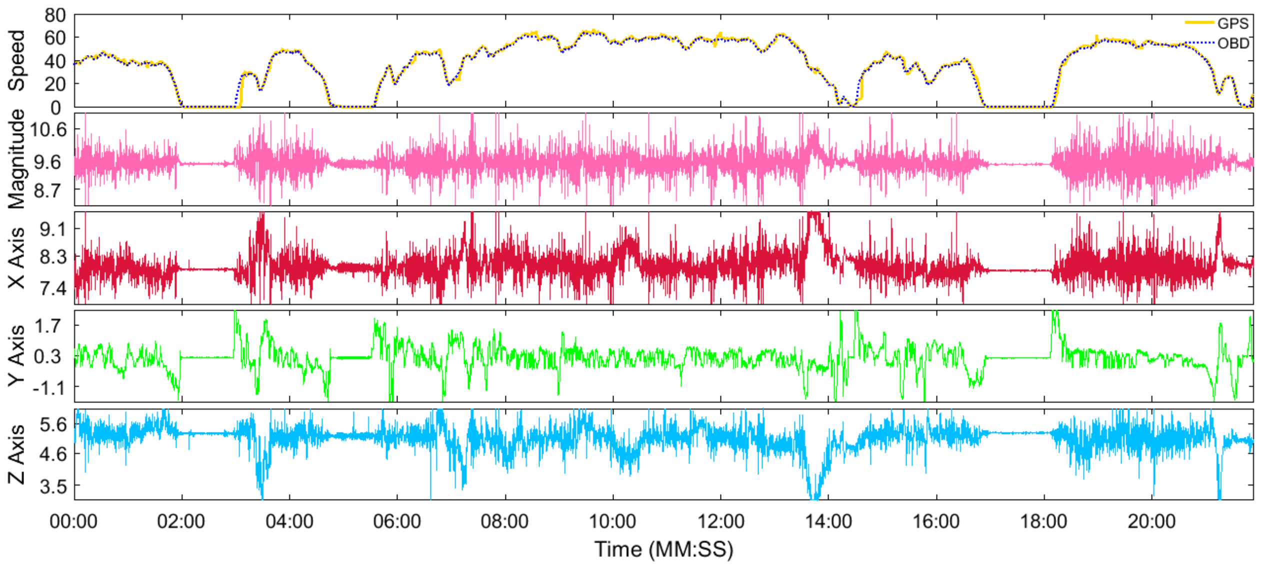


Fig. 13. Three-axis accelerometer data collected by the smartphone for a trip.

4.1 Data Analysis and Feature Creation

The raw accelerometer data consist of points of instantaneous measurements, i.e. the data points are measurements taken at specific instants, for example every tenth of a second. The phone is placed in the vehicle in a stationary position, but might be at an arbitrary orientation. Since it is almost impossible for a phone to be perfectly set such that one direction will be affected by gravity, the other direction by motion, and a third direction by lateral movements alone, phone orientation correction methods need to be applied to mitigate the effect of gravity on the axes. In order to eliminate this phase, and also to make the vehicle motion detection simpler and applicable to any orientation, total acceleration values are used. The total acceleration (acc_{Tot}), or magnitude, is defined as the square root of the sum of squares of accelerometer values measured in each direction (x, y, z), as shown in Eq. (59).

$$acc_{Tot} = \sqrt{\left(acc_x{}^2 + acc_y{}^2 + acc_z{}^2\right)} \tag{59}$$

Having calculated the magnitude, a sliding window is used to extract features. The sliding window overlaps all the points in the previous window except one, which is the beginning point of the previous window. A new feature set is created in which the calculated features are assigned to the middle point of the window. Thus, this newly created feature set will have the same number of points as the original set. The aim of feature creation is to improve the classification performance of the algorithms. As can be seen from the box plot in Fig. 14, the distinguishing capability of the raw magnitude data is very low.

Based on Fig. 13, it is expected that when the vehicle is stationary, the variation of points within a certain duration (window) will be small. On the other hand, when the vehicle is in motion, the fluctuations will increase due to acceleration/deceleration and pavement imperfections. Any feature that is based on the spread of the data should do a better job in classifying than any raw data would do. On the other hand, it can be seen that, although the motion segments show large fluctuations, the mean points of the motion segments do not differ much from those of the standstill segments. Thus, center-based features such as mean are not expected to do well. To validate this intuition, center-based, extreme-value-based, and variation-based features are calculated for a certain window size, and are shown in box plots in Fig. 15, Fig. 16, and Fig. 17. As expected, the center-based features overlap too much, and so the classification will not be a good one. The percentiles, which measure the spread of the data will also not do well in classification, because of the large overlap. The extreme-value statistics show more promise than the center-based and the percentiles since they have less overlap between the two states. Here, the maximum value has more predictive power than the minimum. The variance-based features are shown in Fig. 17. As expected, these features have very high predictive value, since they can differentiate between the two states very well.

The overall process of feature extraction is summarized in Fig. 18. The same process is repeated for the creation of train, validation, and test sets.

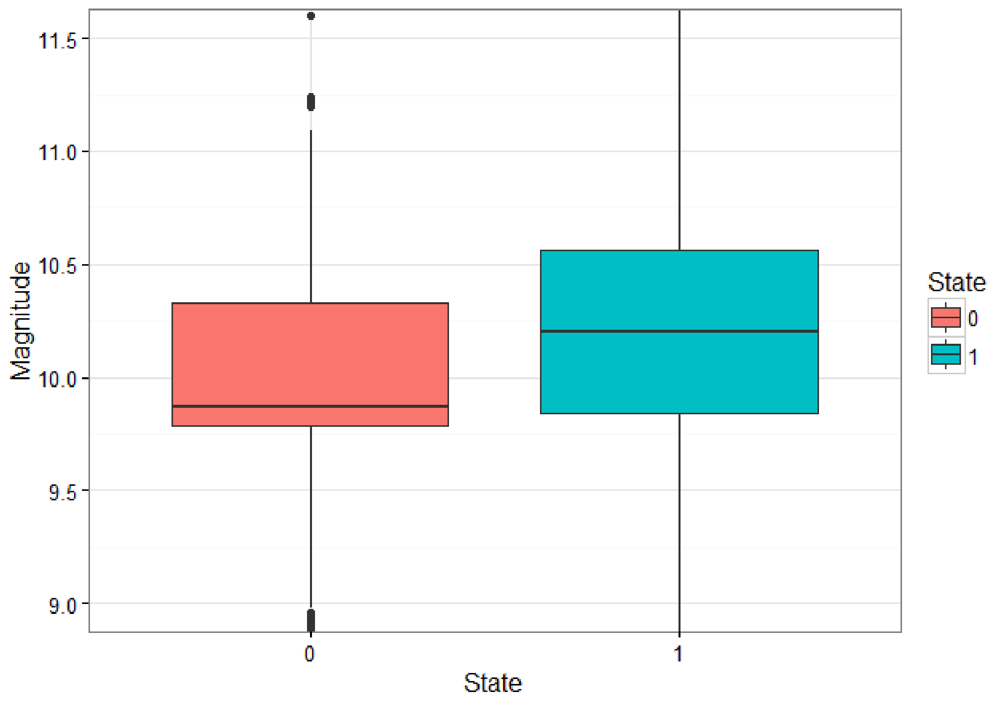


Fig. 14. The boxplot of the magnitude of the raw data.

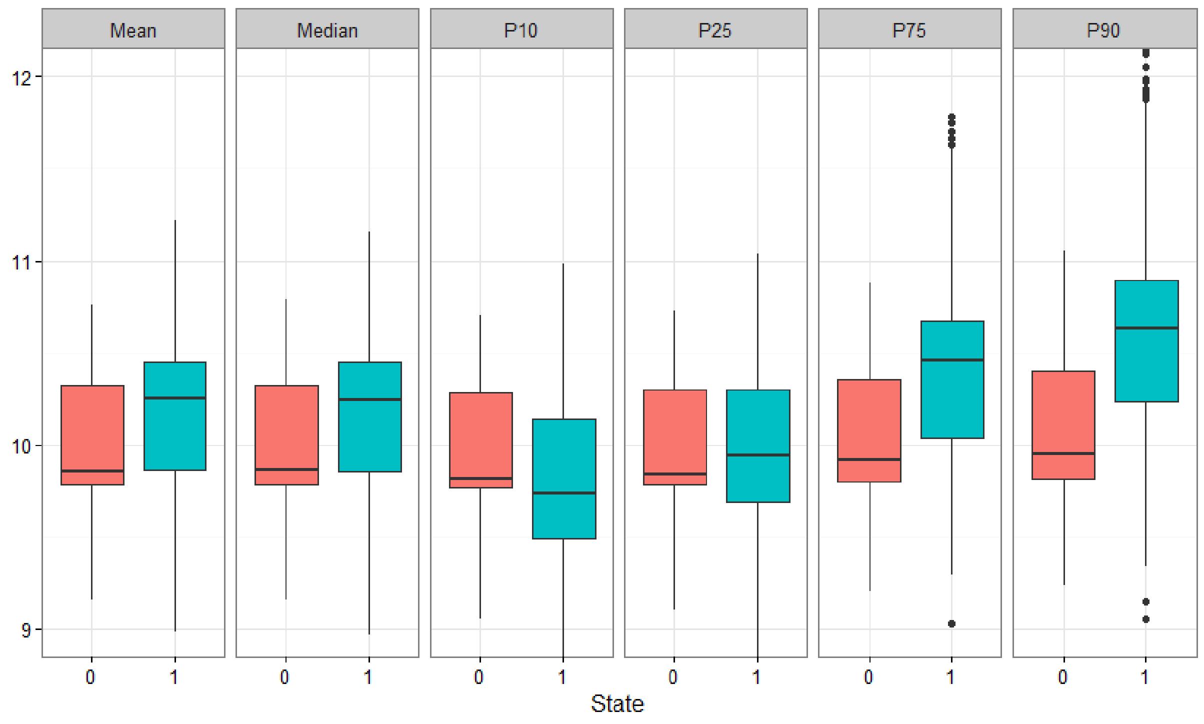


Fig. 15. Center-based features and percentiles extracted from the data with a sliding window.

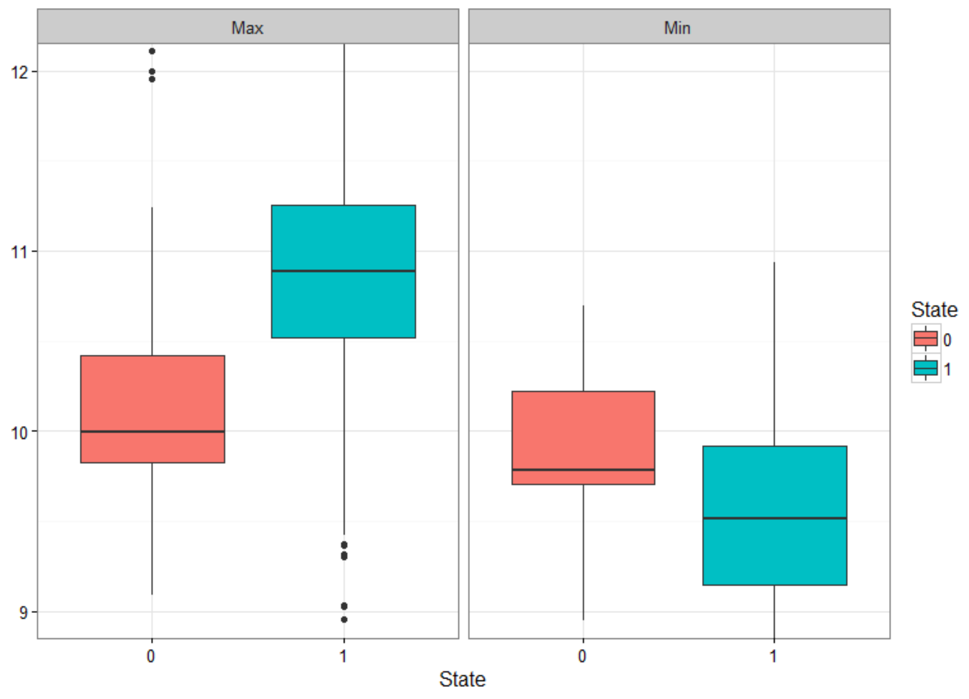


Fig. 16. Extreme-value features extracted from the data with a sliding window.

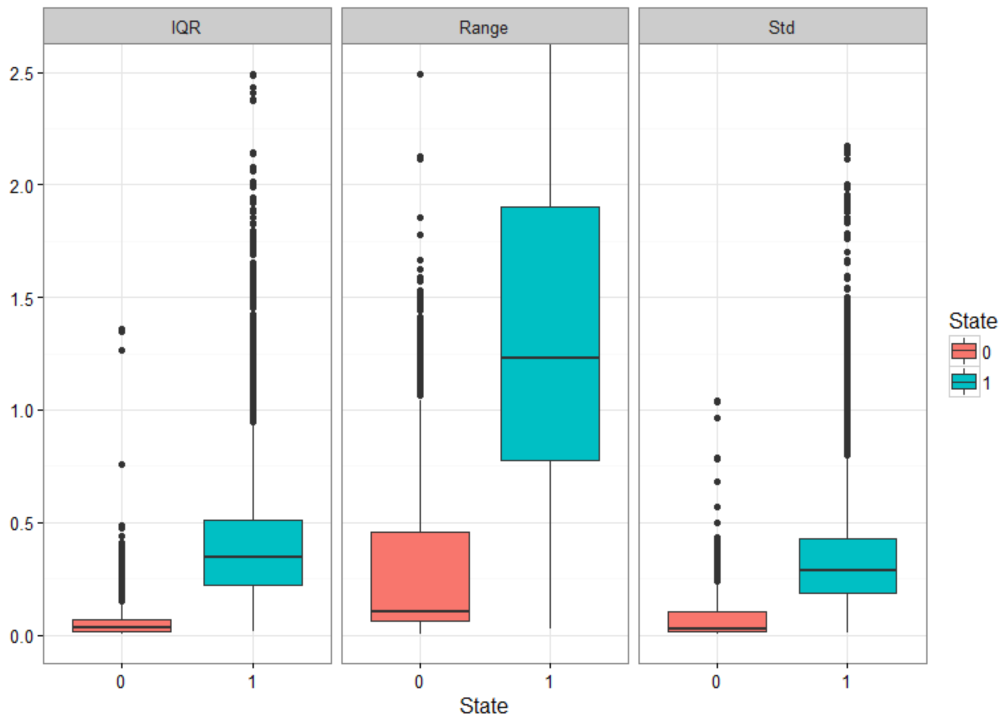


Fig. 17. Variation-based features extracted from the data with a sliding window.

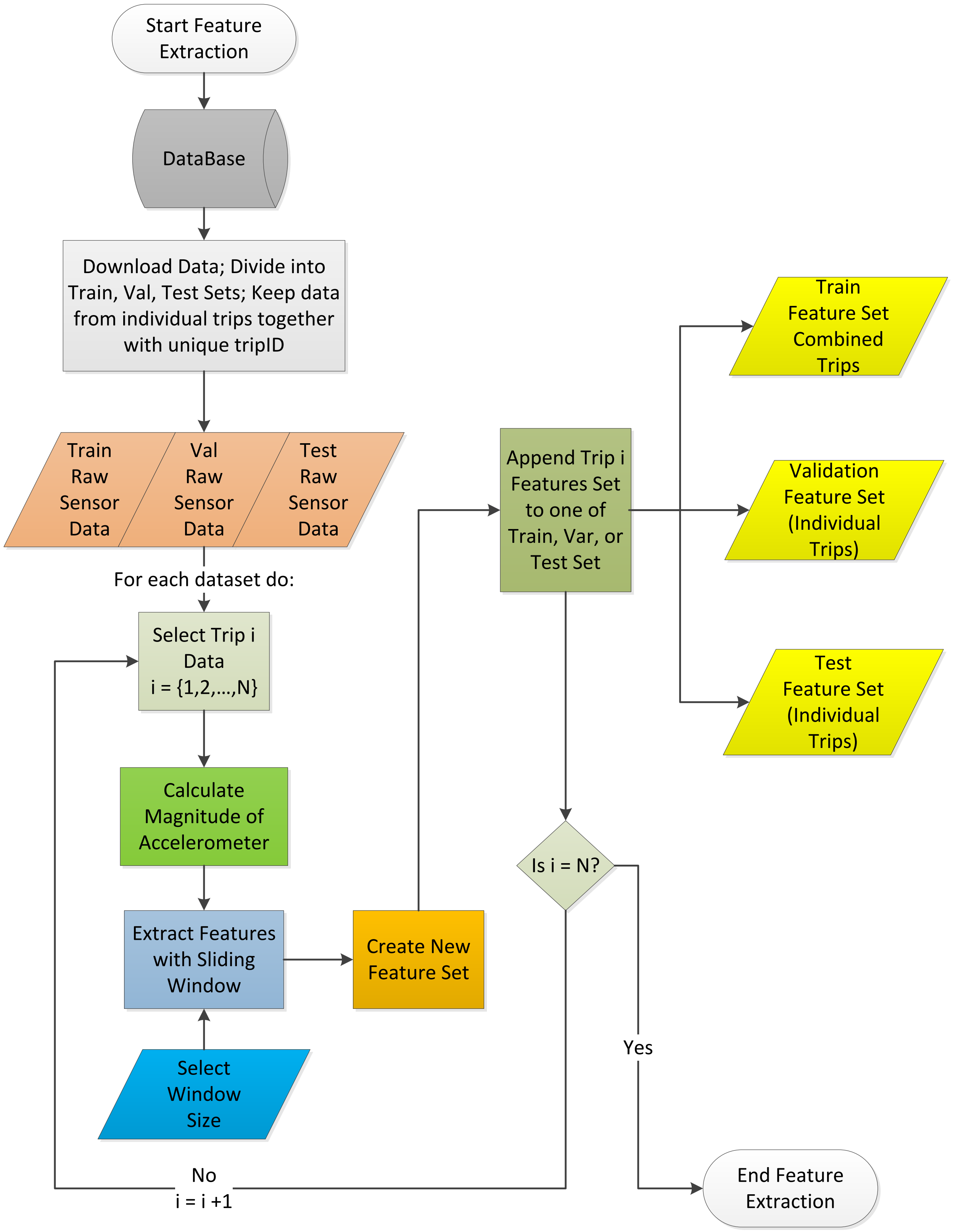

Start Feature Extraction
DataBase
Download Data; Divide into Train, Val, Test Sets; Keep data from individual trips together with unique tripID
Train Raw Sensor Data
Val Raw Sensor Data
Test Raw Sensor Data
For each dataset do:
Select Trip i Data i = {1,2,…,N}
Calculate Magnitude of Accelerometer
Extract Features with Sliding Window
Select Window Size
Create New Feature Set
Append Trip i Features Set to one of Train, Var, or Test Set
Train Feature Set Combined Trips
Validation Feature Set (Individual Trips)
Test Feature Set (Individual Trips)
Is i = N?
Yes
No
i = i +1
End Feature Extraction

Fig. 18. Extracting features and preparing data for analysis.

4.2 Feature Selection

The usefulness of a feature affects the performance of machine learning algorithms significantly. Having many features does not necessarily improve the learning of a model. On the contrary, some features might even degrade the performance of the model. In order to select best features some methods are developed in the literature. Feature selection is important, both to speed up the process and to improve the accuracy of the classification. As shown before, some features do have higher predictive power compared to the rest. Visualizing the features is one method in assessing feature quality. There are also other methods for feature evaluation and selection. Several of the feature evaluation and selection methods have been applied and the results are presented next.

4.2.1 Ranking Features

One of the approaches for feature selection is ranking features. The rankfeatures function in MATLAB is used to score each feature. It is based on class separability criteria and uses independent evaluation criteria for binary classification. Methods such as t-test, entropy, Bhattacharyya, ROC, and Wilcoxon are some criteria used to assess the significance of each feature for separating two labeled groups. The t-test method uses the absolute value two sample t-test with pooled variance. The entropy method calculates the relative entropy, also referred to as Kullback-Leibler distance or divergence. The Bhattacharyya method seeks to find the minimum attainable classification error or the Chernoff bound. The ROC method will give the area between the empirical receiver operating characteristic (ROC) curve and the random classifier slope. A larger area means the feature is performing better in terms of separation of classes. The Mann Whitney Wilcoxon method is a statistical test of the null hypothesis that two samples come from the same population against the alternative hypothesis that two samples come from different populations. The Mann Whitney Wilcoxon method will yield the absolute value of the standardized u-statistic of a two-sample unpaired Wilcoxon test, also known as Mann-Whitney. 'ttest', 'entropy', and 'bhattacharyya' assume normal distributed classes while 'roc' and 'wilcoxon' are nonparametric tests. Each feature is evaluated on its own, without any combination with other features. Thus, all tests are feature independent.

Each feature is normalized independently across all observations. This process, also known as cross-normalization, ensures comparability among different features. The standard score normalization is used as shown in (60).

$$x_{ij,Normalized} = \frac{x_{ij} - \bar{x}_j}{s_j}, for\ i = \{1,2,3, \dots , |x|\} \tag{60}$$

Where,

i	Element *i* in vector feature j,
x_j	The feature vector j,
$\bar{x}_j$	The sample mean of the vector feature j,
s_j	The sample standard deviation of the vector feature j

The rankfeatures function present in the Bioinformatics Toolbox of MATLAB allows the user to take into account the correlation. This is done by multiplying the test statistic value of the next best candidate feature by a coefficient *(1-α)*ρ*. *ρ* is the average of the absolute values of the cross correlation coefficients between the candidate feature and all previously selected features. *α* is a weighting factor which is between 0 and 1 that determines the importance of the correlation effect. When *α* is zero, the cross correlation between the features is not taken into account. When *α* is set a large value, the statistic will get smaller. This means that the candidate feature which is strongly correlated with previously selected features will be less likely to be among the more highly ranked features. This approach will ensure less redundancy in the features selected. As a matter of fact, using strongly correlated features might lead to deterioration in classification performance in some machine learning techniques.

The results of ranking the features based on different statistical tests and alpha values are presented in Fig. 19 through Fig. 21. Here, each vehicle and phone is analyzed separately. Range, standard deviation, and interquartile range are the best features for the Toyota Camry – LG G4 combination. For the Mazda3 – Samsung S3 range, standard deviation, and interquartile range are again the most important. However, some features that weren't as important with Toyota Camry show more significance. The absolute difference of consecutive ranges (Diff), and the norm2 features are such examples. The Toyota Prius – Motorola XT1063 presents quite a different case where the minimum, 10th quartile, and interquartile range were the most significant, followed by range, standard deviation, and norm2. At different levels of alpha, the outcome of each test differs a bit. When making these comparisons, the case where alpha is equal to 1 is used, since at this alpha level, the penalty for having correlated variables is the largest.

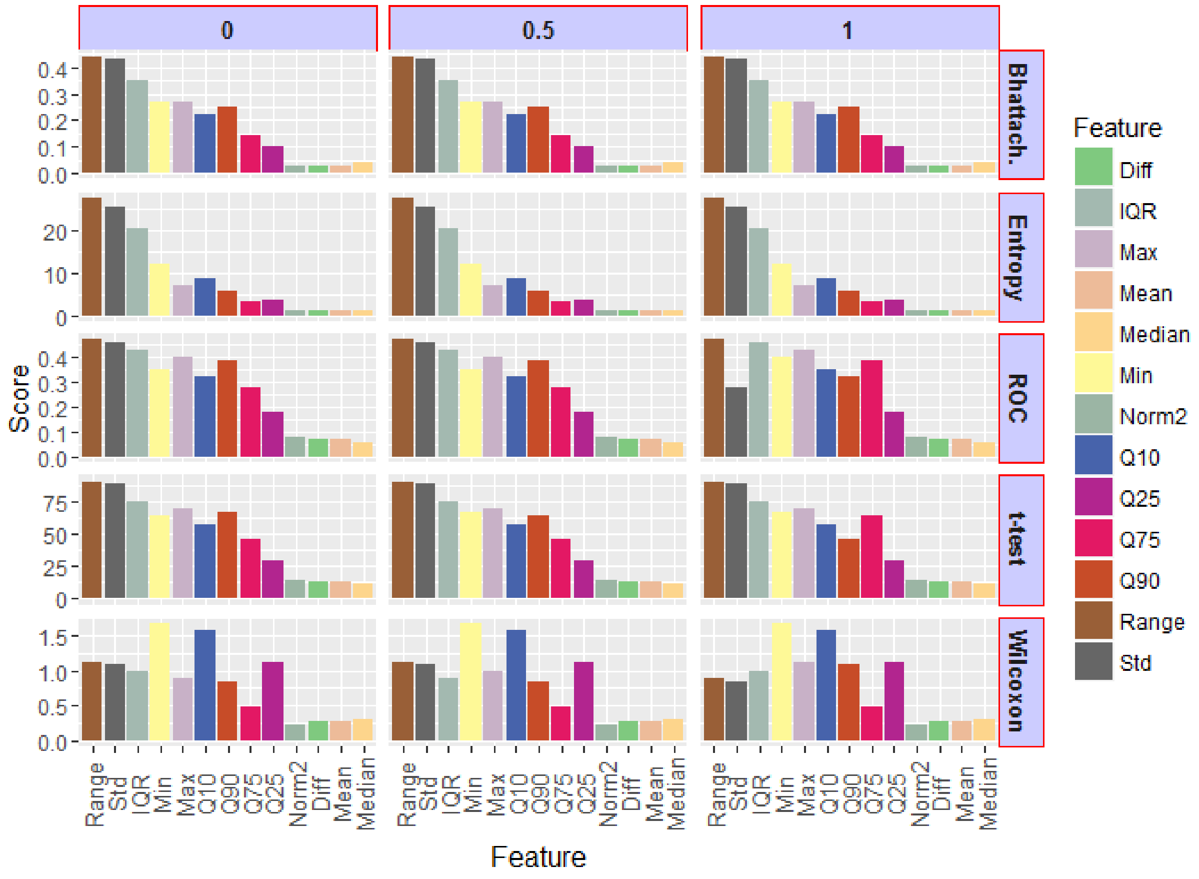


Fig. 19. Ranked feature scores w.r.t. alpha and the test for Toyota Camry, LG G4.

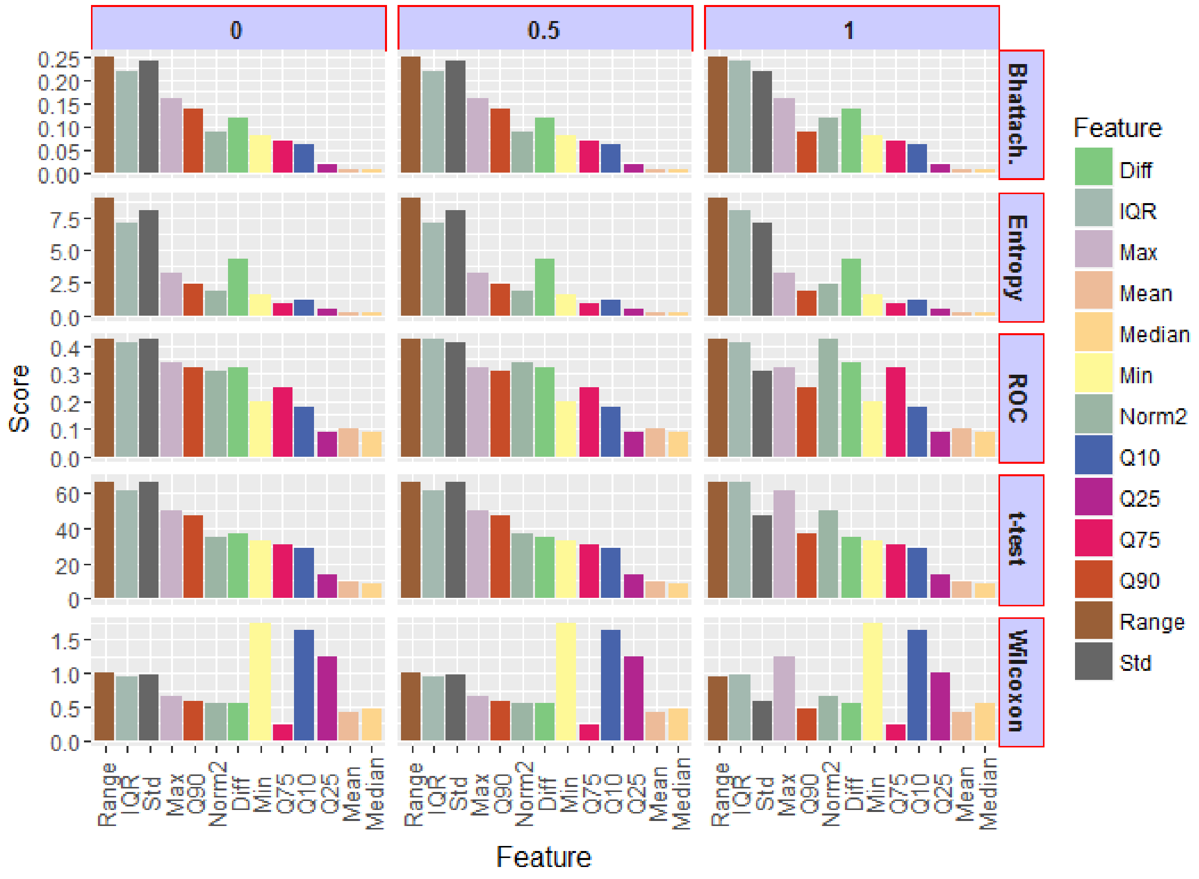


Fig. 20. Ranked feature scores w.r.t. alpha and the test for Mazda 3, Samsung Galaxy S3.

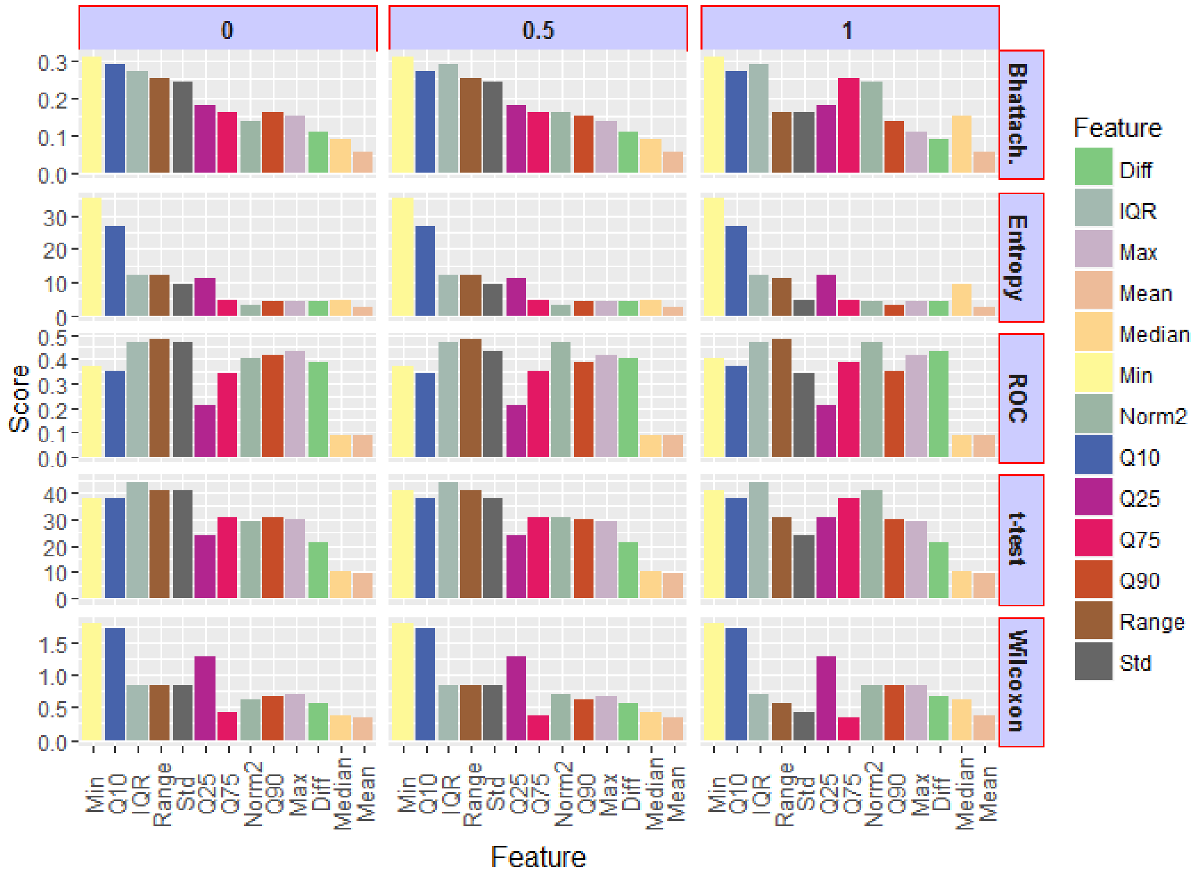


Fig. 21. Ranked feature scores w.r.t. alpha and the test for Toyota Prius, Moto X1063.

4.2.2 ReliefF Algorithm

Relief is a feature selection algorithm proposed by Kira and Rendell [42]. The goal is to find the values of the weight vector, where each element represents the relative importance of each feature. The weight vector at each step is updated based on the one data point from the positive and negative classes which are closest to the currently selected point. The weight vector entries increase if the difference between current and closest point of the same class (near-hit) is less than the difference between the current and the closest point of the different class (near-miss), as calculated in Eq. (61). The algorithm can be run for each data point or on randomly selected points.

$$W_i = W_i - \left(diff(x_i, nearHit_i)\right)^2 + \left(diff(x_i, nearMiss_i)\right)^2 \tag{61}$$

Where,

W_i	i^{th} attribute value of the weight vector
x_i	i^{th} attribute value of the currently selected data point
$diff$	Normalized difference between current point and the closest neighbor
$nearHit_i$	Closest neighbor that is of the same class
$nearMiss_i$	Closest neighbor that is of the other class

Eq. (61) essentially allows the weight vector to decrease if the feature of the selected point differs more with its own class than the other class, and increase otherwise. Thus, the features that are within close proximity to other features within the same class cause the weight vector to become larger. At the end, the features with largest relative weights will be the most important ones for classification.

The ReliefF algorithm is an improvement to the relief algorithm. ReliefF uses K nearest neighbors instead of a single nearest hit and miss point. This, in theory, should lead ReliefF to be less noisy and more robust. Another change is the use of L1 Norm in ReliefF instead of the L2 norm. The attribute weights range from -1 to 1, with important attributes being assigned large positive values.

Since ReliefF essentially performs the K Nearest Neighbor algorithm at each selected data point, the value of *K* can drastically affect the performance of the algorithm. If *K* is set too small, noise will have a great impact, and the result will be unreliable. If *K* is set too large, it will be too slow to respond to changes in the decision surface and will fail to find important attributes. Selecting the right *K* value can be achieved by an exhaustive search.

Feature selection results based on the ReliefF method is presented in Fig. 23 through Fig. 22. As before, each vehicle and phone are analyzed separately. Range, standard deviation, and interquartile range were the best features for the Toyota Camry – LG G4. For the Mazda3 – Samsung S3, norm2, range, standard deviation, and interquartile range were again the most important. The main difference is that the feature norm2 was the most significant for the Mazda3. For the Toyota Prius – Motorola XT1063, range and standard deviation were the most significant, followed by interquartile range. The feature minimum was not as strong as was the case in ranking features.

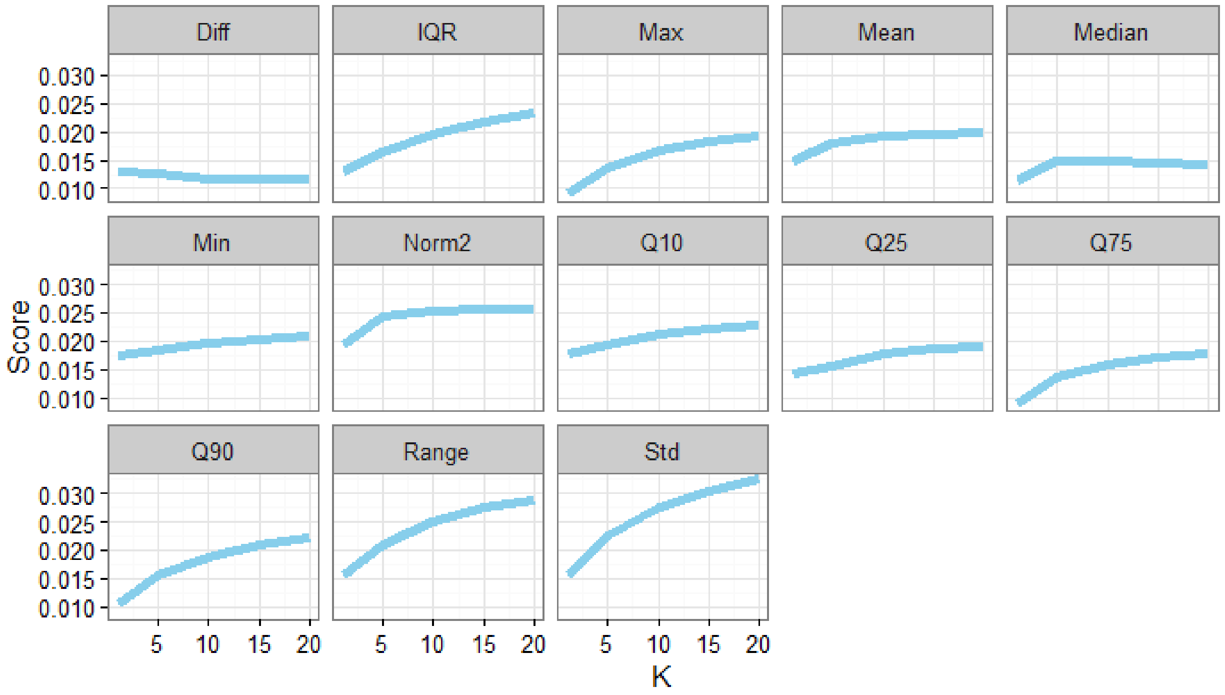


Fig. 22. ReliefF feature weights for each K nearest neighbor for Toyota Prius.

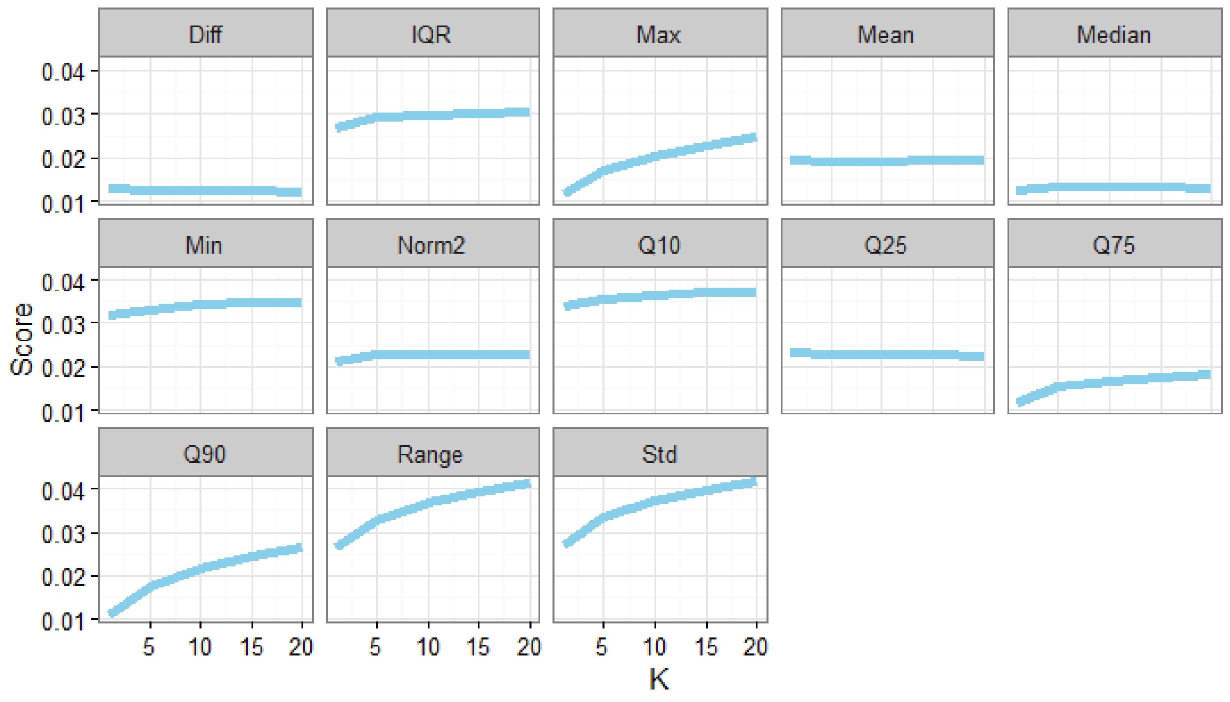


Fig. 23. ReliefF feature weights for each K nearest neighbor for Toyota Camry.

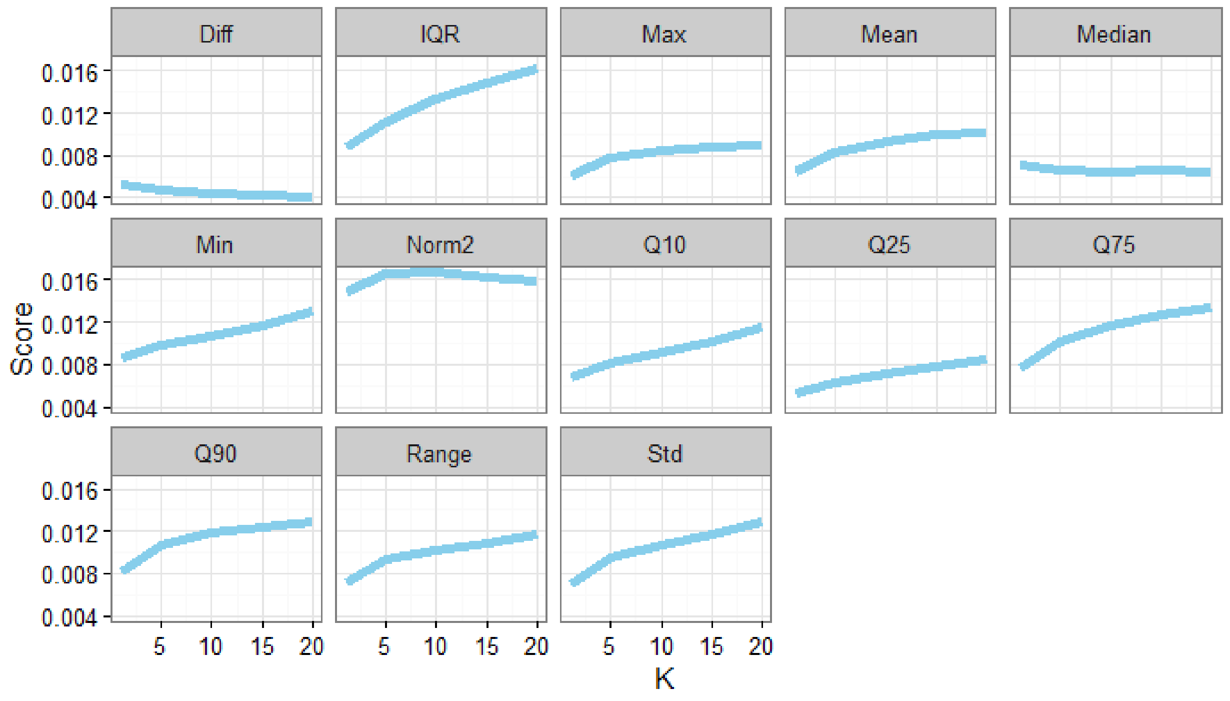


Fig. 24. ReliefF feature weights for each K nearest neighbor for Mazda3.

The feature scoring methods show that some features, such as range and standard deviation, have strong separation capability, while some features, such as mean and median, would not perform well in classiying the states of the vehicle. This outcome is in line with the intuition pointed out before: the features that are based on variation should do better, and the ones that are based on centre of the data should not, as it is evident from Fig. 13.

CHAPTER 5

METHODOLOGY

Machine learning techniques are utilized to detect the stopping and moving events. Supervised learning techniques in which labeled data are used for training and testing phases have proven to be successful in many applications. Techniques such as static neural networks, Bayes classifiers, logistic regression, etc., perform well when the data are generated by independent processes. However, the sensor data collected here are sequential and correlated. The data points are not individual points of independent measurements, but are, rather, a sequence of points collected at a certain frequency. Since the data are sequential, the data points are correlated with each other, violating the independence assumption. Thus, using the classification techniques may prove not to be sufficient, since they cannot exploit the sequential patterns in the data, such as correlations between observations that are close to each other in the sequence.

Support vector machines, Hidden Markov models, and dynamic neural networks are chosen to be candidates to find the best technique that can most accurately and precisely find the vehicle motion and stop points. Although SVM does not take into account time information, because of its special characteristics such as the kernel trick, it was found to perform better than the other static methods, hence is included in the dissertation.

Before going into the application details and the results of the various algorithms, a novel error quantification method will be presented. One of the significant contributions of this research to the state of the art will be the quantification of performance when only a few points of interest are of concern within a time series data.

5.1 Assessing Model Accuracy

The most common method to measure the error of a classification algorithm is done by comparing the actual class with the estimated class. If the two agree, then the error is zero; otherwise, it is one. The sum of non-agreeing cases is divided by the total number of points to obtain an average error. Taking the average allows one to assess different models on the same dataset, or to compare the performance of the same model on different datasets.

$$error = \frac{1}{N}\sum_{i=1}^{N}\delta_i \tag{62}$$

Where;

δ_i is equal to 0 if $y_i = t_i$; $\delta_i = 1$ if $y_i \neq t_i$, for $i = \{1,2,\ldots,N\}$

y_i The estimated class of ith input

t_i The observed (true) class of ith input

In a time series estimation, several error quantification methods are all based on one basic measure, which is the difference between the actual value and the estimated value at a specific instant signified as e_n. Some common time series error measures are listed in Eq. (63) through Eq. (66).

Mean Absolute Deviation (MAD):

$$MAD = \frac{1}{N}\sum_{n=1}^{N}|e_n| \tag{63}$$

Mean Absolute Percent Error (MAPE):

$$MAPE = \frac{1}{N}\sum_{n=1}^{N}{}^{|e_n|}/_{t_n} \tag{64}$$

Mean Squared Error (MSE):

$$MSE = \frac{1}{N}\sum_{n=1}^{N} e_n^2 \tag{65}$$

Root Mean Squared Error (MSE):

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N} e_n^2} \tag{66}$$

In this research, the goal is to identify the stop and start instances during a trip. In order to find these points, accelerometer data is used. Identifying these points is a type of classification, while the data used is a time series data. The aim is not to predict or forecast the accelerometer values, but rather to classify each instance as standstill or motion, and then to find the change points. The measures given above are suitable when the predictions are continuous valued time series. Thus, time series error metrics are not adequate to assess the performance of the classification of vehicle states.

The goal in classification of independent data is to get as many data points correct. However, in estimating the state of the vehicle, although correctly classifying each instance is important, it is more important to be able to detect points of change with high accuracy. High accuracy of a model depends mainly on three criteria:

- Whether it is able to find each stop and start point (events);
- How precise it is in estimating the events, i.e. how close the estimation is to the observed points;
- Whether it is introducing any false stop or start points.

In the case of classification error, since a typical trip of 30 minutes or more will have thousands of points, depending on the sampling frequency, misclassifying a few of them which corresponds to a few seconds, will not increase the error significantly. This might lead the model to predict many false standstill or motion sections without any significant increase in error, due to their short durations. Also, the true stop and start points might be missed or might be estimated with an unacceptable distance. None of these anomalies can be detected with regular classification error, since classification error makes comparison on a point-by-point basis. To address these shortcomings of the regular classification accuracy metric, a novel performance quantification method is introduced, which will be used to evaluate the performance of the state predictions.

The essence of the performance metric will be explained by examples. Fictitious trips of 100 seconds will be used to depict the observed and predicted states. The intervals correspond to a duration of 1 second. Each point corresponds to a state of the vehicle where 1 denotes motion, and 0 denotes standstill. The change points where a transition happens from state 1 to 0 corresponds to a stop point, while a change from 0 to 1 corresponds to a start point. A stop and start point pair will be referred to as an "event." The top (blue) line is the observed trip, whereas the next two lines are the estimations 1 (red color) and 2 (gold color), which were predicted by models 1 and 2, respectively. For brevity, the models and their corresponding predictions will be referred to as model1 and model2 and prediction1 and prediction2, respectively.

As a first example, consider the case in Fig. 25. Prediction1 detects the first stop two seconds earlier and the second start point two seconds later. So, in total, there is a difference of four seconds between actual and predicted states. Both events are detected with close proximity, and there are no false alarms. Prediction2, on the other hand, detects both events with perfect accuracy, but introduces a single false event which lasts four seconds. The classification error metric will yield the same error for both estimations. However, it is clear that model1 is the preferable model in this case. The values obtained by the two approaches are presented in

Table 2. The classification error metric fails to identify the better model. However, the change point detection performance metric does a good job in distinguishing between the two models. The algorithm of the change point detection performance metric will be explained in the next section.

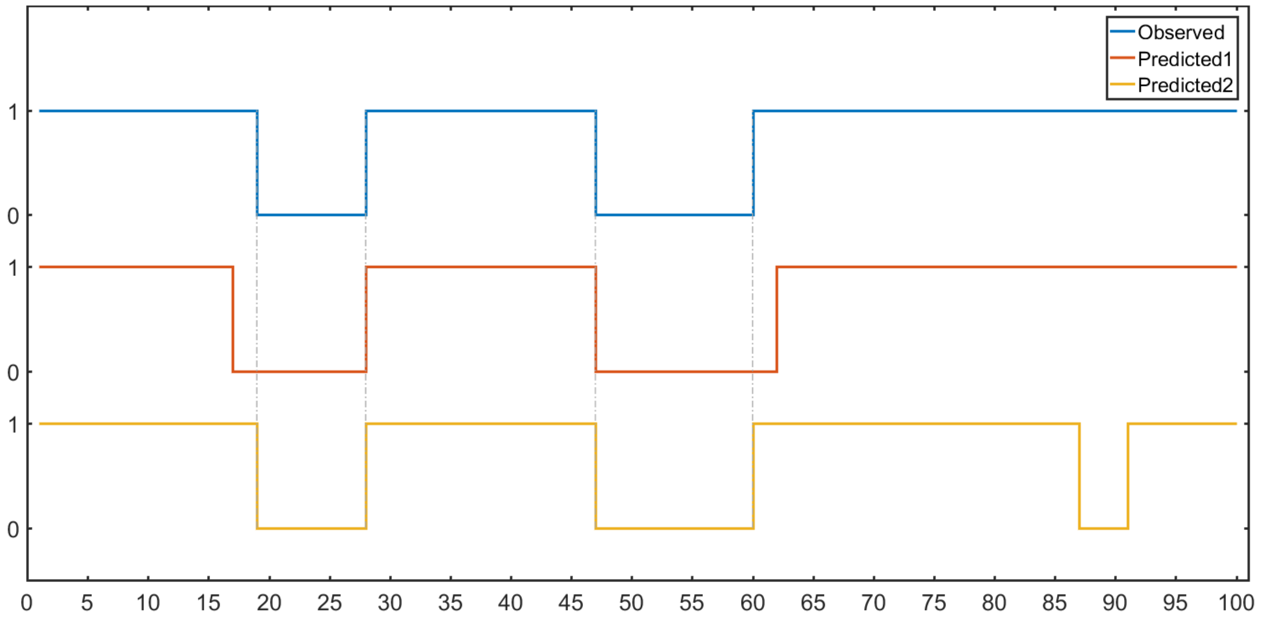


Fig. 25. Example 1: Importance of using the right performance metric for model selection.

Table 2. Classification Error and Change Point Detection Performance Metric Results

	Classification Error	CPDPM
Prediction1	0.04	0.22
Prediction2	0.04	2.00

As a second example, as shown in Fig. 26, model1 predicts first and second events perfectly, but has a two second delay in detecting the third start point. Model2, on the other hand detects all three events with perfect accuracy, while having a false event at the beginning, which lasts for two seconds. Since the total difference between the observed and predicted cases are each two seconds in each model, the classification error metric will yield the same result for both. However, it is clear that model1 performs better, as it has no false event. The change point detection performance metric is able to distinguish between the two, as presented in Table 3.

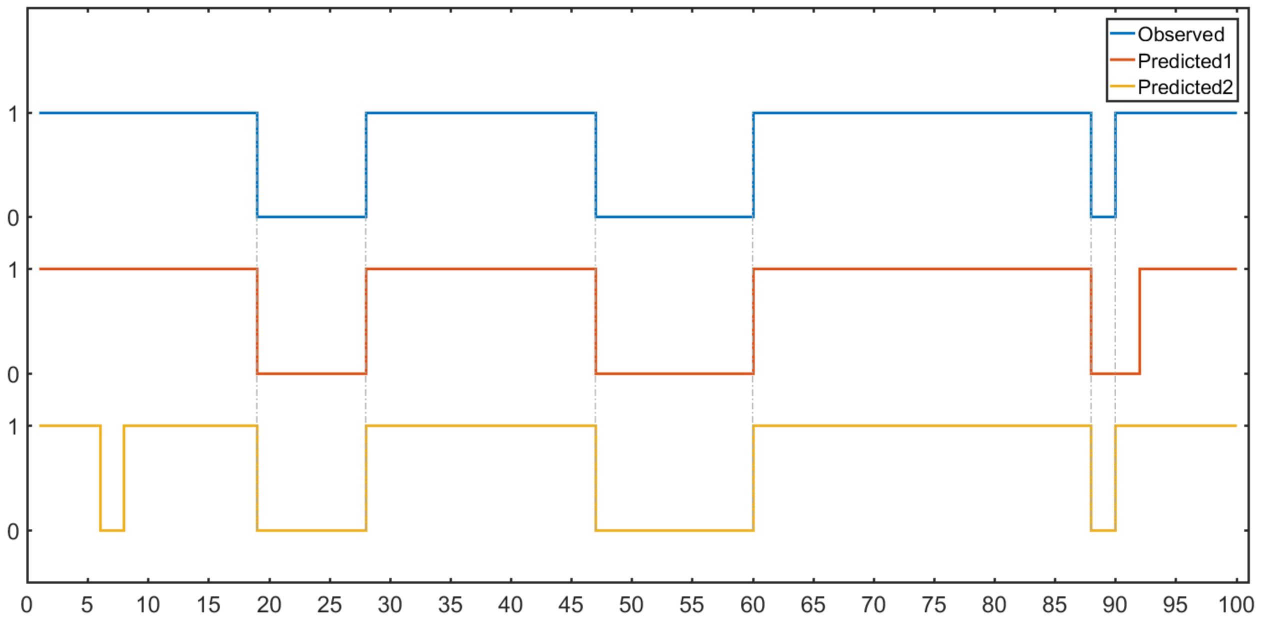


Fig. 26. Example 2: Importance of using the right performance metric for model selection.

Table 3. Classification Error and Change Point Detection Performance Metric Results

	Classification Error	CPDPM
Prediction1	0.02	0.11
Prediction2	0.02	2.00

A third and final example to illustrate the benefit of the change point detection performance metric is shown in Fig. 27. Model1 is able to detect all three events, albeit with some time difference. Model2, however, is only able to detect the first two events, and misses the last one. Since, the total time difference is the same in both models, the classification error metric cannot differentiate between the two models. However, the change point detection performance metric duly penalizes model2 for missing an event, as shown in Table 4.

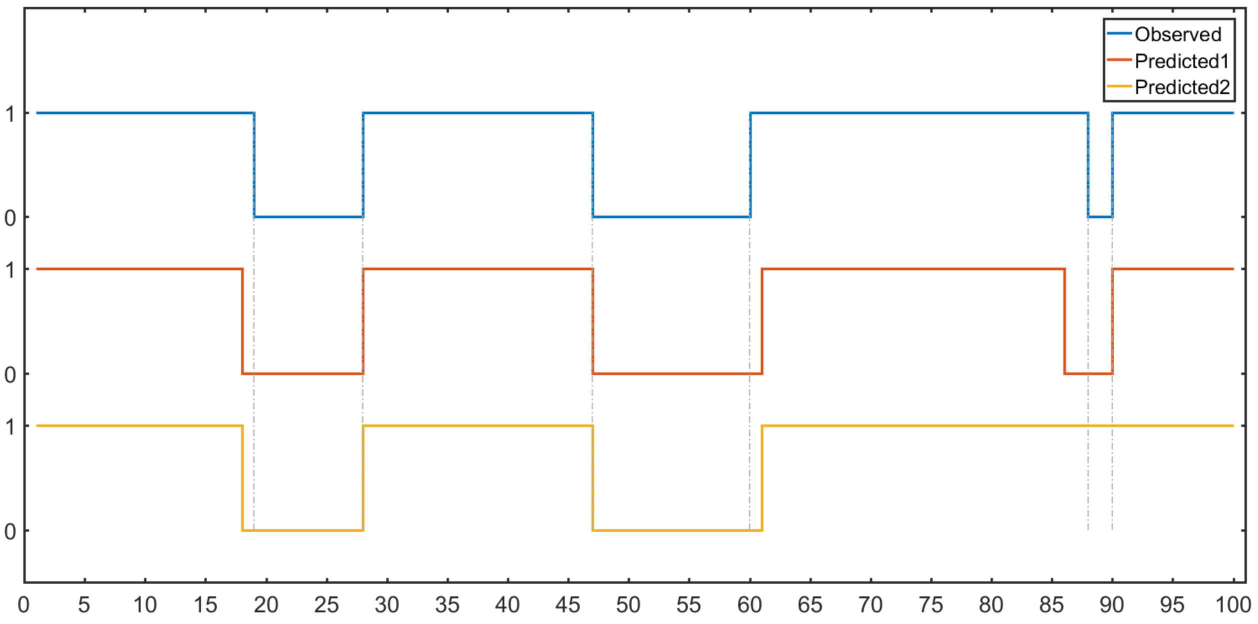


Fig. 27. Example 3: Importance of using the right performance metric for model selection.

Table 4. Classification Error and Change Point Detection Performance Metric Results

	Classification Error	CPDPM
Prediction1	0.04	0.23
Prediction2	0.04	2.11

5.1.1 Error Function

In order to mitigate the inefficiency of the classification error and to tailor the measure of accuracy to the needs of this research, a novel performance evaluation is applied which makes use of the so called "*error function.*" The error function is a special function of sigmoid shape whose output increases as the input is increased. The method proposed will only quantify the accuracy regarding the change points, i.e. the error function will be applied to the time differences between observed and corresponding predicted points. This way, the proximity of the predicted point to the observed one will be assessed. The closer the estimation to the actual case, the smaller the error function will be; also, any missing observed points, and false alarms (either being a stop or start) will be penalized heavily.

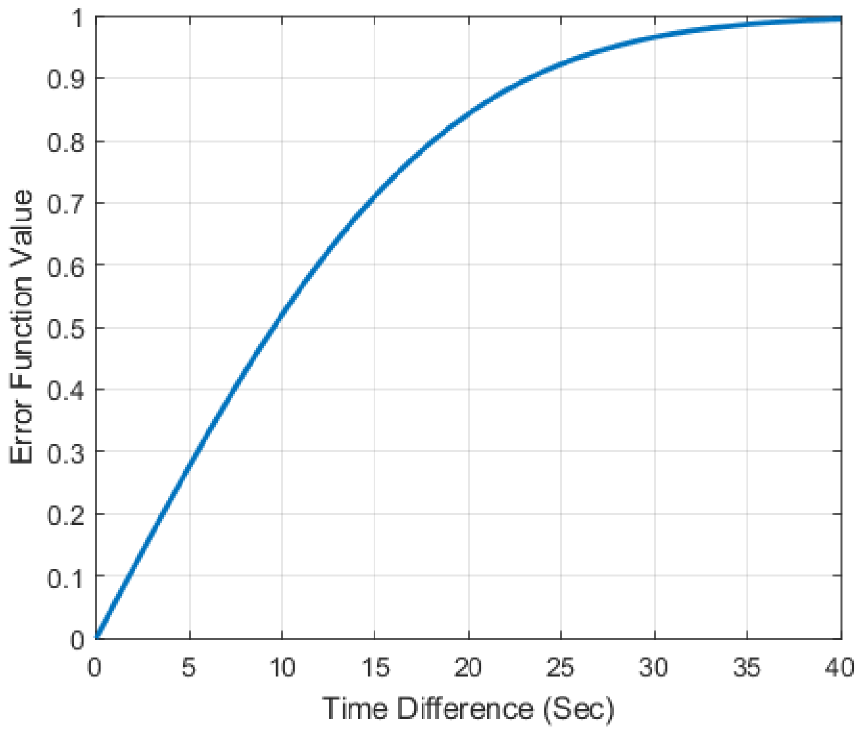


Fig. 28. The error function values corresponding to time differences.

The function value increases with a decreasing slope and flattens out at value 2, yielding an output of 0.9953. For larger values than 2, the curve becomes asymptotic to 1. In the context of the research being conducted, this would mean that an error value of 1 (largest value) is obtained when the estimated point is two seconds or further apart. The absolute value of the time difference between observed and the estimated points is taken, meaning that predicting a stop or start point either *t* seconds before or after the actual point will yield the same error. Hence, only the positive quadrant of the error function is used. The error function is defined in (67).

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,dt \tag{67}$$

Table 5. Error function values of time differences up to 20s when the values are divided by 20

Time Difference (sec)	**Error Value *erf(Δt/20)***	**Time Difference (sec)**	**Error Value *erf(Δt /20)***
0	0		
1	0.06	**11**	0.56
2	0.11	**12**	0.60
3	0.17	**13**	0.64
4	0.22	**14**	0.68
5	0.28	**15**	0.71
6	0.33	**16**	0.74
7	0.38	**17**	0.77
8	0.43	**18**	0.80
9	0.48	**19**	0.82
10	0.52	**20**	0.84

Due to the features having a large overlap between the two states, it is almost impossible to detect the stop and start points perfectly. The goal is to be able to find the models which can closely predict change points, with the least number of missing true points, and the least number of false positives. Thus, in order to add some tolerance for estimations, and to provide some space of variance, the time difference between the estimated and true points is divided by 20. The time differences and their corresponding error function values between 0s and 20s are shown in Table 5 and in Fig. 28. Dividing by 20 will lead to more variation between the different models, which will enable one to have a better assessment over the performance of models. The error increases rapidly at the beginning and slows down as the time difference increases. This serves the purpose of differentiating between models based on the time difference between the predicted and the observed points very well. Any missing observed points and any false alarms are penalized with the largest error function value of one.

5.1.2 The Algorithm for Change Point Detection Performance Metric

In this section, the algorithm for the novel performance evaluation method is presented. The evaluation method is named “change point detection performance metric” (CPDPM). In the following calculations, the actual set will be denoted by Q and the estimated set will be denoted by $\widehat{Q}$. Observed and Estimated states of a trip will be denoted as sets containing values of ones and zeros:

$$\begin{aligned} Q &= \{1,0,0,\dots,1,1,1,1,\dots,1\} \\ \hat{Q} &= \{1,0,1,\dots,1,1,0,0,\dots,1\} \end{aligned} \tag{68}$$

The zeros (0) denote vehicle standstill at each instance, and the ones (1) denote vehicle being in motion. When a vehicle stops or starts moving, it stays in that state for a while. To detect the change points (stop and start points of a vehicle), the algorithm shown in (69) is applied:

$$Q_i - Q_{i-1} \in \{0,1,-1\} \tag{69}$$

The difference between point *i* and point *i-1* can only have three distinct values. “0” denotes no change in the state of the vehicle. “1” denotes that the vehicle was in standstill and started moving. The index of this point will be stored in the Motion Start (*M*) set. A value of “-1” denotes that the vehicle was in motion and has stopped at this instance. The index of this point will be stored in the Stop (*S*) set.

$$M = \{i | Q_i - Q_{i-1} = 1\}$$
$$S = \{i | Q_i - Q_{i-1} = -1\} \tag{70}$$
$$|S| = |M| = N$$

Where $i = 1,2,3, \dots, |Q|$.

Thus, *M* is the set of indices when vehicle starts to move from a standstill, and *S* is the set of indices when the vehicle stops from being in motion. The sizes of the *S* and *M* are equal, where *N* denotes the number of elements in the sets of indices of stop and start points. The same sets are created for the estimated points as well.

$$\widehat{M} = \left\{j \middle| \hat{Q}_j - \hat{Q}_{j-1} = 1\right\}$$
$$\hat{S} = \left\{j \middle| \hat{Q}_j - \hat{Q}_{j-1} = -1\right\} \tag{71}$$
$$\left|\hat{S}\right| = \left|\widehat{M}\right| = \widehat{N}$$

Where $j = 1,2,3, \dots, \left|\hat{Q}\right|$.

Without loss of generality, the number of points in the observed set will be assumed to be less than or equal to those in the estimated set.

$$N \leq \widehat{N} \tag{72}$$

The Hungarian Method is used to match each observed pair of stop and start points to the closest estimated pair. Let *P* be the pair of stop and start points that defines an event. Then *P* is defined as:

$$P = \{P_1, P_2, \dots, P_N\} \tag{73}$$

Where each element P_n of P is a pair of consecutive stop and start points:

$$P_n = (S_n, M_n) \tag{74}$$

Where $n = 1,2,3, \dots, N$.

The same is also defined for the estimated set:

$$\begin{gathered} \hat{P} = \{\hat{P}_1, \hat{P}_2, \dots, \hat{P}_{\hat{N}}\} \\ \hat{P}_{\hat{n}} = (\hat{S}_{\hat{n}}, \widehat{M}_{\hat{n}}) \end{gathered} \tag{75}$$

Where $\hat{n} = 1,2,3, \dots, \widehat{N}$.

The cost C for each assignment between an observed and estimated pair is calculated to be the sum of the error functions of the time differences between corresponding stop and start points. For simplicity, the cost value will be abbreviated as $C_{n\hat{n}}$.

$$C(P_n, \hat{P}_{\hat{n}}) = C_{n\hat{n}} = erf(|S_n - \hat{S}_{\hat{n}}|) + erf(|M_n - \widehat{M}_{\hat{n}}|) \tag{76}$$

The Hungarian Algorithm will need to assign exactly one observed pair to exactly one estimated pair. Thus, if the sets do not have equal number of points, dummy pairs will be added to the lesser set (according to assumption this is the observed set). The cost for these pairs will be assigned a large penalty value C_{max}. δ is an indicator assignment which takes the value of one when two values are paired, and zero otherwise.

Let N <= $\widehat{N}$

$$P' = P \cup \{P_{N+1}, P_{N+2}, \dots, P_{\widehat{N}}\} \tag{77}$$

Objective:

$$\min \sum_{\hat{n}}^{\widehat{N}} \sum_{n}^{N} C_{n\hat{n}} \delta_{n\hat{n}} \tag{78}$$

Subject to:

$$\sum_{n}^{N} \delta_{n\hat{n}} = 1$$
$$\sum_{\hat{n}}^{\hat{N}} \delta_{n\hat{n}} = 1 \qquad (79)$$
$$\delta_{n\hat{n}} \in \{0,1\}$$

$$C_{n\hat{n}} = \begin{cases} erf\left(\left|S_n - \hat{S}_{\hat{n}}\right|\right) + erf\left(\left|M_n - \widehat{M}_{\hat{n}}\right|\right), n \leq N \\ C_{max}, n > N \end{cases} \qquad (80)$$

The assignment of pairs is done such that, for each observed pair, there is a single match in the estimated set. The larger the time difference between the change points of the matched pairs, the larger the error will be. Any false stop or start points are assigned the largest penalty, which is 1. The best model is the one that yields the lowest total error.

A performance table can be established at this stage. The performance table is a modified confusion matrix and is constituted of number of observed points, a number of estimated points, a number of estimations that correctly predict observed points within reasonable time period, a number of false alarms, and a number of missed true points.

The way the above variables are calculated is done by making use of the observed and predicted pairs obtained earlier, and the time difference between these pairs. The following calculations can be done for both the stop and start points separately, and, for simplicity, they will be referred to as point. If the time difference for a point in the pair is below a threshold, the matching is assumed to be correct and the number of “True” points is increased by one. If it is further than the threshold, or if a predicted point has no matching in the observed set, then it is assumed to be a false positive, and the “False” number of points is increased by one. At the end, the number of “Missing” is the number of observed points that are not matched to any point in the predicted set. The number of total points in the observed set is recorded in the “Obs” variable, while the total number of predicted points is shown in the “Pred” variable. In the experimental results, the performance matrix only for the stop points is presented.

After computing the performance matrix variables, the F1 score can also be computed. The F1 score is calculated from the precision and recall metrics. Recall, also called sensitivity, measures the true positive rate, meaning the ratio of true predictions divided by the total number of observations. Normally, recall is calculated as the True Positives divided by the sum of True Positives and False Negatives. Here, instead of using the False Negatives, the “Miss” variable is used. The precision measures the positive predictive value, meaning the ratio of True Positives divided by the total number of predictions (True Positive plus False Positive). The calculations in Eq. (81) are carried out to reach the F1 score. The F1 score provides an overall metric that can be used to compare the performance of the classification error metric with the CPDPM methods.

$$\begin{gathered} Miss = Obs - True \\ False = Pred - True \\ Precision = True/(True + False) \\ Recall = True/(True + Miss) \\ F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \end{gathered} \tag{81}$$

Given the purposes of this research, the noisiness involved with the data, and the fact that the GPS and OBD data collection rate is one second, which can lead to a possible lag of one second between the actual stopping of the vehicle and the logging of the data from GPS or OBD, ten seconds of duration was identified as the upper bound for an estimated point to be accepted as a true match to the observed point. This is a threshold that can be changed, and can be made smaller to increase the strictness of the evaluation. As is shown in the Analysis of the Results section, the majority of the predicted correct matches were found within five seconds of the observed point. Also, based on the CPDPM, any time difference between the observed and predicted points is duly penalized. The reason for determining a time period is purely to establish the number of correctly predicted points, the false alarms, and the missed points in the performance matrix.

5.2 Experiments and Results

In the following sections, the implementation details and the experimental results of the support vector machines, the recurrent neural networks, and the hidden Markov models are presented. In the experiments, an ML method is first trained on the training data, and the trained model is applied to the trips in the validation data. The classification error and the CPDPM are calculated for each of the ten trips in the validation set. Finally, the average classification error and the average CPDPM are calculated. Average CPDPM is calculated as the sum of all CPDPM values from each validation trip, divided by the total number of stops. The mean classification error is computed as the total number of points misclassified, divided by the total number of points in all of the individual trips in the validation set. The CPDPM values presented in the tables are, in a sense, the average CPDPM value that each stop point endures. The lower the value in each metric, the better the model is. The best model selected according to a metric is shown under “Metric” column. The top half shows the best results according to the classification error metric, and the bottom half shows the best results according to the CPDPM. At the end, the best model according to each metric is selected and is applied to the testing trips. Once testing is done, the same performance metrics are calculated for each of the twenty trips in the testing set, and the average classification error and the average CPDPM values are calculated the same way as before, and are shown in the tables. The aims are to find the best performing model and to test whether the CPDPM does a better job than the classification error in terms of finding the better model.

The tables also show the total number of observed stop points (Obs), the total number of predicted stop points (Pred), the true estimations (True), the missed stop points (Miss), and the false positives (False). The F1 score of the stop points are shown in F1Stop column. Some model-specific parameters are also shown, such as the *C* and sigma value for SVM, the number of delays for NN, and the binning thresholds used for HMM.

Other than the parameters of the specific machine learning models, the features used and the length of the window size also play important roles in the success of finding the change points. For this, several window sizes are tried. After individual tests with different features, and the knowledge from the feature selection presented earlier, two main features are decided upon. First is the range, and the results using only this feature are shown with number "1" under the Feature column in the results tables. The second feature is the absolute difference between consecutive range values, used in the Hidden Markov models. For SVM and neural networks, four features are selected: the range, the absolute difference of consecutive ranges, the interquartile range, and the standard deviation. The second feature set is shown with number "2" under the Feature column in the tables.

The schema for model development and testing is given in Fig. 29. This schema provides an overall summary on how the model training, validation, and testing is done. The model parameters are specific to each model such as the sigma, and box constraint values for RBF SVM. The optimum model is the one that yields the lowest validation error. At the end, state prediction is done on the testing data using the optimum model, and the testing error is found.

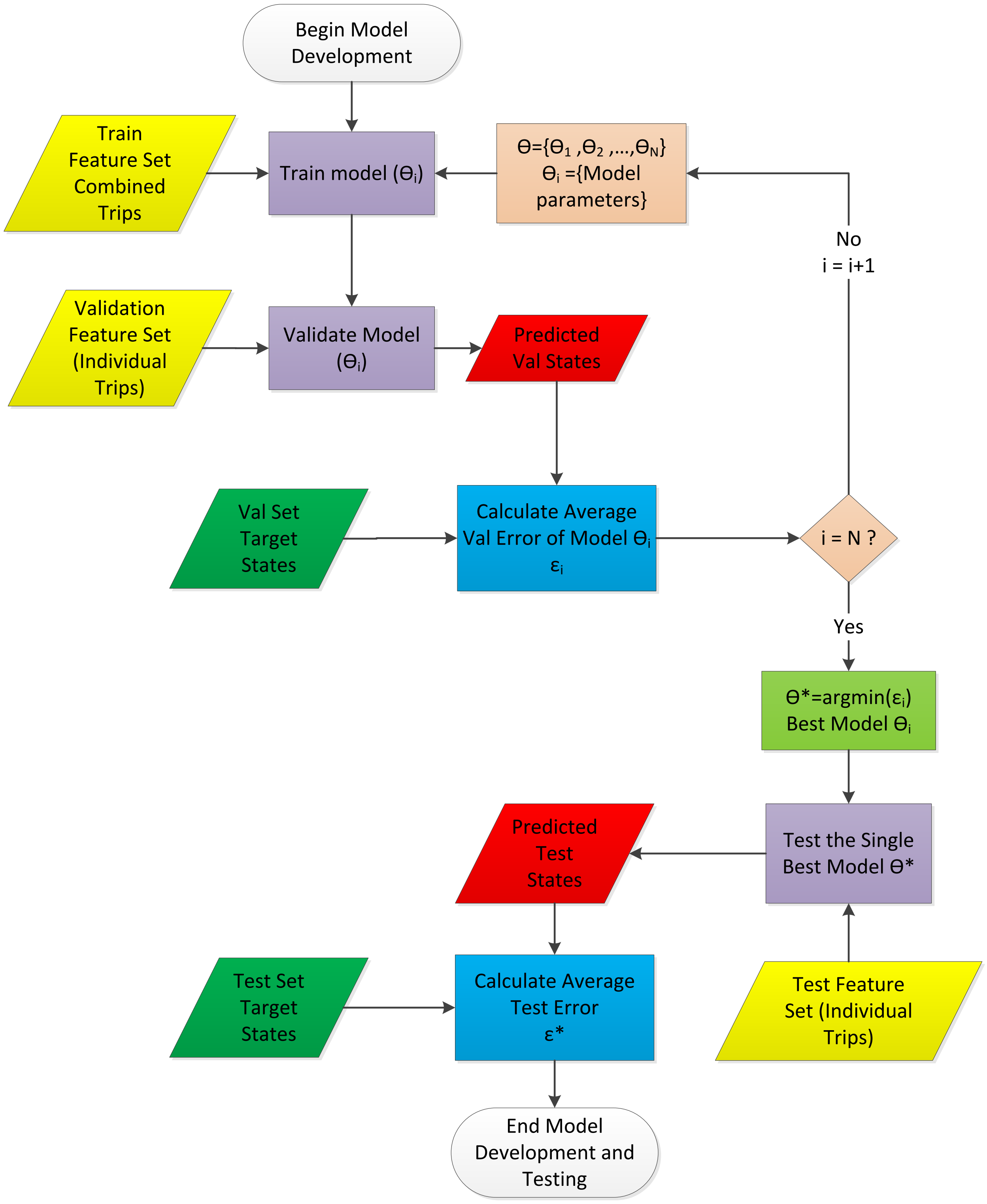

Begin Model Development
Train Feature Set Combined Trips
Train model (Θ_i)
Θ={Θ_1 ,Θ_2 ,…,Θ_N} Θ_i ={Model parameters}
No
i = i+1
Validation Feature Set (Individual Trips)
Validate Model (Θ_i)
Predicted Val States
Val Set Target States
Calculate Average Val Error of Model Θ_i ε_i
i = N ?
Yes
Θ*=argmin(ε_i) Best Model Θ_i
Predicted Test States
Test the Single Best Model Θ*
Test Set Target States
Calculate Average Test Error ε*
Test Feature Set (Individual Trips)
End Model Development and Testing

Fig. 29. The schema for model development and testing.

5.2.1 Support Vector Machines

Three different support vector machines were tested: Linear, Polynomial, and Radial Basis Function Kernel. The linear SVM is the most basic one, where the kernel function used to compute the Gram matrix is the dot product, as shown in Eq. (82). No transformation is done on the data, and the resulting boundary between classes will be a linear line. The only parameter to be optimized is the box constraint.

$$G(x_1, x_2) = x_1{}' x_2 \tag{82}$$

The Gram matrix of a set of *n* vectors is an *n*-by-*n* matrix with element (*j*,*k*) defined as in Eq. (83).

$$G\left(x_j, x_k\right) = < \phi\left(x_j\right), \phi(x_k) > \qquad \{x_1, .., x_n;\ x_j\ \in\ R^p\} \tag{83}$$

Where,

<*a*,*b*>	Inner product between *a* and *b*
ϕ	Kernel function used
p	Number of dimensions of a vector

Thus, the Gram matrix is the inner product matrix of the kernel ϕ transformed vectors.

Polynomial SVM uses the polynomial kernel that transforms the data, as shown in Eq.(84). The polynomial order *p* determines how complex the polynomial terms will get. Using a big value in *p* might cause the computations to take too long, and overfit the training data. Values for *p* that are tested are 1, 2, and 3. The other parameter to be optimized is, again, the box constraint.

$$G(x_1, x_2) = (1 + x_1{}' x_2)^p \tag{84}$$

The radial basis function (RBF) kernel SVM is one of the most well-known SVM methods, as it transforms the data into a higher space, which enables classification of nonlinear data. The kernel computation shown in Eq. (85) allows for the so-called “kernel trick”, which allows SVM to operate in the transformed predictor space to find a separating hyperplane. The parameters to be optimized involve the box constraint and the kernel scale.

$$G(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{\sigma^2}\right) \tag{85}$$

An exhaustive search is done to find the optimum parameters of the RBF SVM. The box constraint values and the sigma values used by RBF SVM are presented in Table 6. In the following subsections, only the RBF SVM results are presented, as the other methods did not do any better than the RBF SVM. MATLAB fitcsvm function was utilized for training and testing SVM models.

Table 6. Parameters that were used in the exhaustive search of RBF SVM.

Parameter	Values
Window Size	0.2, 0.6, 1.0, 2.0, 3.0
Box Constraint	0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100, 200
Sigma	0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100

5.2.1.1 Dataset 1: Driver 1 - Toyota Prius 2007 - Motorola XT 1063

Table 7. Validation of RBF SVM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063

Metric	Feature	Win	C	Sigma	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	2.0	0.01	50	0.034	2.26	0.618	125	244	114	11	130
	2	3.0	10	5	0.028	2.11	0.647	125	240	118	7	122
CPDPM	1	3.0	0.01	0.05	0.041	1.11	0.741	125	161	106	19	55
	2	3.0	50	100	0.031	2.05	0.648	125	233	116	9	117

Table 8. Testing of RBF SVM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063

Metric	Feature	Win	C	Sigma	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	2.0	0.01	50	0.028	2.46	0.593	251	525	230	21	295
	2	3.0	10	5	0.025	2.77	0.548	251	559	222	29	337
CPDPM	1	3.0	0.01	0.05	0.033	0.92	0.750	251	298	206	45	92
	2	3.0	50	100	0.026	2.95	0.543	251	581	226	25	355

5.2.1.2 Dataset 2: Driver 2 - Toyota Camry 2012 - LG G4

Table 9. Validation of RBF SVM for Driver 2 - Toyota Camry 2012 - Lg G4

Metric	Feature	Win	C	Sigma	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	3.0	0.01	0.1	0.031	3.12	0.562	36	85	34	2	51
	2	3.0	0.5	0.5	0.028	14.12	0.219	36	283	35	1	248
CPDPM	1	3.0	0.05	0.5	0.031	3.01	0.571	36	83	34	2	49
	2	0.2	5	0.001	0.087	2.25	0.268	36	46	11	25	35

Table 10. Testing of RBF SVM for Driver 2 - Toyota Camry 2012 - Lg G4

Metric	Feature	Win	C	Sigma	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	3.0	0.01	0.1	0.070	3.77	0.429	137	348	104	33	244
	2	3.0	0.5	0.5	0.055	15.35	0.197	137	1152	127	10	1025
CPDPM	1	3.0	0.05	0.5	0.070	4.00	0.417	137	367	105	32	262
	2	0.2	5	0.001	0.134	2.54	0.194	137	172	30	107	142

5.2.1.3 Dataset 3: Driver 3 - Mazda 3 - Samsung Galaxy S3

Table 11. Validation of RBF SVM Driver 3 - Mazda 3 - Samsung Galaxy S3

Metric	Feature	Win	C	Sigma	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	3.0	0.01	0.1	0.031	3.12	0.272	36	85	34	2	51
	2	3.0	0.5	0.5	0.028	14.12	0.265	36	283	35	1	248
CPDPM	1	3.0	0.05	0.5	0.031	3.01	0.436	36	83	34	2	49
	2	0.2	5	0.001	0.087	2.25	0.179	36	46	11	25	35

Table 12. Testing of RBF SVM Driver 3 - Mazda 3 - Samsung Galaxy S3

Metric	Feature	Win	C	Sigma	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	3.0	0.01	0.1	0.070	3.77	0.300	137	348	104	33	244
	2	3.0	0.5	0.5	0.055	15.35	0.267	137	1152	127	10	1025
CPDPM	1	3.0	0.05	0.5	0.070	4.00	0.373	137	367	105	32	262
	2	0.2	5	0.001	0.134	2.54	0.154	137	172	30	107	142

5.2.2 Recurrent Neural Networks

The time series data has specific characteristics that cannot be utilized well with static methods such as SVM. For example, the interdependence and correlation between the consecutive data points is lost when they are treated as individual entities. Also, the regular feedforward neural network is not capable of capturing the time information. On the other hand, dynamic recurrent neural networks have proven to be good estimators in time series predictions. There are many different versions of such networks. The type of neural network used in this dissertation is called the Nonlinear Autoregressive Network with Exogenous Inputs (NARX). The defining function of NARX is given in Eq. (86) where the dependent variable *y(t)* is regressed on previous predictions of the output and the previous values of the independent (exogenous) inputs. A NARX model representation is given in Fig. 30 [43].

$$y(t) = f(y(t-1), \dots, y\left(t-n_y\right), u(t), u(t-1), \dots, u(t-n_u)) \tag{86}$$

Where,

$y(t)$	Current prediction
$y(t-n_y)$	Previous predictions
$u(t)$	Current input
$u(t-n_u)$	Previous inputs

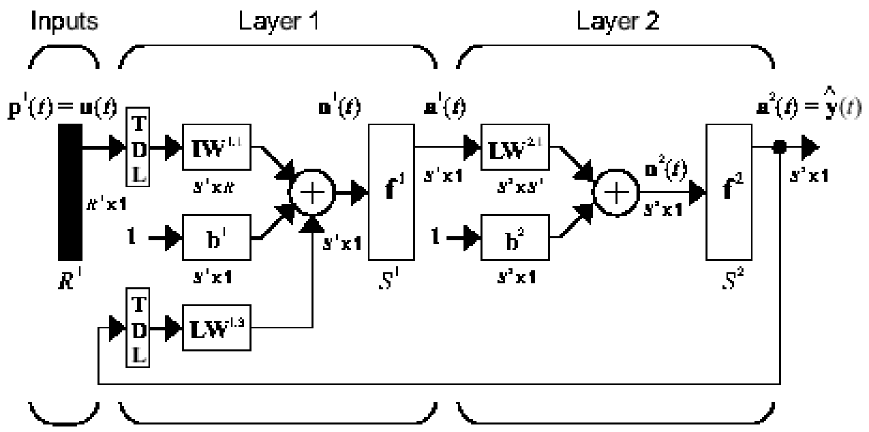


Fig. 30. NARX model representation with delayed feedback loops and delayed inputs [43].

There are two main training procedures in the NARX model. One is where the true outputs are used; this is also referred to as series-parallel or open loop architecture. The other approach is called parallel or closed loop architecture. It uses the predicted outcomes as feedback loops to the input layer. These two approaches are depicted in Fig. 31. In this dissertation, closed loop architecture is used, since it is assumed that the target variables will not be available during testing, and it was important not to overfit the training data.

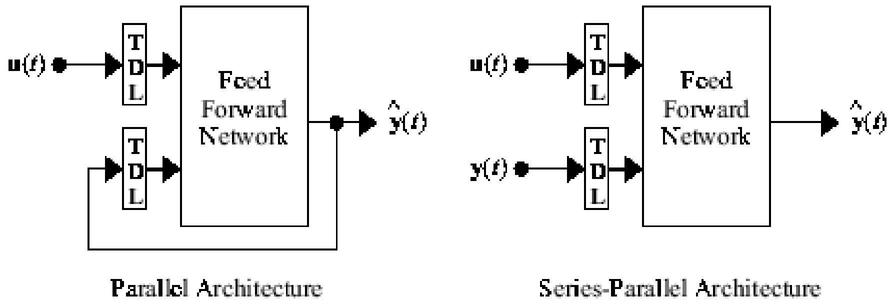


Fig. 31. Closed and open loop architectures, respectively [43].

Both the input and the targets were delayed by same number of points, if there was any delay desired. The parameters and their values, which were used in the experiments, are shown in Table 13. The delay allowed the system to have memory of the past and thus to take advantage of the time series data characteristics. MATLAB Neural Networks Toolbox was utilized. The network was updated by making use of the backpropagation and Levenberg Marquardt algorithms.

A sample NARX model application is shown in Fig. 32. Here, a model with five tapped delay lines is shown with an input of four components. The hidden layer has ten nodes, and the sigmoid transition function is utilized. The output has one component, which is the state prediction for the current input ***u****(t)*. The results of applying NN to data from different vehicles are presented next.

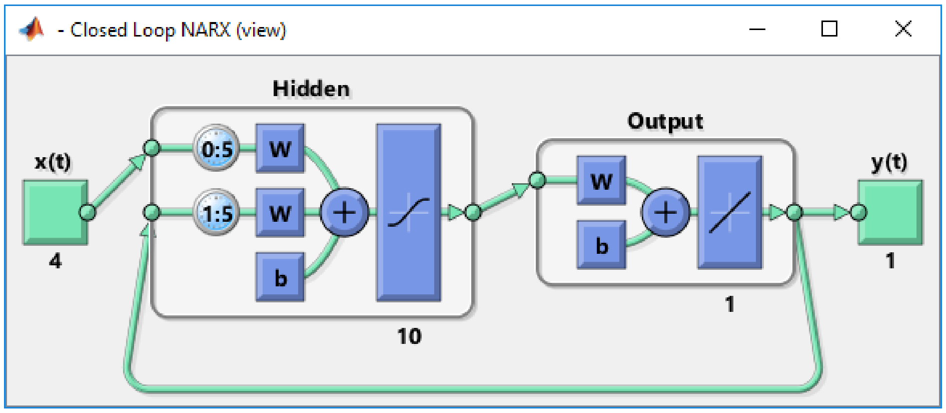


Fig. 32. Closed loop NARX with 5 tapped delay lines (MATLAB NN Toolbox).

Table 13. The window sizes, and number of delays used in NARX exhaustive search.

	Values
Window	0.2, 0.6, 1.0, 1.5, 3.0
Delays	1, 3, 5, 10
Hidden Nodes	10
Hidden Layers	1

5.2.2.1 Dataset 1: Driver 1 - Toyota Prius 2007 - Motorola XT 1063

Table 14. Validation of NN for Driver 1 - Toyota Prius 2007 - Motorola XT 1063.

Metric	Feature	Window	Delays	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.0	1	0.030	1.10	0.730	125	168	107	18	61
	2	2.0	1	0.025	0.49	0.891	125	131	114	11	17
CPDPM	1	3.0	1	0.037	0.88	0.768	125	146	104	21	42
	2	3.0	3	0.026	0.42	0.911	125	123	113	12	10

Table 15. Testing of NN for Driver 1 - Toyota Prius 2007 - Motorola XT 1063.

Metric	Feature	Window	Delays	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.0	1	0.026	0.85	0.759	251	297	208	43	89
	2	2.0	1	0.024	0.63	0.827	251	276	218	33	58
CPDPM	1	3.0	1	0.030	0.77	0.767	251	255	194	57	61
	2	3.0	3	0.026	0.65	0.799	251	247	199	52	48

5.2.2.2 Dataset 2: Driver 2 - Toyota Camry 2012 - LG G4

Table 16. Validation of NN for Driver 2 - Toyota Camry 2012 - Lg G4.

Metric	Feature	Window	Delays	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.0	5	0.021	5.90	0.372	36	136	32	4	104
	2	2.0	5	0.018	1.15	0.729	36	49	31	5	18
CPDPM	1	3.0	3	0.024	0.74	0.776	36	31	26	10	5
	2	1.0	3	0.018	0.55	0.838	36	38	31	5	7

Table 17. Testing of NN for Driver 2 - Toyota Camry 2012 - Lg G4.

Metric	Feature	Window	Delays	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.0	5	0.056	7.34	0.281	137	602	104	33	498
	2	2.0	5	0.037	1.33	0.622	137	178	98	39	80
CPDPM	1	3.0	3	0.062	1.19	0.612	137	95	71	66	24
	2	1.0	3	0.053	1.04	0.683	137	144	96	41	48

5.2.2.3 Dataset 3: Driver 3 - Mazda 3 - Samsung Galaxy S3

Table 18. Validation of NN Driver 3 - Mazda 3 - Samsung Galaxy S3.

Metric	Feature	Window	Delays	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.0	3	0.048	1.73	0.612	74	109	56	18	53
	2	3.0	10	0.040	14.66	0.196	74	599	66	8	533
CPDPM	1	0.6	3	0.049	1.20	0.727	74	91	60	14	31
	2	0.2	5	0.052	1.05	0.748	74	81	58	16	23

Table 19. Testing of NN Driver 3 - Mazda 3 - Samsung Galaxy S3.

Metric	Feature	Window	Delays	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.0	3	0.063	1.98	0.544	160	248	111	49	137
	2	3.0	10	0.075	13.54	0.161	160	1178	108	52	1070
CPDPM	1	0.6	3	0.076	1.45	0.541	160	173	90	70	83
	2	0.2	5	0.073	1.34	0.556	160	160	89	71	71

5.2.3 Hidden Markov Models

Hidden Markov Models (HMM) have proven to be effective in capturing the time related information such as correlation between sequential data points. In HMMs, each data point is a state which is hidden (latent). The measurement that can be seen is called an observation or an emission. Each hidden state produces one observation. The observations are dependent on the current state directly, and dependent on the previous states indirectly via the hidden states. The Hidden Markov Model representation of a trip's data is given in Fig. 33. Here, the Z vector contains the hidden states representing either being in motion or being stationary. The observations are the acceleration data that are measured by the smartphone, expressed as the vector X.

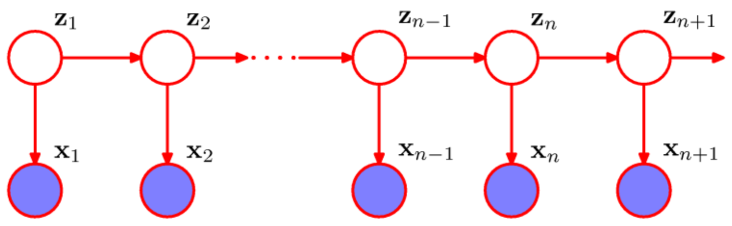


Fig. 33. HMM Representation. Z is the hidden states, X is the observations vector [41].

The application of Hidden Markov Models was done by taking a single feature and slicing this feature into bins. This transformed the continuous features into discrete values. Three different binning models with different thresholds were used. The features used were the range of each window and the absolute value of the difference between consecutive range values. The thresholds used for binning are shown in Table 20. The reason for using the last bin with upper bound of 100 is to capture any outliers.

Table 20. Binning the continuous features into discrete values

Bin Type	The Thresholds
Bins 1	0-0.5, 0.5-1, 1-2, 2-3, 3-100
Bins 2	0-0.05, 0.05-0.1, 0.1-0.2, 0.2-0.3, 0.3-0.4, 0.4-0.5, 0.5-1, 1-2, 2-3, 3-100
Bins 3	0-0.05, 0.05-0.1, 0.1-0.15, 0.15-0.2, 0.2-0.3, 0.3-0.4, 0.4-0.5, 0.5-0.6, 0.6-0.7, 0.7-0.8, 0.8-0.9, 0.9-1, 1-2, 2-3, 3-100

After binning is complete, the transition and emission probability matrices are found by using the maximum likelihood method in which the labels of each data point (being in standstill or motion state) are used to train the model. Since some stops are very short (1 or 2 seconds), choosing the right window size plays an important role in assuring the accuracy of the detection model. A large window might have missed these small stops, and a very small window might introduce many false stops. Thus, an exhaustive search is performed to find the optimum window size and the optimum thresholds for binning. The window sizes vary from 0.2 to 10 seconds with 0.2 second increments. In the validation step the optimum combination of window size and binning thresholds are determined. Finally, in the testing phase, the optimum model is applied to the testing trips and the accuracy of the model is found.

Fig. 34 to Fig. 36 show the state predictions of the best HMM models, where the best models are selected based on the CPDPM. In the plots, green lines represent the observed true states of the vehicle, and the red lines represent the predicted states. Each of the first 15 trips in the testing sets are drawn separately in their respective subplots, enumerated 1 to 15. Since there are two states, the upper segments of the lines show vehicle motion periods, while the lower segments of the lines show the standstill periods. The vertical lines represent the change points.

The x axis shows the time in Min:Sec format. Since some trips were much longer than the rest, to keep it consistent, each trip is truncated at 20 minutes.

The developed models and the testing results for the three datasets are presented in the following subsections.

5.2.3.1 Dataset 1: Driver 1 - Toyota Prius 2007 - Motorola XT 1063

Table 21. Validation of HMM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063.

Metric	Feature	Window	Bin	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	0.6	2	0.028	1.57	0.695	125	203	114	11	89
	2	0.4	2	0.027	0.46	0.896	125	134	116	9	18
CPDPM	1	7.6	1	0.054	0.79	0.803	125	114	96	29	18
	2	0.4	3	0.027	0.45	0.899	125	133	116	9	17

Table 22. Testing of HMM for Driver 1 - Toyota Prius 2007 - Motorola XT 1063.

Metric	Feature	Window	Bin	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	0.6	2	0.024	1.53	0.683	251	408	225	26	183
	2	0.4	2	0.023	0.48	0.858	251	257	218	33	39
CPDPM	1	7.6	1	0.044	0.85	0.760	251	207	174	77	33
	2	0.4	3	0.023	0.48	0.857	251	258	218	33	40

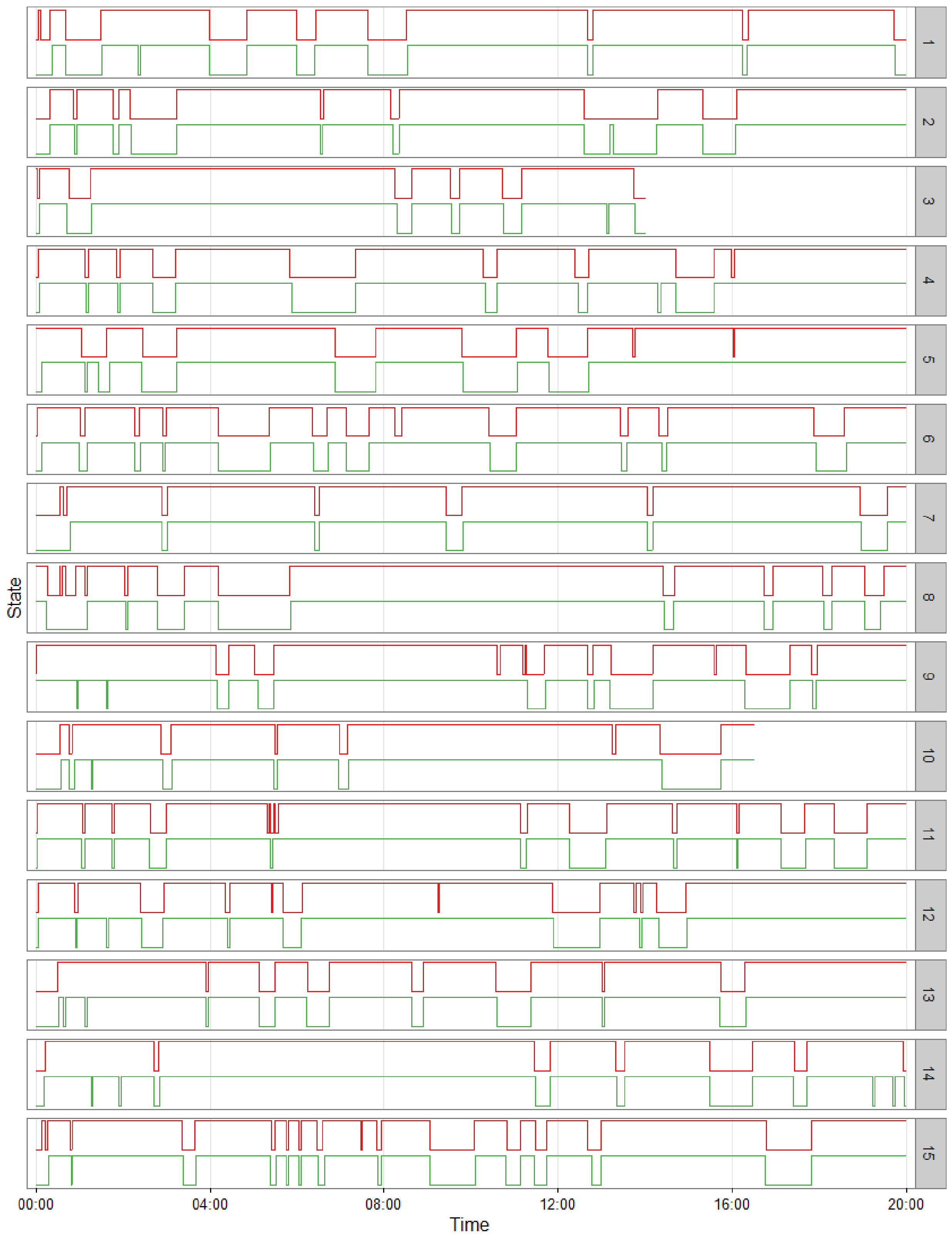


Fig. 34. Prius state estimation of test trips. Red: Prediction, Green: Observed.

5.2.3.2 Dataset 2: Driver 2 - Toyota Camry 2012 - LG G4

Table 23. Validation of HMM for Driver 2 - Toyota Camry 2012 - Lg G4.

Metric	Feature	Window	Bin	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	10	2	0.039	1.21	0.667	36	42	26	10	16
	2	1.6	2	0.032	0.99	0.630	36	37	23	13	14
CPDPM	1	8.8	3	0.048	1.04	0.718	36	42	28	8	14
	2	1.4	2	0.034	0.91	0.640	36	39	24	12	15

Table 24. Testing of HMM for Driver 2 - Toyota Camry 2012 - Lg G4.

Metric	Feature	Window	Bin	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	10	2	0.054	1.25	0.669	137	174	104	33	70
	2	1.6	2	0.051	0.95	0.654	137	132	88	49	44
CPDPM	1	8.8	3	0.061	1.19	0.645	137	167	98	39	69
	2	1.4	2	0.050	0.92	0.674	137	136	92	45	44

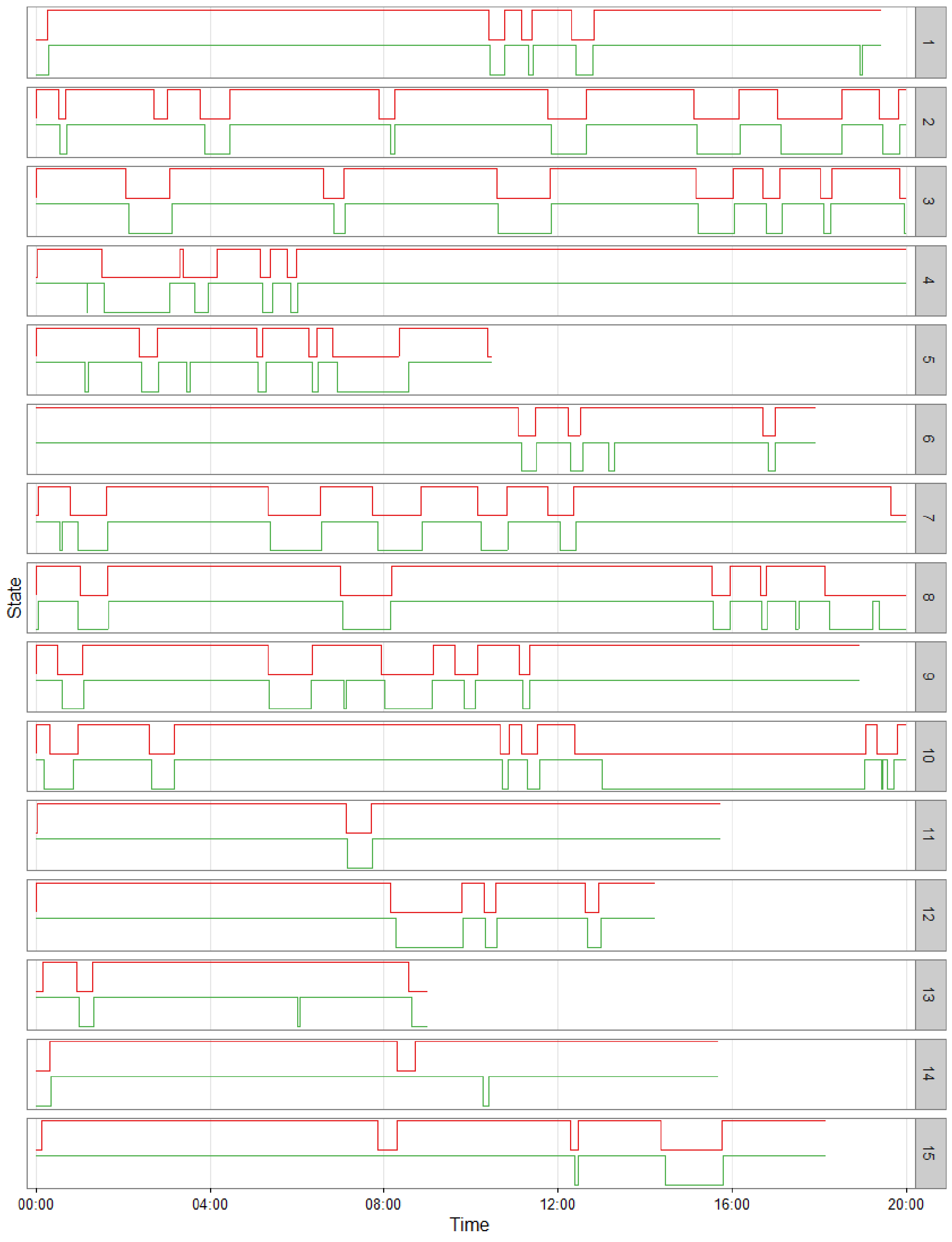


Fig. 35. Camry state estimation of test trips. Red: Prediction, Green: Observed.

5.2.3.3 Dataset 3: Driver 3 - Mazda 3 - Samsung Galaxy S3

Table 25. Validation of HMM Driver 3 - Mazda 3 - Samsung Galaxy S3.

Metric	Feature	Window	Bin	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.6	3	0.042	2.16	0.592	74	132	61	13	71
	2	0.8	2	0.040	1.02	0.747	74	76	56	18	20
CPDPM	1	4.4	2	0.046	1.11	0.688	74	86	55	19	31
	2	1.0	3	0.041	0.98	0.709	74	67	50	24	17

Table 26. Testing of HMM Driver 3 - Mazda 3 - Samsung Galaxy S3.

Metric	Feature	Window	Bin	ClassErr	CPDPM	F1Stop	Obs	Pred	True	Miss	False
Classif	1	1.6	3	0.054	2.10	0.530	160	270	114	46	156
	2	0.8	2	0.051	1.05	0.605	160	134	89	71	45
CPDPM	1	4.4	2	0.047	0.94	0.736	160	166	120	40	46
	2	1.0	3	0.055	1.10	0.599	160	117	83	77	34

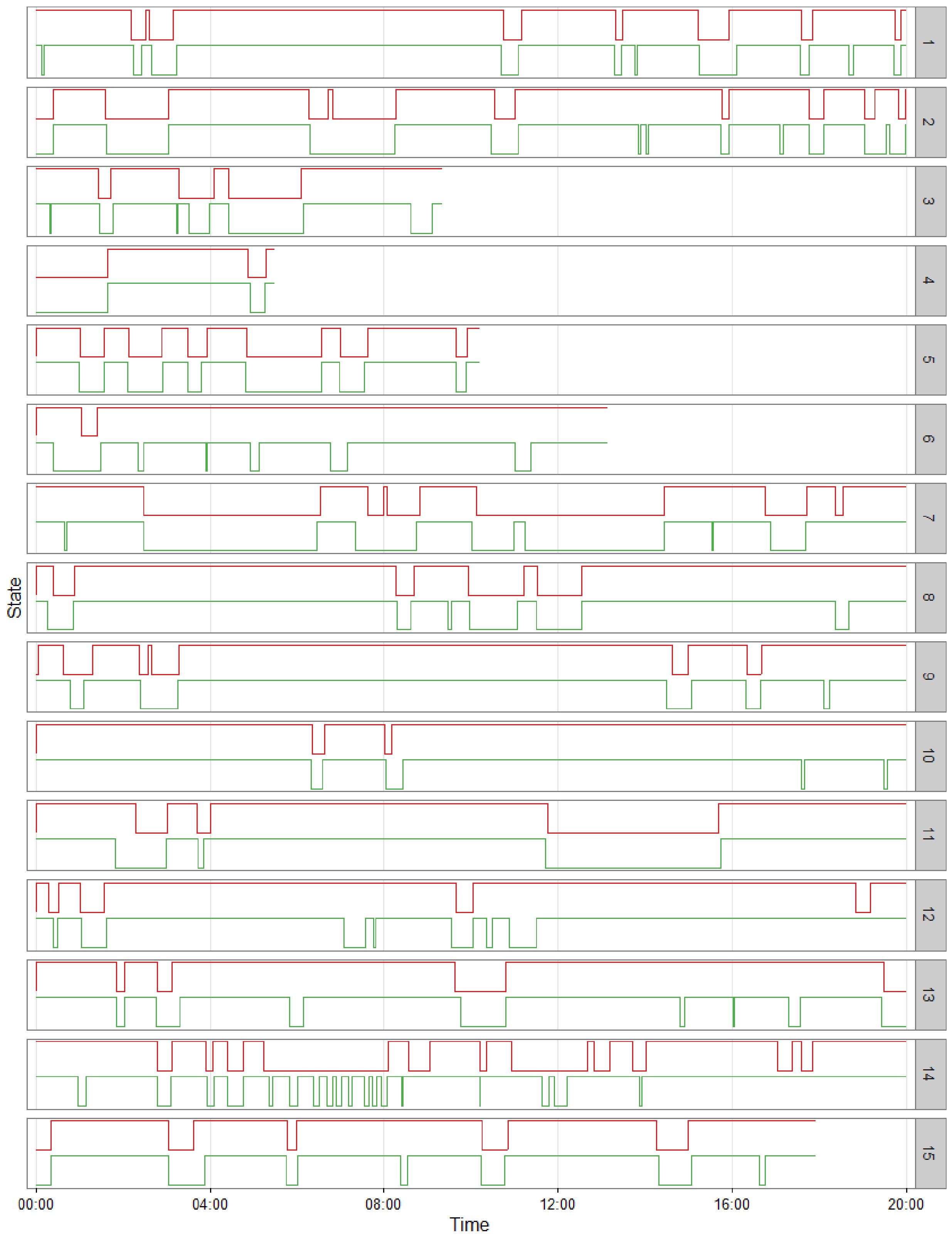


Fig. 36. Mazda state estimation of test trips. Red: Prediction, Green: Observed.

5.3 Analysis of the Results

The validation and testing results' F1 scores are combined and presented in Table 27 and Table 28, respectively.

Table 27. F1 scores of the validation trips

		RBF SVM			NARX NN			HMM		
Metric	Feature	Prius	Camry	Mazda	Prius	Camry	Mazda	Prius	Camry	Mazda
Classif	1	0.618	0.562	0.272	0.730	0.372	0.612	0.695	0.667	0.592
	2	0.647	0.219	0.265	0.891	0.729	0.196	0.896	0.630	0.747
CPDPM	1	0.741	0.571	0.436	0.768	0.776	0.727	0.803	0.718	0.688
	2	0.648	0.268	0.179	0.911	0.838	0.748	0.899	0.640	0.709

Table 28. F1 scores of the testing trips

		RBF SVM			NARX NN			HMM		
Metric	Feature	Prius	Camry	Mazda	Prius	Camry	Mazda	Prius	Camry	Mazda
Classif	1	0.593	0.429	0.300	0.759	0.281	0.544	0.683	0.669	0.530
	2	0.548	0.197	0.267	0.827	0.622	0.161	0.858	0.654	0.605
CPDPM	1	0.750	0.417	0.373	0.767	0.612	0.541	0.760	0.645	0.736
	2	0.543	0.194	0.154	0.799	0.683	0.556	0.857	0.674	0.599

Table 29. Window sizes of the best models, as found in different ML techniques.

		RBF SVM			NARX NN			HMM		
Metric	Feature	Prius	Camry	Mazda	Prius	Camry	Mazda	Prius	Camry	Mazda
Classif	1	2.0	3.0	1.0	1.0	1.0	1.0	0.6	10.0	1.6
	2	3.0	3.0	3.0	2.0	2.0	3.0	0.4	1.6	0.8
CPDPM	1	3.0	3.0	3.0	3.0	3.0	0.6	7.6	8.8	4.4
	2	3.0	0.2	0.2	3.0	1.0	0.2	0.4	1.4	1.0

Some of the important outcomes can be summarized as follows:

1. Overall, the best performing vehicle was the Toyota Prius, and the worst performing was the Mazda3. This might be due to the mechanical differences in the vehicles, and also due to the driver. Toyota Prius engines go into sleep mode after they have been stopped, while Mazda engines don't. The driver behavior is also important, since some drivers tend to do more rolling stops, rather than full stops, which can definitely affect the accelerometer, making it harder to detect the stop points.
2. The HMM outperformed the other methodologies, followed by NN. The SVM performed the worst. Considering how fast the training of HMM was, and the simplicity of the feature used, which was either the range (Feature 1), or the absolute difference of consecutive ranges (Feature 2), HMM became the model of choice for stop detection. The SVM and NN training times were too long, and were deemed to be not feasible, considering their outcomes.
3. Regarding the quality of features in change point detection, Feature 2 performed the best in HMM and in NN. Feature 2 used in NN consisted of a combination of features which were range, absolute difference of consecutive ranges, interquartile range, and standard deviation. In the RBF SVM method, Feature 1, which is range only, performed better.
4. In the majority of the cases, the introduced CPDPM was more capable of identifying the models that showed better performance. Since the best model selection is done during validation, based on classification error or the introduced Change Point Detection Performance metrics, the difference was more obvious in the validation results. Five out of the six times, using the HMM method, CPDPM was able to find the better models. However, in some cases during testing, a model selected by CPDPM performed worse

than the one selected by classification error, such as the NN for Prius. This might be due to some differences in the validation and testing trips, including random error.

5. When the window sizes of the best models were checked, as shown in Table 29, it can be seen that no pattern exists in the window size, i.e. sometimes large, and sometimes small windows performed better. Larger windows contain more information on the time series, while they are slow to react to changes. As the window size is increased, it gets harder to classify the small stops correctly, since the effect of the stop gets lost within the large window. Small windows tend to yield many stop predictions and have many false positives, while large windows tend to yield less stop predictions and have many missed observations.
6. The number of true estimates were determined by their proximity to the observed points. A threshold of 10 seconds was chosen as the boundary condition for being accepted as true estimate. This choice of threshold did not affect the calculation of the CPDPM method at all. The further away the points, the more they became penalized by using this method. The threshold was established purely to prepare the performance table, and to compute the F1 score. In order to test the effect of the threshold on the number of true estimates, threshold values ranging from 1 to 10 seconds were applied on the best performing HMM models of the three datasets. The results are shown in Table 30 through Table 32. It can be seen that for lower thresholds, it was hard to achieve high True numbers. As the threshold increased, there were more stop points that were classified as correct. In Fig. 37, the percentage of True, False, and Miss values were calculated by dividing these values with the corresponding number of observed stops.

Here, it is clear that, for the Prius and the Mazda after 5 seconds, there was not much gain in increasing the threshold.

Table 30. Prius HMM best model testing output for different threshold values.

Threshold (s)	Obs	Pred	True	Miss	False
1	251	258	123	128	135
2	251	258	188	63	70
3	251	258	198	53	60
4	251	258	209	42	49
5	251	258	215	36	43
6	251	258	216	35	42
7	251	258	218	33	40
8	251	258	218	33	40
9	251	258	218	33	40
10	251	258	218	33	40

Table 31. Camry HMM best model testing output for different threshold values.

Threshold (s)	Obs	Pred	True	Miss	False
1	137	136	5	132	131
2	137	136	9	128	127
3	137	136	20	117	116
4	137	136	44	93	92
5	137	136	54	83	82
6	137	136	63	74	73
7	137	136	77	60	59
8	137	136	83	54	53
9	137	136	87	50	49
10	137	136	92	45	44

Table 32. Mazda HMM best model testing output for different threshold values.

Threshold (s)	Obs	Pred	True	Miss	False
1	160	166	28	132	138
2	160	166	67	93	99
3	160	166	95	65	71
4	160	166	99	61	67
5	160	166	104	56	62
6	160	166	110	50	56
7	160	166	113	47	53
8	160	166	114	46	52
9	160	166	118	42	48
10	160	166	120	40	46

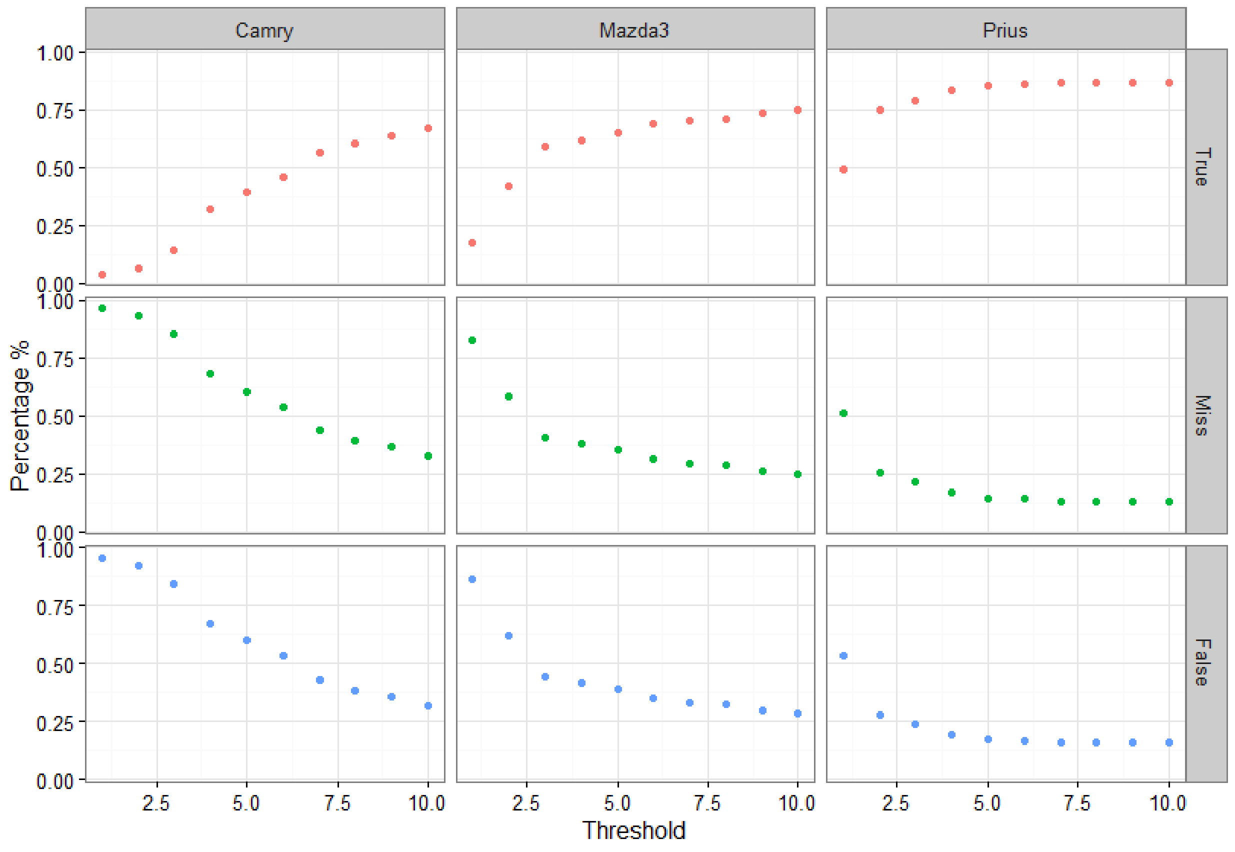


Fig. 37. Percentage True, False, and Miss values w.r.t. the threshold and the vehicle type.

CHAPTER 6

SPEED ESTIMATION

The kinetic theory tells that the integration of acceleration gives the speed of a vehicle. Thus, the integration of the acceleration values collected with the smartphone in the direction of motion would theoretically yield the speed. However, speed estimation directly by integration of accelerometer data is not possible, since the accelerometer data in the direction of motion is not pure acceleration, but involves white noise, phone sensor bias, vibration, gravity component, and other effects. These get integrated together with the motion data and produce inaccurate results. A calibration method that can adjust the speed at certain points is needed. The stop and start point detection algorithm provides the necessary calibration points.

6.1 Algorithm for Estimating Speed

In this section, the process for estimating speed will be explained. One of the trips will be used to illustrate the case. The route taken during the trip and the vehicle's speed are shown in Fig. 38. The trip was approximately 14 miles long, and took about 22 minutes. The accelerometer sensor has three axes. In order to be able to estimate speed, the accelerometer measurements taken in the direction of motion is needed. In practice, the true orientation of the phone with respect to the vehicle will not be known. However, here it is assumed the y-axis of the phone is oriented along the direction of the movement of the vehicle. If the phone is positioned in the vehicle in a random orientation, orientation correction methods are needed. The raw accelerometer values in the three axes logged from the smartphone, the magnitude of the acceleration 3D vector, and the GPS and OBD speeds of the vehicle are shown in Fig. 39. First, the kinematic equation shown in Eq. (87) is used to estimate speed.

$$V_f = V_i + a\Delta t \tag{87}$$

Here,

V_f	Final velocity,
V_i	Initial velocity,
a	Acceleration,
Δt	Time interval.

The frequency of measurement is 10Hz. Thus, the time interval will be 0.1 seconds. Initial velocity is taken as the first speed value of the GPS, and then the vehicle speed at each instance is estimated recursively, based on Eq. (87).

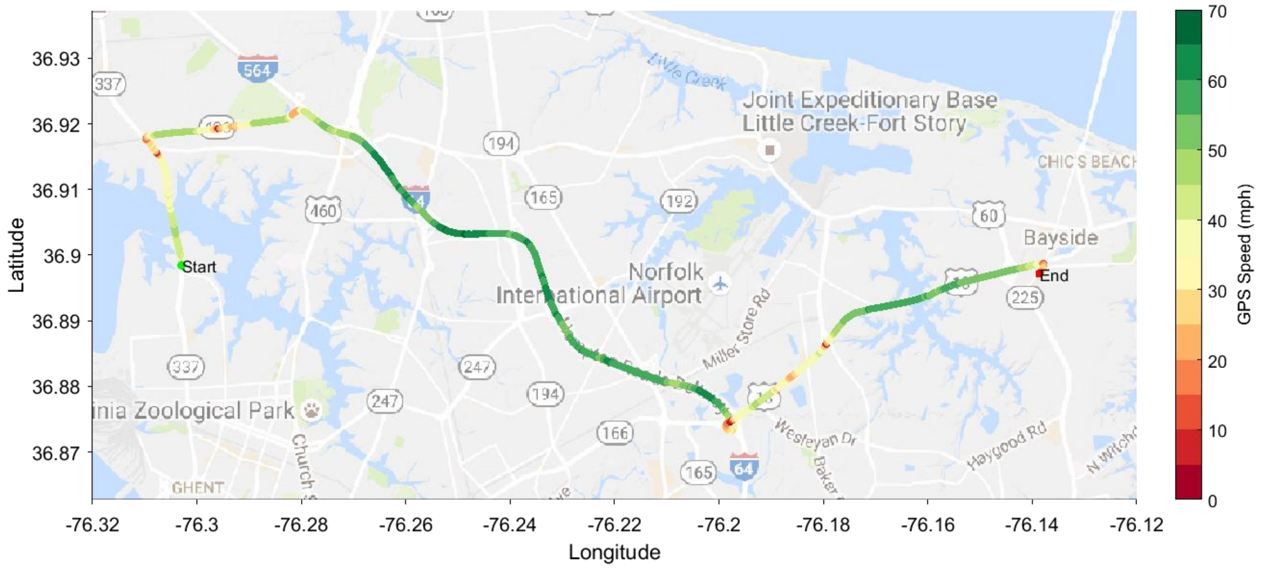


Fig. 38. The route and speed of a Toyota Camry driven from Norfolk to Virginia Beach.

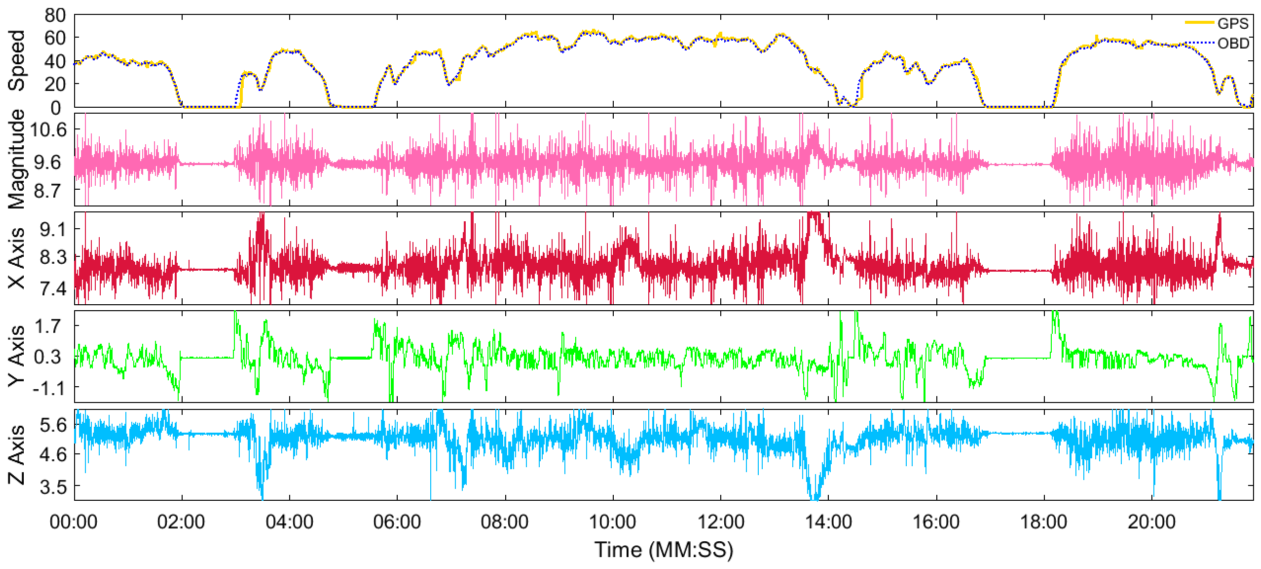


Fig. 39. Raw accelerometer, magnitude of acceleration, and speed of GPS and OBD.

Normally, once this step is completed, one would expect that the speed estimate would be more or less correct, based on the kinematic equation above. However, as mentioned before, because the phone is not oriented perfectly in the direction of motion, and other noise factors get accumulated as well during the integration process, the result is an upwardly (or downwardly) sloped monotonically increasing curve, as can be seen in Fig. 40. As is evident, the monotonic increase can be identified by a slope within each motion and standstill segment. A segment is defined as the region between a stop and start point (standstill), or between a start and stop point (motion), which are shown in Fig. 41. Here, the red vertical line signifies the stopping of the vehicle, while the green vertical line represents the start point. These points are referred to as change points. The state of each point *i* of a trip is denoted as a set *Q* containing values of ones and zeroes:

$$Q = \{1,0,0, \dots ,1,1,1,1, \dots ,1\} \tag{88}$$

The zeros (0) denote vehicle standstill at each instance, and the ones (1) denote the vehicle being in motion. To detect the change points (the stop and start points of a vehicle), the algorithm shown below is applied:

$$Q_i - Q_{i-1} \in \{0,1,-1\} \tag{89}$$

The difference between point *i* and point i-1 can only have three distinct values. “0” denotes no change in the state of the vehicle. “1” denotes that the vehicle was in standstill and started moving. The index of this point is stored in the Motion Start (*M*) set. A value of “-1” denotes that the vehicle was in motion and has stopped at this instance. The index of this point is stored in the Stop (*S*) set.

$$M = \{i|Q_i - Q_{i-1} = 1\}$$

$$S = \{i|Q_i - Q_{i-1} = -1\} \tag{90}$$

$$C = S \cup M$$

Where $i = 1,2,3, \dots, |Q|$.

Thus, *M* is the set of indices when vehicle starts to move from a standstill, and *S* is the set of indices when the vehicle stops from being in motion. *C* is the sorted union of the stop and start change points in the trip, where C_k would represent one of the change points’ index. There are *K + 1* change points, including the very first and end points of a trip, and there are *K* segments.

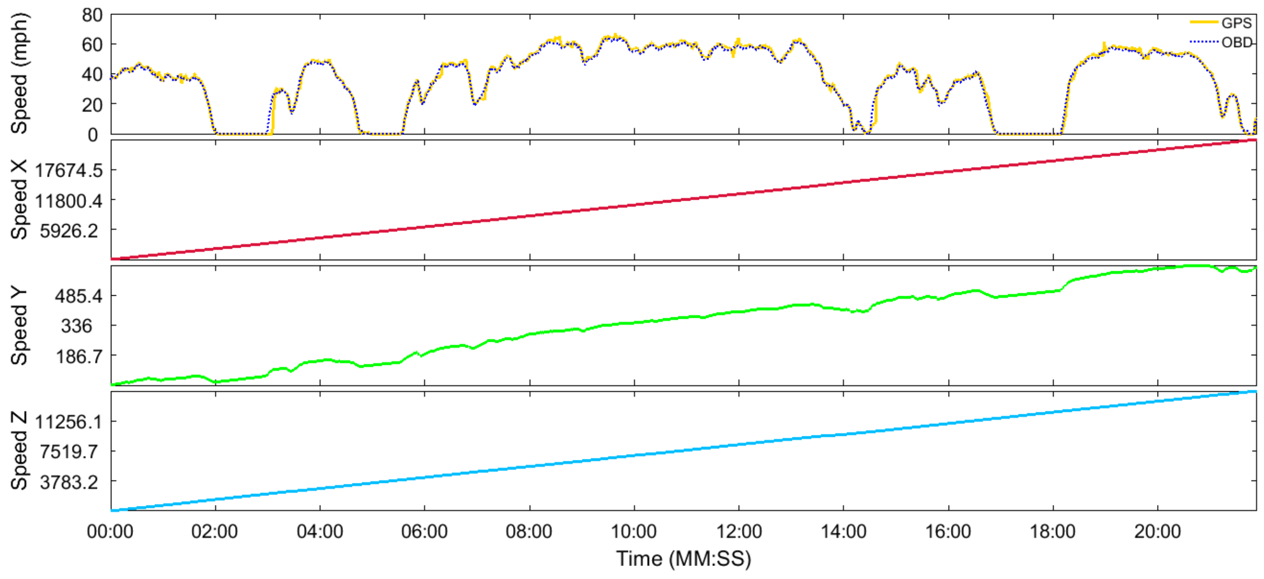


Fig. 40. Speed estimations for each axis using the kinematic equation of $V_f = V_i + a\Delta t$.

Once the state detection phase is complete and the stop and start points are detected, the slope of each segment is computed, except for the first and last segments, unless these segments are a standstill. This is because, if the first and last segments start with motion, the calculated slope will be wrong, as is evident in Fig. 41. The slope for each segment is calculated as the difference in speed between the last and first points of a segment, divided by the number of points in the segment. Since the change points represent the first point of each segment, the index of the last point of a segment is one less than the index of the next change point.

$$m_k = \big(V(C_{k+1} - 1) - V(C_k)\big)/(C_{k+1} - C_k) \tag{91}$$

Where,

m_k	Slope of segment *k*
$V(C_k)$	Speed at the index of the state change point C_k
C_k	The index of the change point *k* in the set *C*

Once the slope within each region is computed, the median is taken and the median $\widetilde{m}$ is used in the rest of the computations. If so desired, the individual slope values can be used in each segment as well. However, this might be a little noisy. The stop and start points are used to calibrate the speed estimation at standstill segments, where the speed is set to zero.

$$V = \{0 | Q_i = 0, for\ i = 1,2,3, \ldots, |Q|\} \tag{92}$$

The speed at the beginning of each motion segment and the slope will be used to calibrate the estimated speed in the motion segments. The speed of each point is subtracted by the speed of the segment's first point and the product of the slope and the number of points from the beginning of the segment. The remainder is the calibrated speed estimation, which is the top part of the black vertical line shown in Fig. 41. Here, by getting rid of the $V_{Initial}$ and V_{Slope} portions from V_i, the actual speed is left, which is denoted by $V_{Calibrated}$. In a sense, each point in the motion segment is pulled down to the expected speed level.

$$\begin{gathered} V = \left\{V_i - \widetilde{m}(i - C_k) - V(C_k) \,\middle|\, Q_i = 1, k > 1\right\} \\ V = \left\{V_i - \widetilde{m}(i - C_k) \,\middle|\, Q_1 = 1, k = 1 \right\} \end{gathered} \tag{93}$$

Where $i = 1,2,3, \ldots, |Q|$.

The calibration phase finalizes the speed estimation process. The calibrated speed estimation for each axis is shown in Fig. 42. Here, it becomes obvious that the motion direction of the vehicle aligns mostly with the Y axis of the phone, as the speed estimation obtained on this axis is the best among the three. It can be seen that the performance of the speed estimation is quite good and that it mimicked the actual speed very closely.

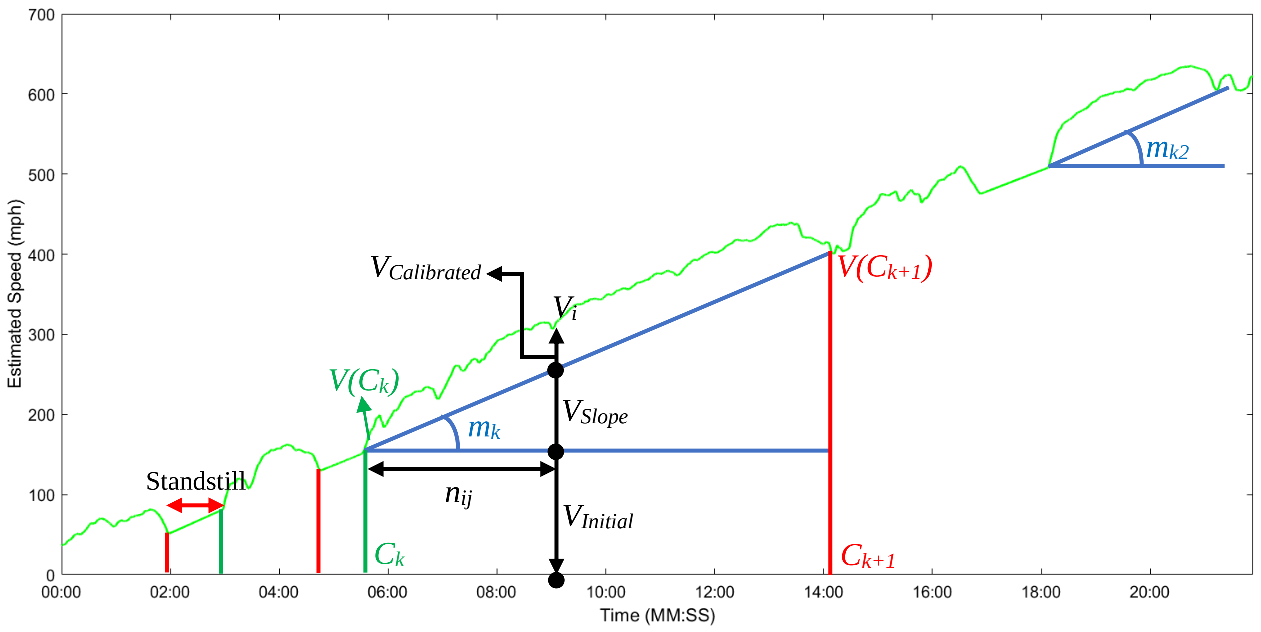


Fig. 41. Calculating the slope, and finding the beginning and ending speeds of segments.

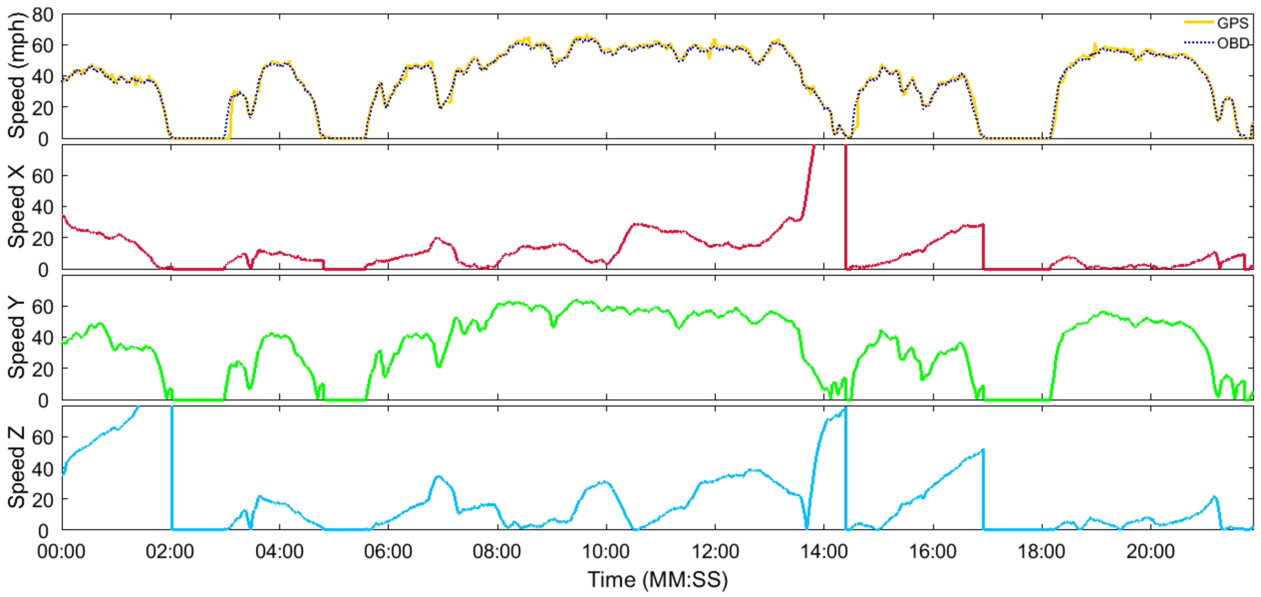


Fig. 42. Calibrated speed estimation on each axis using the observed change points.

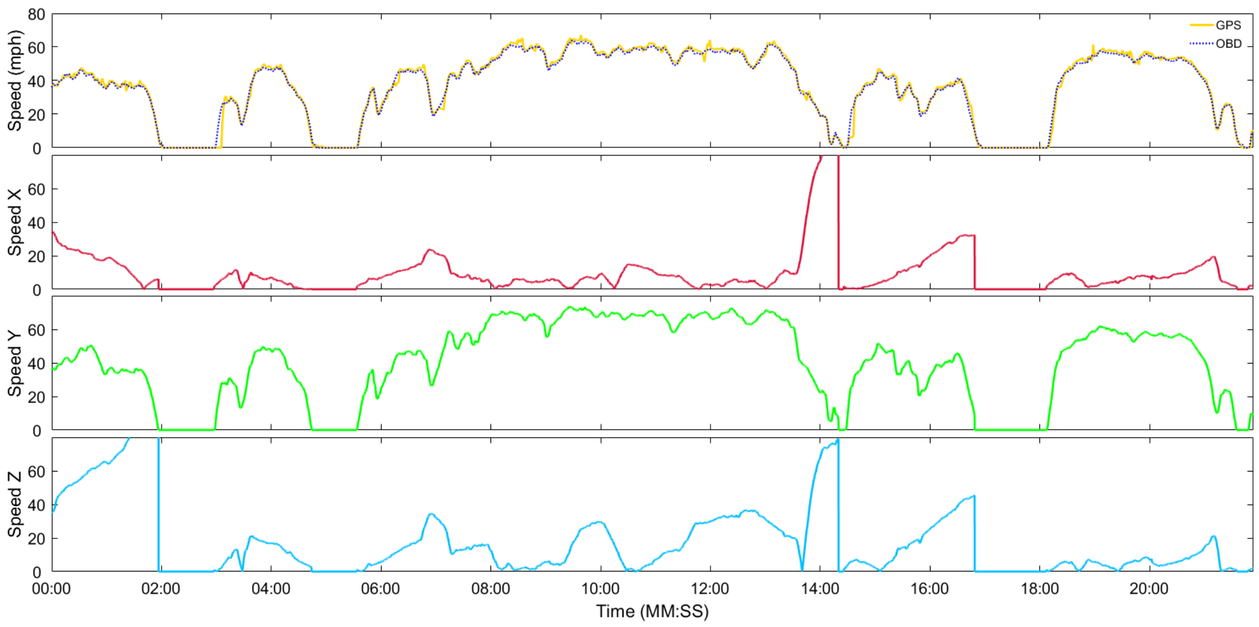


Fig. 43. Calibrated speed estimation on each axis using the predicted change points.

An important issue that needs to be mentioned is that, until now, the change points from observed set are used for calibration. The high accuracy of speed estimation using observed points proves that the process of calibration is a feasible method. However, the real performance of the overall change point detection and speed estimation can be obtained by testing with predicted change points. The estimated speed obtained by using the predicted change points is shown in Fig. 43. Visually, it can be deduced that the method still performs very well, with some overestimation between the minutes 6 and 14. Here, the vehicle is travelling on the highway. Some other dynamics might be causing this overestimation. The RMSE of estimated speed with respect to GPS speed using the observed and predicted points are shown in Table 33. As can be seen, both achieve good accuracy, with approx. 6 RMSE. Using the predicted points is not very different than using the observed ones, and only degrades the accuracy by 1.2 RMSE.

The speed estimation algorithm is applied to all the trips in the testing set. The mean RMSE of these trips is provided on the right of Table 33. The speed estimation using the predicted points performed very well, with slightly more error than using the observed points. The model could be further improved by heuristic methods, such as increasing or decreasing the estimated speed by a certain amount if it under or overestimates consistently in certain type of segments. One such approach could be defined for segments over highways, which have higher speeds for longer durations.

Table 33. The RMSE of estimated speed w.r.t. GPS speed.

	Single Trip RMSE	Testing Set Mean RMSE
Observed Change Points	5.49	9.78
Predicted Change Points	6.68	11.16

CHAPTER 7

CONCLUSIONS

This dissertation presents a potential approach to detect whether a vehicle (with an onboard smartphone) is in motion or stationary, based on the data collected by the accelerometer within the smartphone. Due to the inherent noise in the accelerometer data and the variation in driving behavior, detecting vehicle stops was found to be nontrivial. Despite these complexities, this dissertation demonstrates that detecting the stops is feasible, although further enhancements are needed to improve on accuracy and on minimizing the missing true points and the number of false alarms. Based on the testing results presented, on average, 80% of the stops made by the vehicles were detected by the models. The best average was 90% with Toyota Prius, and the worst average was 75% with Mazda. It should be noted that some trips had many stop-go cases where the vehicle did not come to an exact halt, but rather to a rolling stop. These cases were hard to classify. Another issue is very small stops of one or two seconds. These might occur at red lights, where it just turns green as the car is about to come to a halt, and the car starts accelerating again. Still, many of these cases were correctly identified as a stop, as can be seen in Fig. 34-Fig. 36. When the vehicle came to a total halt for more than a few seconds, it was identified correctly. The False Positives and Misses could potentially be eliminated with further fine-tuning of the model parameters through an optimization framework.

The data collection was done by students and faculty who used an in-house developed Android phone app. Later, the data was uploaded to a database, which was used for model development and testing. The data collection was done in a real environment which involved streets, arterials, and highways. The results presented in this dissertation focused on three different drivers who had three different vehicles, and three different phones. This way, the robustness of the change point detection model was tested. The results show that the model is robust and is able to detect change points with high accuracy, provided that some changes were made to the parameters. That’s why vehicle-phone specific models were developed, since each vehicle and each phone had its own unique attributes. For example, there were significant differences between a Toyota Prius, which goes to silent mode when the vehicle is stopped for some duration, and another car which doesn’t have this property. Phones also differed in collecting data, since some were more sensitive and precise. Based on these facts, a sort of distributed machine learning approach was taken, in which vehicle-phone specific models were developed and tested.

A lot of different machine learning models, features, and window sizes were experimented with to find the best model with right parameters. Logistic regression, linear discriminant analysis, K nearest neighbors, classification trees, support vector machines, static neural networks, recurrent neural networks, hidden Markov models are a few. There were several parameters to be optimized in all the models, and finding the optimum combination was nontrivial. Based on the experiment results, the SVM, recurrent NN, and the HMM were found to be the best, and the testing results using these models were presented. HMM was shown to outperform all of the methods in terms of its higher accuracy detecting the change points, its fewer false positives, and its higher precision. Although the other machine learning models were fed with several different features, HMM excelled at detecting the change points in the time series data, by using a feature as simple as the range or the absolute difference of consecutive ranges.

The Change Point Detection Performance Metric (CPDPM) proved to be very useful in assessing the models where the goal was to find the change points in a time series data with high accuracy and precision. Both theoretically and empirically, CPDPM was proven to be a better performance metric compared to the regular classification error, which is the common method used in classification tasks. For the classification error metric, what matters is that the predicted points are classified correctly. There is no notion of change point, which entails time information, such as the closeness of predicted points to the actual observed points. By incorporating these, and by making use of the so-called “error function”, a novel performance metric was developed. As far as the author’s knowledge goes, there is no such performance metric currently in the literature, and it is believed that the proposed method will be of great benefit in similar cases, where the goal is to find change points or specific instances within time series data.

Speed estimation directly by integration of accelerometer data was not possible with data collected from a smartphone, since the accelerometer data was not pure acceleration in motion direction, but involved white noise, phone sensor bias, vibration, gravity component, and other effects. These got integrated together with the actual data and produced inaccurate results. A calibration method that can adjust the speed at certain points was needed. The predicted stop and start points, and the slope obtained by integrating acceleration data, were used to calibrate the estimated speed. The proposed method for estimating speed proved to be very successful, and can be used as an alternative to or as improvement to GPS based systems, which sometimes cannot receive signal due to urban canyon and other effects.

7.1 Potential Applications of Vehicle Motion Detection

The proposed methods of vehicle movement detection and estimating speed have the potential to be used in several fields. Some of the possible applications are defined in the following sections.

7.1.1 GPS Speed and Localization Correction

Having a GPS sensor in the smartphones is of great benefit, both to the consumer and to the researchers who make use of it doing research. The GPS sensor will provide both speed and location information. However, GPS signals can be highly misleading at times such as in urban canyons, tunnels, and adverse weather conditions. The speed estimation obtained from using the accelerometer data can be used to correct the GPS speed by means of methods such as Kalman filters. Also, when the vehicle stops, the GPS is almost never able to pinpoint the location exactly, but rather lingers around sometimes in rather large radii. This sometimes causes the speed to never show as 0 mph but rather a hanging speed of around as high as 10 mph. Some of these problems are shown in Appendix B. Detecting that the vehicle has stopped will help correct the GPS unit to show 0 mph correctly, and will also improve on localization.

7.1.2 Fuel consumption, emission estimation

The model developed will provide the fuel consumption estimation method with two key inputs: one is speed and the other is vehicle idle and motion states. Assuming that the vehicle type, model, and year are known, estimating the fuel consumption and the gas emission is a matter of plugging in the variables, since the formulas and coefficients related to fuel consumption and gas emissions have already been published, over many years of research.

7.1.3 Transportation mode recognition

Each vehicle behaves differently and has a unique footprint while moving. Sensors such as accelerometer and gyroscopes have certain characteristics for each transportation type, as is evident in Fig. 44. By exploiting these characteristics, and by developing a model that can learn the patterns hidden in the sensor data, the transportation mode used can be detected. Instead of using GPS speed, as is done in the literature, the speed estimated from sensors can serve as an input into the mode recognition model.

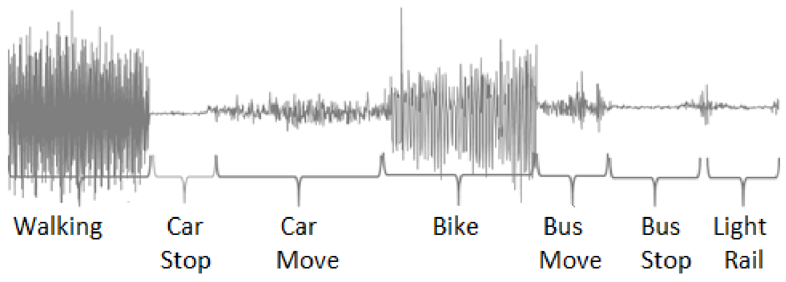


Fig. 44. Acceleration data collected by a smartphone for each transportation type.

7.1.4 Queue length, Average speed, Travel time estimation

Several properties related to traffic, such as queue length, average speed, average travel time are called "measures of effectiveness." According to the information provided by FHWA, speed and travel time measuring can be achieved by various techniques and can be grouped into three categories:

1- Spot speed measurement techniques, which measure vehicle speeds only for a given point of geography or a given point in time.

2- Vehicle tracing techniques, which measure vehicle travel times only for a select portion of all trips.

3- Trip maker tracking techniques, which are similar to vehicle tracing techniques but measure traveler trip times rather than vehicle trip times.

However, as can be understood, these techniques are limited both in scope and in time, and they also require special equipment, personnel, and other means in order to achieve a certain level of accuracy, which might be even questionable in conditions such as adverse weather, not proper materials, and the probability of missing information. It goes without saying that all of these efforts involve expenditure and budget.

The signal controllers that are already present at almost all the intersections could be equipped with transponders that could send info to and receive info from smartphones via Bluetooth or some other technology. These systems are referred to as Vehicle to Infrastructure (V2I) technology in the industry. The main difference here is the use of the smartphone, instead of the vehicle, as the source. Establishing such a technology would allow the transmission of information from smartphones to the infrastructure. For example, vehicle stop and start times, standstill duration at an intersection, distance to controller (queue length), or in case the signal is green, vehicle estimated speed, could all be transmitted to the transponder. Queue length, the travel time between two points, the number of stops on a corridor, the delays at a certain intersection, the total delay over a certain segment, the average vehicle speed, and more could all be obtained by a simple and cost efficient system. These data have huge importance in designing intersections, optimizing green times, directing traffic to alternative routes, and the like. In the long term, they can serve as input for infrastructure investments such as building new roads, building on and off ramps on highways, introducing roundabouts, adding new lanes to existing roads, etc. The ubiquitous use of smartphones could be utilized to develop smart cities.

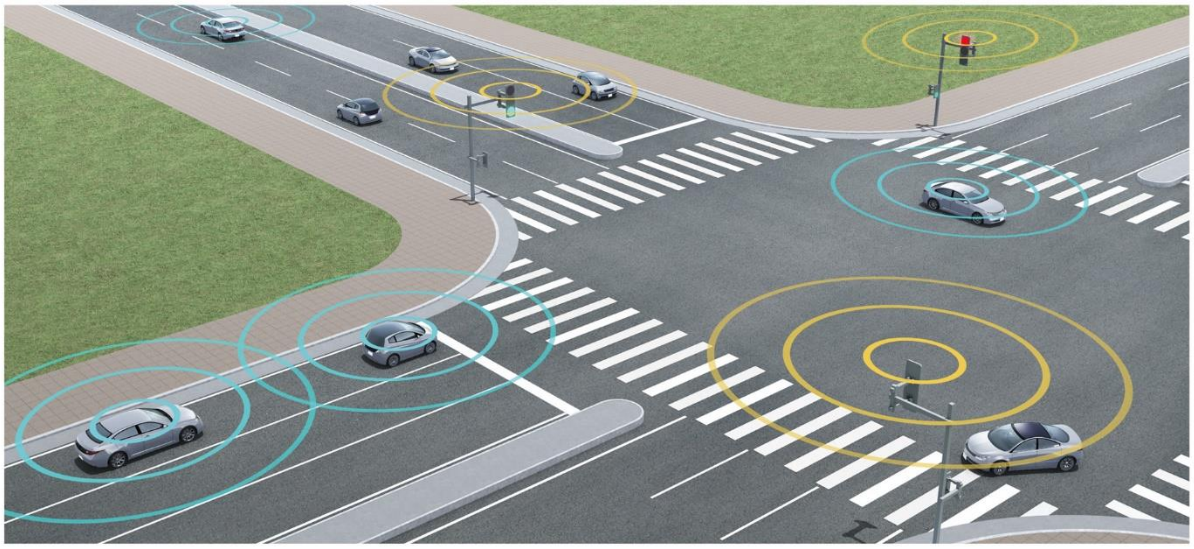


Fig. 45. Ubiquitous use of smartphones could be utilized improve traffic.

REFERENCES

[1] W. Ballantyne, G. B. Turetzky, G. Slimak, and J. Shewfelt, "Achieving low energy-per-fix in cell phone," GPS World, vol. 17, pp. 24-32, 2006.

[2] D. Raskovic and D. Giessel, "Battery-aware embedded GPS receiver node," in 4th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, MobiQuitous 2007, August 6, 2007 - August 10, 2007, Philadelphia, PA, United states, 2007.

[3] F. Simjee and P. H. Chou, "Accurate battery lifetime estimation using high-frequency power profile emulation," in ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 8-10 Aug. 2005, Piscataway, NJ, USA, 2005, pp. 307-10.

[4] B. W. Parkinson, "GPS error analysis," in Global Positioning System: Theory and applications. vol. I, ed, 1996, pp. 469-483.

[5] N. M. Drawil, H. M. Amar, and O. A. Basir, "GPS Localization Accuracy Classification: A Context-Based Approach," IEEE Transactions on Intelligent Transportation Systems, vol. 14, pp. 262-273, 2013.

[6] N. R. Center. (2012, Jan 15, 2014). New mobile majority: a look at smartphone owners in the U.S. Available: http://blog.nielsen.com/nielsenwire/online_mobile/who-owns-smartphones-in-the-us/

[7] N. R. Center. (2012, Jan 15, 2014). Two Thirds of New Mobile Buyers Now Opting for Smartphones. Available: http://www.nielsen.com/us/en/newswire/2012/two-thirds-of-new-mobile-buyers-now-opting-for-smartphones.html

[8] C. S. Consulting. (2013, Jan 15, 2014). US Wireless Market Update Q1 2013. Available: http://www.chetansharma.com/blog/2013/06/20/us-wireless-market-update-q1-2013/

[9] N. R. Center. (2013, Jan 15, 2014). Consumer Electronics Ownership Blasts Off in 2013. Available: http://www.nielsen.com/us/en/newswire/2013/consumer-electronics-ownership-blasts-off-in-2013.html

[10] P. R. Center. (2013, Jan 15, 2014). Smartphone Ownership 2013. Available: http://pewinternet.org/Reports/2013/Smartphone-Ownership-2013.aspx

[11] P. R. Center. (2012, Jan 15, 2014). Smartphone Ownership Update. Available: http://www.pewinternet.org/Reports/2012/Smartphone-Update-Sept-2012/Findings.aspx

[12] eMarketer. (2014, March 05, 2013). Forecast: number of smartphone users in the U.S. 2010-2016.

[13] P. R. Center. (2013, Jan 31, 2014). Who's winning the U.S. smartphone market? Available: http://www.nielsen.com/us/en/newswire/2013/whos-winning-the-u-s-smartphone-market-.html

[14] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in 6th ACM Conference on Embedded Networked Sensor Systems, SenSys 2008, November 5, 2008 - November 7, 2008, Raleigh, NC, United states, 2008, pp. 323-336.

[15] T. Klosowski. (2013, August 2016). The Best Apps that Use Your Phone's Boring Features in Clever Ways. Available: http://lifehacker.com/the-best-apps-that-use-your-phones-boring-features-in-710672640

[16] (2016, August 18, 2016). 10 Free Accelerometer Apps for iPhone and iPod Touch. Available: https://turbofuture.com/cell-phones/10-Best-Accelerometer-Apps

[17] M. J. Mathie, A. C. F. Coster, N. H. Lovell, and B. G. Celler, "Accelerometry: providing an integrated, practical method for long-term, ambulatory monitoring of human movement," Physiological Measurement, vol. 25, pp. 1-20, 04/ 2004.

[18] G. Hansson, P. Asterland, N. G. Holmer, and S. Skerfving, "Validity and reliability of triaxial accelerometers for inclinometry in posture analysis," Medical and Biological Engineering and Computing, vol. 39, pp. 405-413, 2001.

[19] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," SIGKDD Explor. Newsl., vol. 12, pp. 74-82, 2011.

[20] L. Bao and S. Intille, "Activity Recognition from User-Annotated Acceleration Data," in Pervasive Computing. vol. 3001, A. Ferscha and F. Mattern, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 1-17.

[21] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman, "Activity recognition from accelerometer data," in 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference, AAAI-05/IAAI-05, July 9, 2005 - July 13, 2005, Pittsburgh, PA, United states, 2005, pp. 1541-1546.

[22] M. Oliver, H. Badland, S. Mavoa, M. J. Duncan, and S. Duncan, "Combining GPS, GIS, and Accelerometry: Methodological Issues in the Assessment of Location and Intensity of Travel Behaviors. ," Journal of Physical Activity & Health vol. 7, pp. 102-108, 2010.

[23] A. R. Cooper, A. S. Page, B. W. Wheeler, P. Griew, L. Davis, M. Hillsdon, et al., "Mapping the Walk to School Using Accelerometry Combined with a Global Positioning System," American Journal of Preventive Medicine, vol. 38, pp. 178-183, 2// 2010.

[24] P. J. Troped, M. S. Oliveira, C. E. Matthews, E. K. Cromley, S. J. Melly, and B. A. Craig, "Prediction of Activity Mode with Global Positioning System and Accelerometer Data," Prediction of Activity Mode with Global Positioning System and Accelerometer Data, vol. 40, p. 7, 2008.

[25] J. L. Wolf, M. G. S. Oliveira, P. Troped, C. E. Mathews, E. K. Cromley, and S. J. Melly, "Mode and Activity Identification Using GPS and Accelerometer Data," in Transportation Research Board 85th Annual Meeting, Washington DC, United States, 2006.

[26] J. Wang, R. Chen, X. Sun, M. F. H. She, and Y. Wu, "Recognizing Human Daily Activities From Accelerometer Signal," Procedia Engineering, vol. 15, pp. 1780-1786, // 2011.

[27] M. Ermes, J. Parkka, J. Mantyjarvi, and I. Korhonen, "Detection of Daily Activities and Sports With Wearable Sensors in Controlled and Uncontrolled Conditions," Information Technology in Biomedicine, IEEE Transactions on, vol. 12, pp. 20-26, 2008.

[28] J. Lester, P. Hurvitz, R. Chaudhri, C. Hartung, G. Borriello, and "MobileSense - Sensing Modes of Transportation in Studies of the Built Environment," presented at the International Workshop on Urban, Community, and Social Applications of Networked Sensing Systems - UrbanSense08, 2008.

[29] WISDM (Wireless Sensor Data Mining) Available: http://www.cis.fordham.edu/wisdm/ ; http://storm.cis.fordham.edu/~gweiss/wisdm/android.html

[30] C. Syuan-Yi, H. Chung-Ming, L. Shih Yang, and T. Lai, "Activity recognition for triggering cooperative networking among on-vehicle smart devices," in ITS Telecommunications (ITST), 2013 13th International Conference on, 2013, pp. 80-84.

[31] T. Feng and H. J. P. Timmermans, "Transportation mode recognition using GPS and accelerometer data," Transportation Research Part C: Emerging Technologies, vol. 37, pp. 118-130, 12// 2013.

[32] V. Manzoni, D. Manilo, K. Kloeckl, and C. Ratti, "Transportation mode identification and real-time CO2 emission estimation using smartphones."

[33] C. F. Ratti and K. Kloeckl, "CO2GO: Transportation Mode Identification and Real-Time CO2 Emissions Estimation Using Mobile Phone Sensors," USA Patent, 2011.

[34] T. M. Mitchell, Machine Learning, 1 ed. ed.: McGraw-Hill, Mar. 1997.

[35] P. Langley, Elements of Machine Learning. San Francisco, 1996.

[36] V. Vapnik, Estimation of Dependences Based on Empirical Data [in Russian]. Nauka, Moscow, 1979.

[37] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273-297, 1995/09/01 1995.

[38] C. J. C. BURGES, "A Tutorial on Support Vector Machines for Pattern Recognition," Data Mining and Knowledge Discovery, vol. 2, pp. 121-167, 1998.

[39] Y. Zhang and Y. Xie, "Travel mode choice modeling with support vector machines," Transportation Research Record: Journal of the Transportation Research Board, pp. 141-150, 2008.

[40] H. Chih-Wei and L. Chih-Jen, "A comparison of methods for multiclass support vector machines," Neural Networks, IEEE Transactions on, vol. 13, pp. 415-425, 2002.

[41] C. M. Bishop, "Pattern recognition," Machine Learning, vol. 128, 2006.

[42] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in AAAI, 1992, pp. 129-134.

[43] MathWorks. (August 2016). Design Time Series NARX Feedback Neural Networks. Available: http://www.mathworks.com/help/nnet/ug/design-time-series-narx-feedback-neural-networks.html

APPENDICES

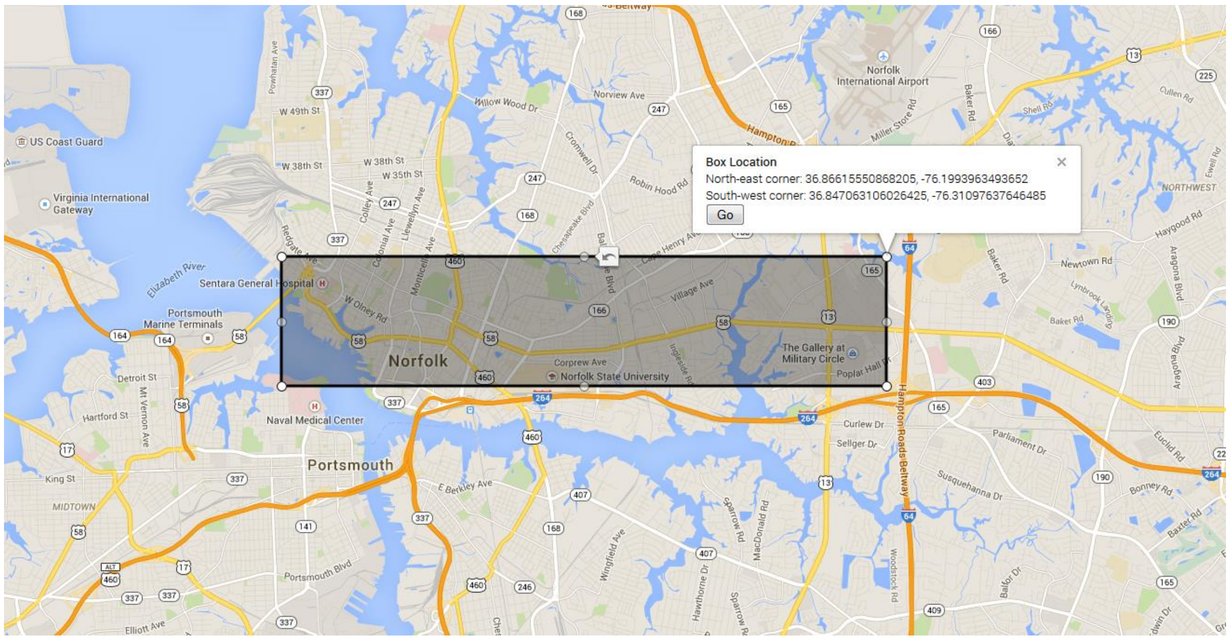
APPENDIX A: PREVIOUS METHODOLOGY

In this section, some of the research which involved a hybrid model of SVM-HMM and dynamic neural networks is presented. The hybrid model was abandoned later on, since it was found that HMM alone performed much better and was more reliable over a large variety of trips. The success of SVM-HMM in this section lies in the fact of having rather clean data, which is not that common in the real world.

Table 34. Saturn Ion 2007 Trip Info.

Training Data					**Testing Data**				
Duration (m)	**# Stops**	**Max Speed**	**Trip Distance**	**Motion %**	**Duration (m)**	**# Stops**	**Max Speed**	**Trip Distance**	**Motion %**
2.31	1	45.8	1.27	97.6	5.36	4	49.6	2.01	67.6
3.1	1	40.8	1.3	82.8	2.06	1	47.9	0.47	43.5
4.9	3	45.2	1.98	74.9	4.9	3	48.3	1.74	72.4
5.98	4	43.4	2.02	65.2	5	3	42.1	1.2	61.4
6.55	6	48.3	1.98	68.5	4.01	1	45.2	1.8	83.1
4.95	3	45.2	1.64	77	5.8	3	45.2	1.79	66.9
5.7	4	44.0	1.68	63.7	5.03	3	44.6	1.77	70.7

The summary of training and testing trips that are used in this section are provided in Table 34, which includes the trip duration in minutes, the number of times the vehicle stopped in each trip, the max speed in mph, the trip distance in miles, and the percentage of time the vehicle was in motion. These trips were from driving a Saturn Ion 2007 on arterials and city streets in Norfolk VA, as shown in Fig. 46. The device used to collect data was a Samsung Galaxy S3.


Box Location

North-east corner: 36.86615550868205, -76.1993963493652

South-west corner: 36.847063106026425, -76.31097637646485

Fig. 46. Geographic boundaries of the trip data collected in Norfolk, VA

Linear Support Vector Machines

Linear SVMs try to find a linear boundary between the classes without any transformation of the input data. It is the simplest form of SVM, and the only parameter to find the optimum value is the box constraint. Soft margin SVM with a slack variable ξ_i, and a penalty parameter *C* need to be optimized. The *C* value acts not only as a penalty on misclassified points, but also as the boundary value for the Lagrange multipliers. Thus, neither too small nor too large a *C* value will work well. An optimum value of *C* can be found through experimentation.

The first experiments were carried out with $C = 1$ using different features. The MATLAB built-in function "fitcsvm" was utilized with default values for the parameters, and the *C* value was set to 1. To solve the quadratic constrained optimization problem, the Sequential Minimal Optimization (SMO) algorithm was used.

The classification error is calculated as:

$$\frac{\sum_{i=1}^{N}(\delta_i|y_i,\hat{y}_i)}{N} * 100 \qquad (94)$$

Where,

δ is an indicator function which equals to 1 if $y_i = \hat{y}_i$, 0 otherwise.

N is the total number of points of all 7 test trips.

The CPDPM columns were calculated by summing CPDPM values of the seven test trips and then dividing by the corresponding number of points. For example, the CPDPM stop points were obtained by summing the CPDPM values of stop points and then dividing by total number of estimated stop points of the seven test trips.

The total trip time of the test trips was 32.3 minutes, with standstill totaling 10.4 minutes, and motion totaling 21.9 minutes. Features were extracted from the magnitude of accelerometer data, and were calculated over non-overlapping sliding windows of one second. The features experimented with were as follows:

Table 35. Features used and their explanation.

Feature	Explanation
Mean	The mean of the points in the window.
StDev	The standard deviation of the points in the window.
Range	The range of the points in the window.
Max	The maximum value in the window.
Median	The median value in the window.
P.25.75	The 25th and 75th percentiles in the window.
P.10.90	The 10th and 90th percentiles in the window.
P.25.50.75	The 25th, 50th and 75th percentiles in the window.
P.10.50.90	The 10th, 50th and 90th percentiles in the window.
P.10.25.50.75.90	The 10th, 25th, 50th, 75th and 90th percentiles in the window.
IQR	The interquartile range in the window.

Feature	Explanation
Norm1	The L1 Norm of points in the window.
Norm2	The L2 Norm of points in the window.
Diff	The consecutive differences between points in the window.
SignDiff	The signs of the differences in the window.
SumSignDiff	The sum of the signs of the differences in the window.
RangeNormalized	Standardized range value in the window.
RangePrevN	Previous N window range values.
RangeNDiffBetween	Previous N range values, and the differences between these ranges.
RangeNDiffFirstRest	Previous N range values, and the differences between the most recent range and the previous ranges.

Table 36. The consolidated testing error values of 7 trips.

Feature	Classification Error (%)	CPDPM Stop Points	CPDPM Start Points	CPDPM Combined
Mean	32	5	5	5
StDev	7.46	2.93	2.9	2.92
Range	7.61	2.84	2.83	2.83
Max	12.27	4.58	4.31	4.45
Median	32	5	5	5
P.25.75	9.58	4.39	4.19	4.29
P.10.90	9.68	3.29	3.17	3.23
P.25.50.75	9.94	4.47	4.31	4.39
P.10.50.90	10.51	3.86	3.53	3.7
P.10.25.50.75.90	10.67	3.98	3.66	3.82
IQR	8.23	3.48	3.53	3.5
Norm1	32	5	5	5
Norm2	32	5	5	5
Diff	32	5	5	5
SignDiff	32	5	5	5
SumSignDiff	32	5	5	5

Feature	Classification Error (%)	CPDPM Stop Points	CPDPM Start Points	CPDPM Combined
RangeNormalized	7.46	3.69	4.03	3.86
RangePrev2	7.09	1.4	1.35	1.37
RangePrev3	7.97	1.11	0.98	1.04
RangePrev4	8.44	1.17	0.97	1.07
RangePrev5	8.6	1.51	1.15	1.33
Range2DiffBetween	7.09	1.4	1.35	1.37
Range3DiffBetween	7.92	1.11	0.98	1.04
Range4DiffBetween	8.44	1.17	0.97	1.07
Range5DiffBetween	8.54	1.51	1.15	1.33
Range3DiffFirstRest	7.92	1.11	0.98	1.04
Range4DiffFirstRest	8.44	1.17	0.97	1.07
Range5DiffFirstRest	8.44	1.5	1.15	1.33

In Table 36, to make distinguishing easier between the experiments, the error values are color coded. The classification error largest value is red, and the lowest values are colored blue. The CPDPM values are coded separately, since error calculation is different from classification error. Here the largest value is red, and the smallest is green. As can be seen in Table 36, some of the features performed extremely badly. These features are the Mean, Median, Norm1, Norm2, Diff, SignDiff, and SumSignDiff. On the other hand, features that measured the variation in the data were more effective.

The first test trip in the Saturn Ion car test set was analyzed closely in order to see how the different features affected the state estimation using the same method with same parameters, and also to check whether the change point detection performance metric was of any benefit. Fig. 47 shows the state prediction results of linear SVM using different features and a *C* value (box constraint) of 1. The bottommost line with a lime green color depicts the observed states of this trip. In each line, the upper segment is the state of being in motion, and the lower segment shows the standstills. The performance values of this specific trip for each case are provided in Table 37. For example, the classification error is the same for the features Q.25.75 and Q.10.90, however their corresponding CPDPM values are different. By checking Fig. 47, it can be seen that the state estimation using feature Q.25.75 was worse than Q.10.90, as it had two more false standstill estimations at times around 3:00 and 4:40. This was captured using the change point detection performance metric, but not with the regular classification error, because point by point comparison of predictions with observed states produces the same total number of misclassified points, hence the same classification error. The best features were shown as Q.25.50.75 and Q.10.50.90 by the classification error, while the CPDPM determined the best features to be RangePrev3, Range3DiffBetween, and Range3DiffFirstRest. The features selected to be the best by CPDPM were all very similar in state estimation, with only one false standstill around 3:50. However, the Q.10.50.90 feature state estimation had four false alarms, and the Q.25.50.75 was even worse, with five false alarms. It is also worth noting that the models with the best features, as determined by CPDPM, had larger classification error values (4.35%) than the ones determined by using the regular classification error (4.04%). This is a further testament that the CPDPM did a better job in identifying the models with regard to quantifying the performance related to change point detection.

Fig. 48 shows the results of applying linear SVM to each of the seven test trips. The best model, as identified by the classification error, suffered a lot from false alarms. CPDPM was more reliable in selecting the better performing model.

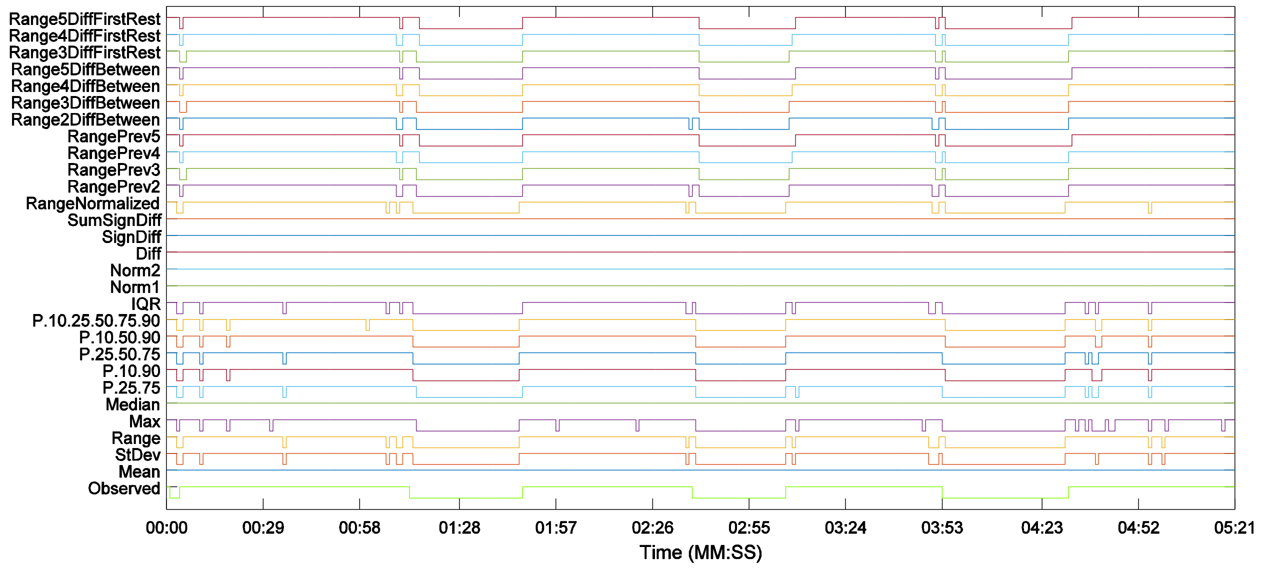


Fig. 47. The SVM predictions of the same test trip of Saturn Ion using different features.

Table 37. The SVM error results for each feature of test trip 1.

Feature	Classification Error (%)	CPDPM Stop Points	CPDPM Start Points	CPDPM Combined
Mean	31.99	5	5	5
StDev	6.21	2.08	2.51	2.29
Range	5.9	1.66	2.09	1.87
Max	7.76	3.59	3.14	3.36
Median	31.99	5	5	5
Q.25.75	4.35	1.59	1.23	1.41
Q.10.90	4.35	1.09	0.97	1.03
Q.25.50.75	4.04	1.31	1.16	1.23
Q.10.50.90	4.04	1.09	0.97	1.03
Q.10.25.50.75.90	4.35	1.32	1.22	1.27
IQR	5.59	2.06	2.48	2.27
Norm1	31.99	5	5	5
Norm2	31.99	5	5	5
Diff	31.99	5	5	5
SignDiff	31.99	5	5	5

Feature	Classification Error (%)	CPDPM Stop Points	CPDPM Start Points	CPDPM Combined
SumSignDiff	31.99	5	5	5
RangeNormalized	4.35	0.89	1.34	1.11
RangePrev2	4.66	0.44	0.81	0.62
RangePrev3	4.35	0.36	0.59	0.47
RangePrev4	4.97	0.41	0.59	0.5
RangePrev5	4.97	0.39	0.64	0.51
Range2DiffBetween	4.66	0.44	0.81	0.62
Range3DiffBetween	4.35	0.36	0.59	0.47
Range4DiffBetween	4.97	0.41	0.59	0.5
Range5DiffBetween	4.97	0.39	0.64	0.51
Range3DiffFirstRest	4.35	0.36	0.59	0.47
Range4DiffFirstRest	4.97	0.41	0.59	0.5
Range5DiffFirstRest	4.97	0.39	0.64	0.51

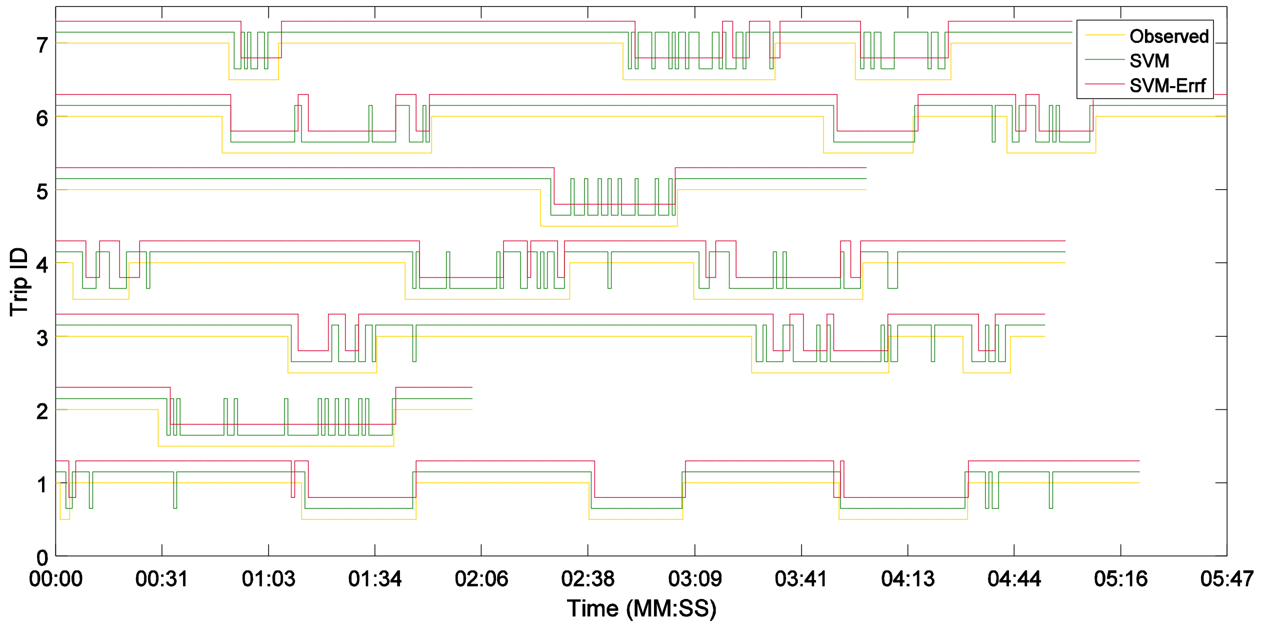


Fig. 48. Lin. SVM, *C*=1, best models based on classif. error (green), and CPDPM (red).

Hidden Markov Models

A hybrid model of SVM, followed by HMM, was applied for predicting the change points in a trip. The observations or emissions were the accelerometer data, and the latent variables were the states of the vehicle, either being in motion or standing still. Although the observations were the acceleration data, using acceleration directly in the HMM model was not possible. A method that mapped the acceleration data to the binary values of 0 for being stationary, and 1 for being in motion, was used. For this purpose, the SVM output of state estimation was used as an input observation vector to the HMM. The summary of the whole process, from data collection to prediction, is depicted in Fig. 49. The transition and emission probability matrices are presented in Table 38. These matrices were optimized by using a simple threshold cutoff method. Windows of one second were taken and the range features were calculated using the accelerometer magnitude. The threshold used was 0.3. Any point whose range value was above this threshold was classified as motion, and below was as standstill. This was the first step. The output from this step was used as an input to the second step, which was the training of HMM. The knowledge of ground truth observed states (based on GPS or OBD speed) was used to construct the transition probability matrix. The emission vector which was the vector of 0s and 1s obtained from threshold method, and the observed state vector, were used to construct the emission probability matrix. After obtaining the trained matrices, the values were rounded, and were used in the rest of HMM applications without any re-training.

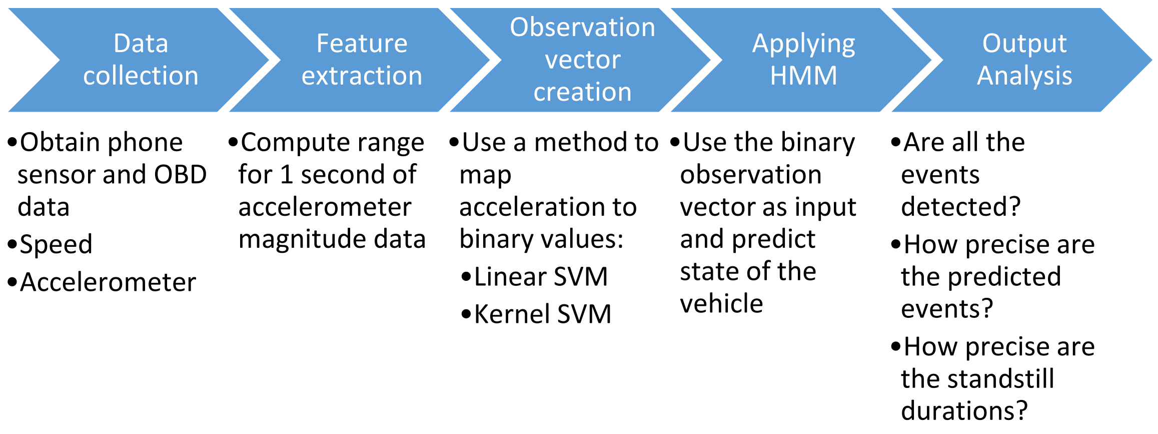

Data collection
Feature extraction
Observation vector creation
Applying HMM
Output Analysis
•Obtain phone sensor and OBD data
•Speed
•Accelerometer
•Compute range for 1 second of accelerometer magnitude data
•Use a method to map acceleration to binary values:
•Linear SVM
•Kernel SVM
•Use the binary observation vector as input and predict state of the vehicle
•Are all the events detected?
•How precise are the predicted events?
•How precise are the standstill durations?

Fig. 49. The summary of the classification process from start to end.

Table 38. The transition and emission probability matrices

$\text{State}_t \rightarrow \text{State}_{t+1}$	**Standstill**	**Motion**	**Actual\ Observation**	**0**	**1**
Standstill	0.90	0.10	**Standstill**	0.85	0.15
Motion	0.10	0.90	**Motion**	0.15	0.85

Application of the SVM-HMM hybrid model results is presented in Table 39. Here, it can be seen that HMM improved on the results of SVM. This is because HMM smoothed out many false alarms, since the transition from dominant motion state to standstill state was rather low. This is evident in Fig. 50, in which state estimation was done on the same test trip as in the SVM. Here, each line represents a model with a different feature(s) as input to the SVM. The output of SVM was used as an emission vector input to HMM. With the application of SVM followed by HMM, nearly all models behaved in the same way, in which they all missed the very first standstill. Except for two models, there were no false alarms. This results hint that SVM-HMM model might not be very good at detecting small stops, because of the low probability associated with them after the Viterbi hidden state prediction process.

Table 39. The average errors of the 7 test trips using HMM for state estimation

Feature	Classification Error (%)	CPDPM Stop Points	CPDPM Start Points	CPDPM Combined P
Mean	32	5	5	5
StDev	5.95	0.69	0.58	0.64
Range	6.06	0.7	0.58	0.64
Max	8.96	1.4	1.3	1.35
Median	32	5	5	5
P.25.75	8.13	0.73	0.65	0.69
P.10.90	8.65	0.65	0.51	0.58
P.25.50.75	8.18	0.68	0.62	0.65
P.10.50.90	8.86	0.86	0.73	0.8
P.10.25.50.75.90	9.01	0.92	0.79	0.86
IQR	6.27	0.64	0.57	0.6
Norm1	32	5	5	5
Norm2	32	5	5	5
Diff	32	5	5	5
SignDiff	32	5	5	5
SumSignDiff	32	5	5	5
RangeNormalized	4.45	0.53	0.53	0.53
RangePrev2	6.63	0.72	0.52	0.62
RangePrev3	7.82	0.95	0.78	0.87
RangePrev4	8.39	0.88	0.65	0.76
RangePrev5	8.39	0.79	0.57	0.68
Range2DiffBetween	6.63	0.72	0.52	0.62
Range3DiffBetween	7.77	0.95	0.78	0.87
Range4DiffBetween	8.39	0.88	0.65	0.76
Range5DiffBetween	8.34	0.78	0.57	0.68
Range3DiffFirstRest	7.77	0.95	0.78	0.87
Range4DiffFirstRest	8.39	0.88	0.65	0.76
Range5DiffFirstRest	8.23	0.77	0.57	0.67

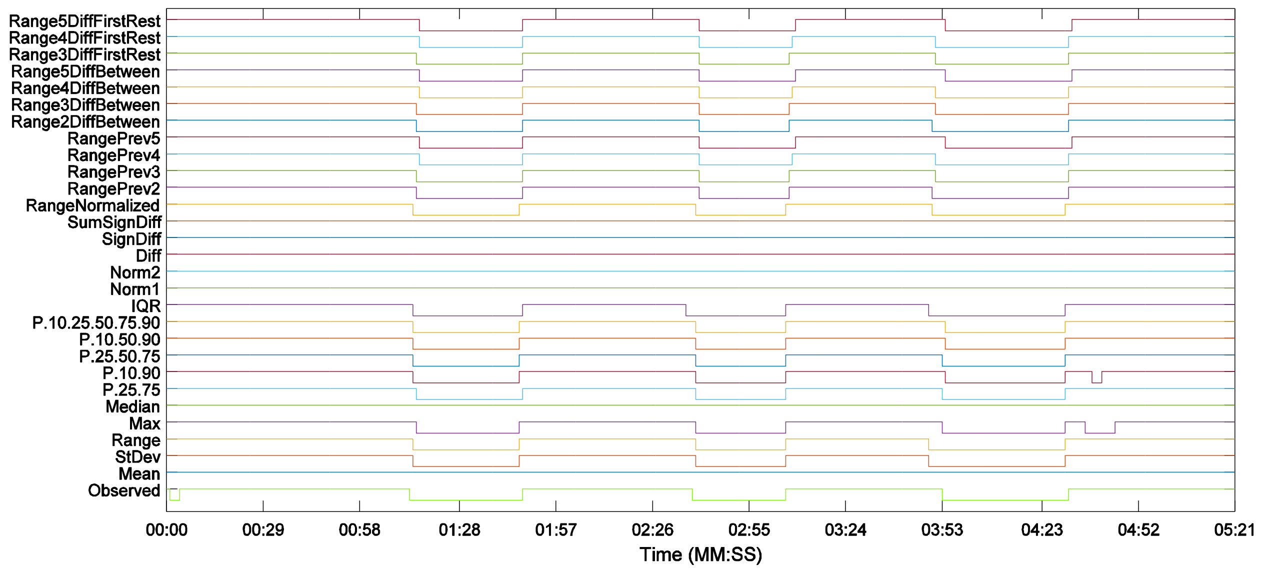


Fig. 50. The state estimation of the first test trip from Saturn Ion using different features.

The error values for the single test trip are provided in Table 40. Here, both the classification error and the CPDPM agreed on the same model as being the best. However, a problem arose when two models had the same number of false alarms. The features of Max and Q.10.90 each had a false standstill towards the end, with the model using Max as feature having a longer duration. The CPDPM gave the same result for both, but the classification error had a larger value for the model using Max. In this case, the classification error did better in identifying the better model. The CPDPM treated all false alarms in the same way, by assigning them the maximum penalty of two in total. Thus, it could not distinguish between their duration. This can be addressed by taking into consideration the duration of a false alarm.

Table 40. The HMM error results for each feature of test trip 1.

Feature	Classification Error (%)	CPDPM Stop	CPDPM Start	CPDPM Combined
Mean	31.99	5	5	5
StDev	3.42	0.41	0.31	0.36
Range	3.42	0.41	0.31	0.36
Max	5.28	0.34	0.31	0.32
Median	31.99	5	5	5
P.25.75	2.17	0.34	0.28	0.31
P.10.90	3.42	0.34	0.31	0.32
P.25.50.75	2.17	0.31	0.31	0.31
P.10.50.90	2.48	0.34	0.31	0.32
P.10.25.50.75.90	2.48	0.34	0.31	0.32
IQR	3.42	0.44	0.28	0.36
Norm1	31.99	5	5	5
Norm2	31.99	5	5	5
Diff	31.99	5	5	5
SignDiff	31.99	5	5	5
SumSignDiff	31.99	5	5	5
RangeNormalized	3.11	0.39	0.31	0.35
RangePrev2	3.42	0.44	0.28	0.36
RangePrev3	3.11	0.42	0.28	0.35
RangePrev4	3.73	0.44	0.31	0.38
RangePrev5	4.04	0.42	0.36	0.39
Range2DiffBetween	3.42	0.44	0.28	0.36
Range3DiffBetween	3.11	0.42	0.28	0.35
Range4DiffBetween	3.73	0.44	0.31	0.38
Range5DiffBetween	4.04	0.42	0.36	0.39
Range3DiffFirstRest	3.11	0.42	0.28	0.35
Range4DiffFirstRest	3.73	0.44	0.31	0.38
Range5DiffFirstRest	4.04	0.42	0.36	0.39

Fig. 51 shows the application of linear SVM followed by HMM to each of the seven test trips. The best feature was selected based on the two error metrics. The green line is the model selected by classification error, and the red line is the model selected by CPDPM. Both of the error metrics agreed on the same feature to select the best model. That's why both the green and the red lines are on top of each other. It is clear that SVM-HMM hybrid model had many fewer false alarms, but also missed some true change points.

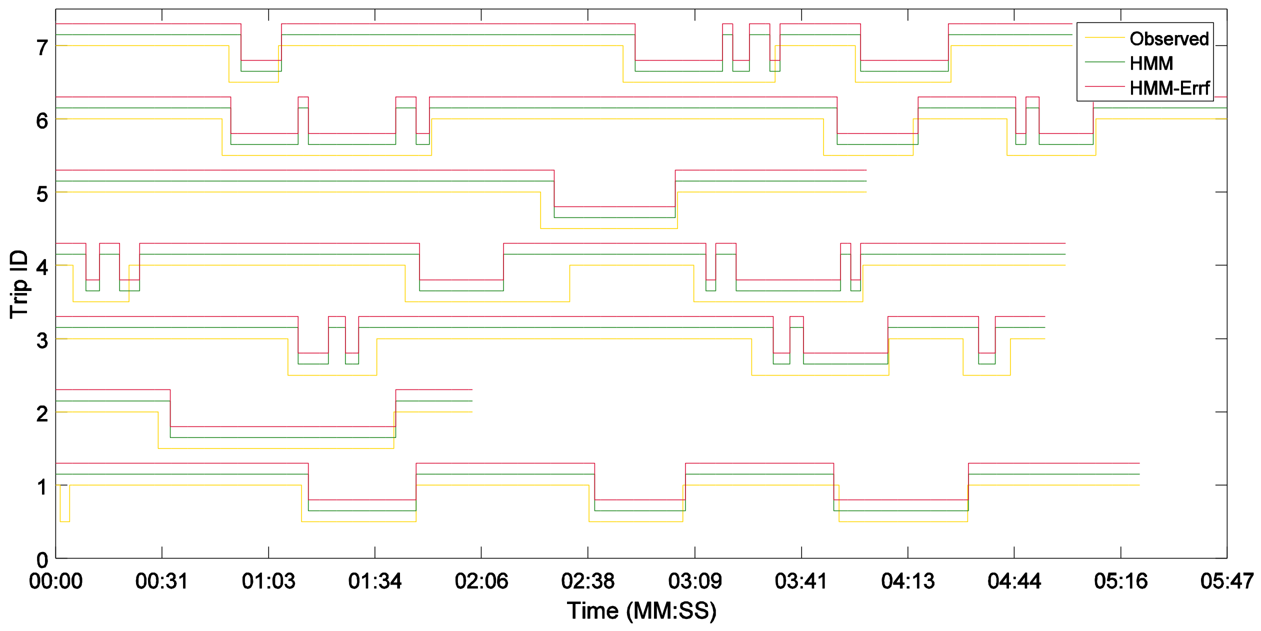


Fig. 51. The best HMMs. Classif. error (green), CPDPM (red) agree on the same feature.

Linear SVM can be improved by experimenting with different C values. C values in a large range from 0.0001 to 50 were analyzed. The following table shows the best models using the two error metric criteria, the best feature, and the best box constraint parameter values. Using the best SVM model in each case, the state estimations were drawn and are shown in Fig. 52 and Fig. 53. The SVM performed better than the one in Fig. 50, and so did the HMM. Both methods were smoother and had fewer false alarms.

Table 41. The best linear SVM by using grid search over several C values.

Method	Error Metric	Feature	Box Constraint
SVM	ClassError	P.10.25.50.75.90	0.01
	CPDPM	RangePrev5	0.008
SVM-HMM	ClassError	P.10.25.50.75.90	0.01
	CPDPM	P.10.25.50.75.90	0.008

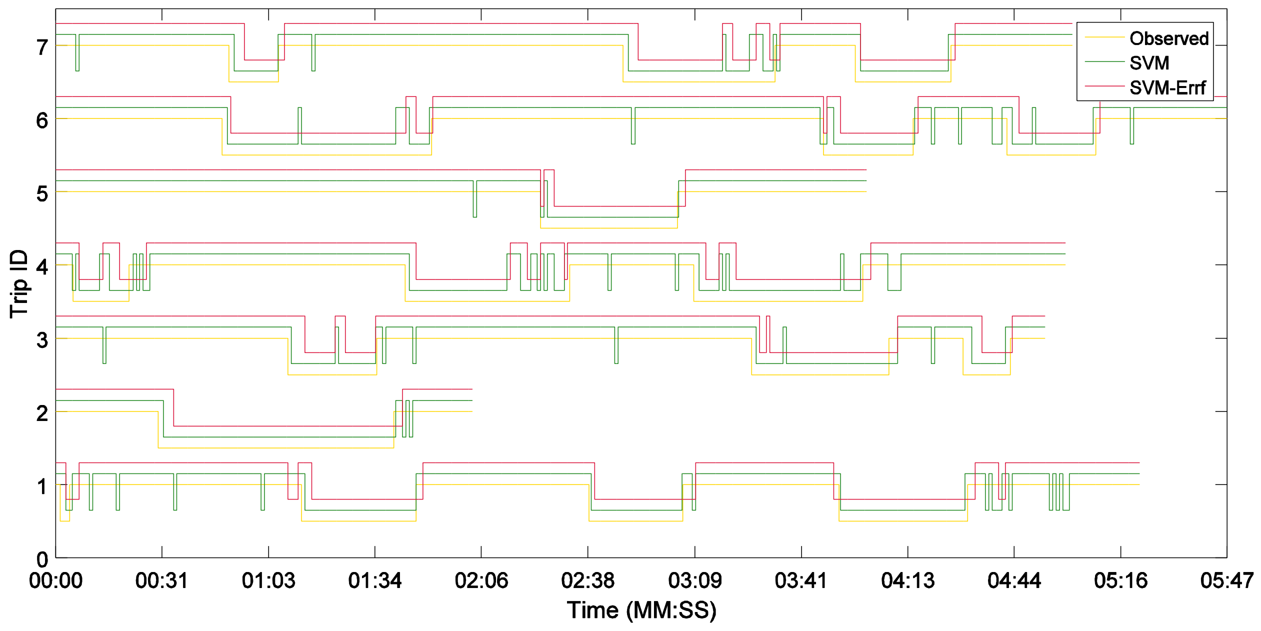


Fig. 52. The best SVMs based on Classification error (green), CPDPM (red).

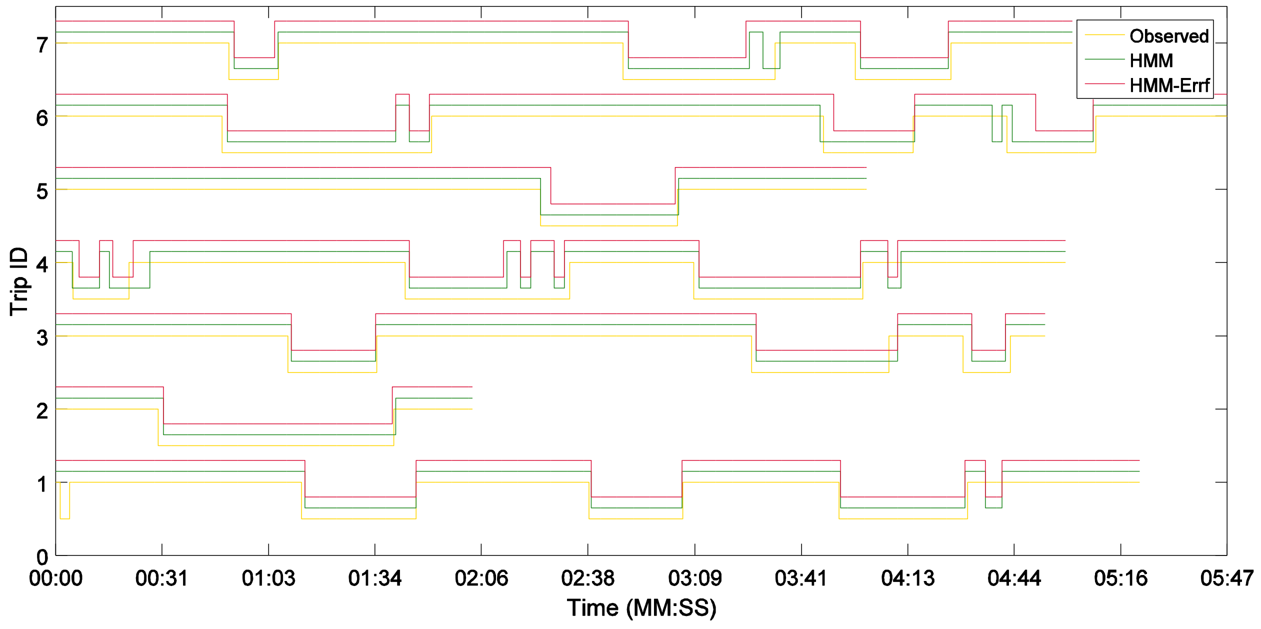


Fig. 53. Best HMM results. Class. error (green), CPDPM (red) agree on the same feature.

The RBF SVM is another method that is more suitable for linearly non separable data. The following figure shows the RBF SVM and HMM results.

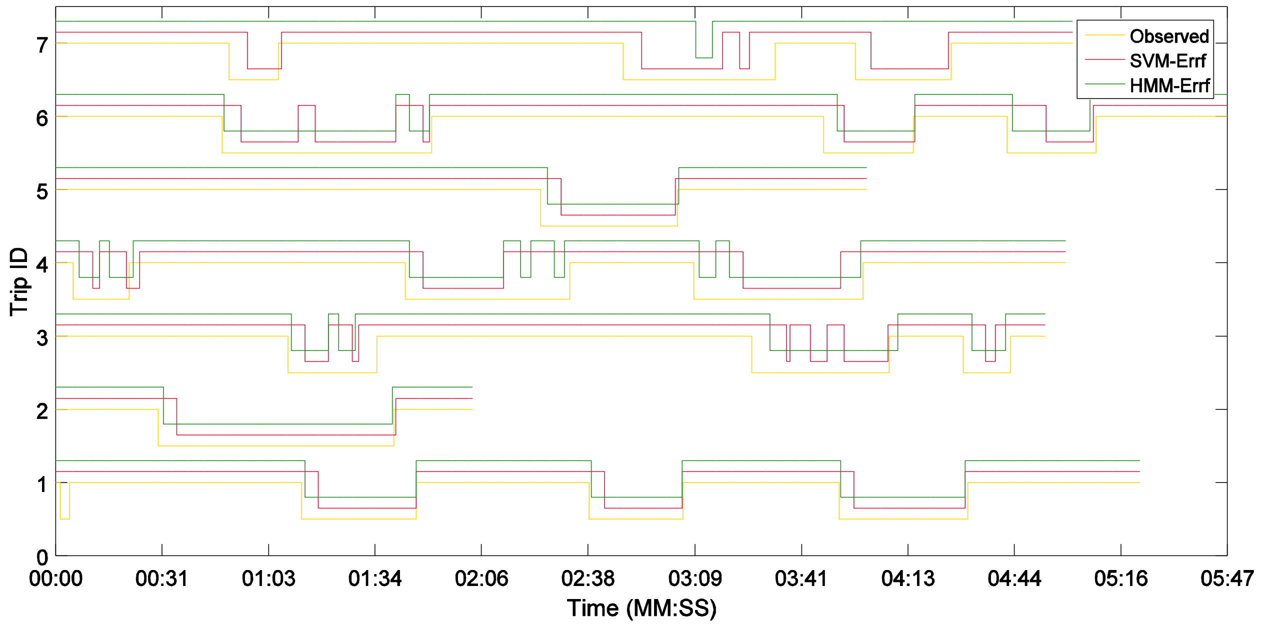


Fig. 54. RBF SVM state estimation.

APPENDIX B: PROBLEMS RELATED TO SENSORS

During data collection and processing many problems have been encountered with. The most common problem was the loss of GPS signal. Another problem related to GPS was the fact that it did not show 0 mph, even when the car was at full stop. Sometimes it lingered steadily around 3, or 10 mph, and sometimes it just behaved erratically going up and down when the vehicle was at stop. The weird behavior of accelerometer data was also seen in the data. Although very seldom, the loss of OBD signal was also encountered, possibly due to an interference with the Bluetooth signal between the OBD and the phone. Some of these problems are shown in the figures below. In Fig. 60, change of phone orientation can be seen at around 8 min mark. It is also interesting that the GPS was not able to receive the correct signal until before this point.

Due to the anomalies that were found in the data, the threshold for creating ground truth data in which it is determined when a car is moving and when it is at standstill is set to 3 mph when using the GPS speed, and to 1 mph when using the OBD speed.

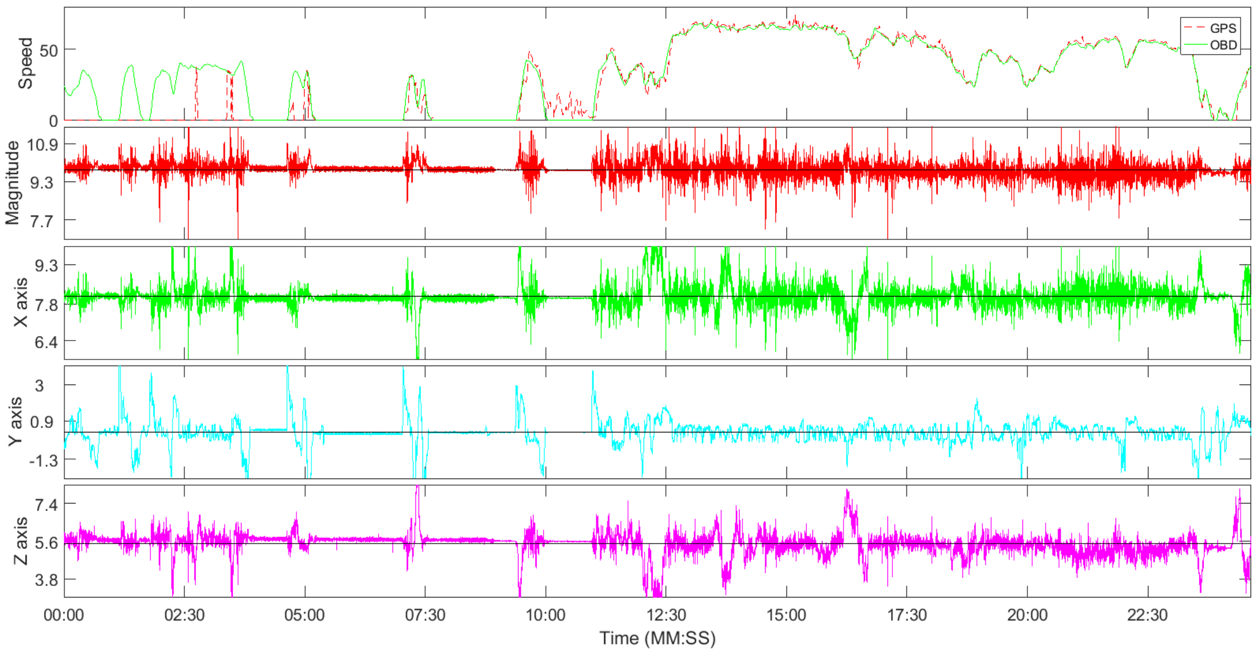


Fig. 55. Loss of GPS signal several times, not reaching zero when the vehicle has stopped.

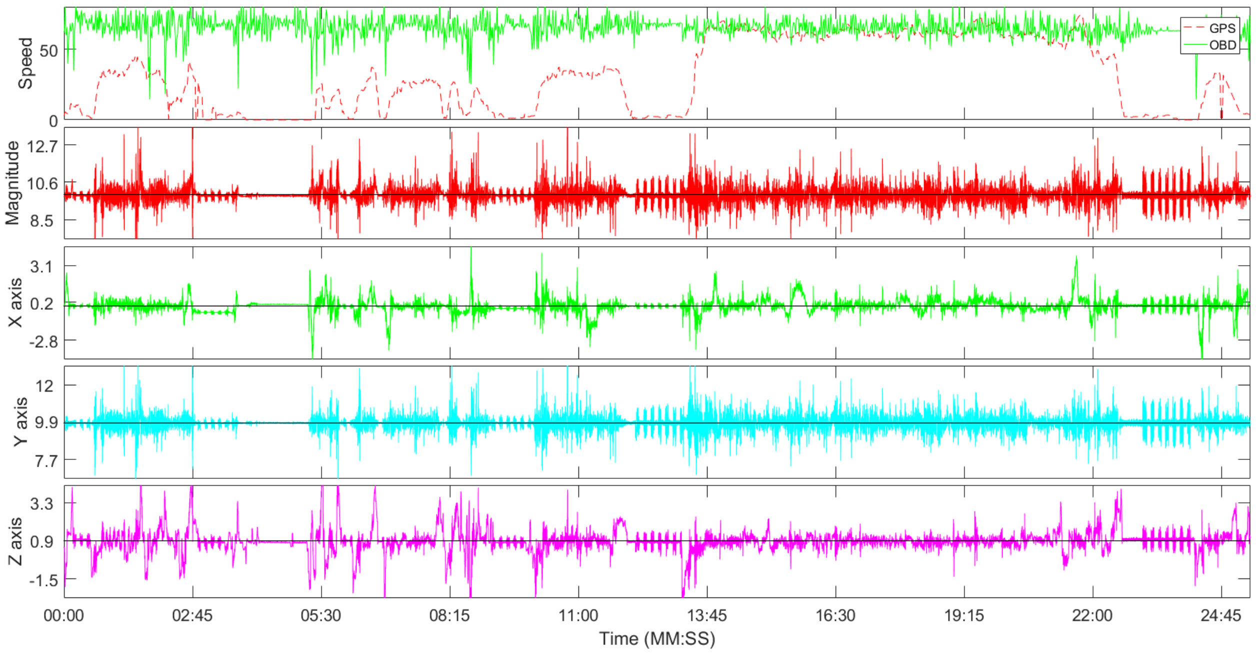


Fig. 56. Loss of OBD signal, and the erratic behavior of the accelerometer data.

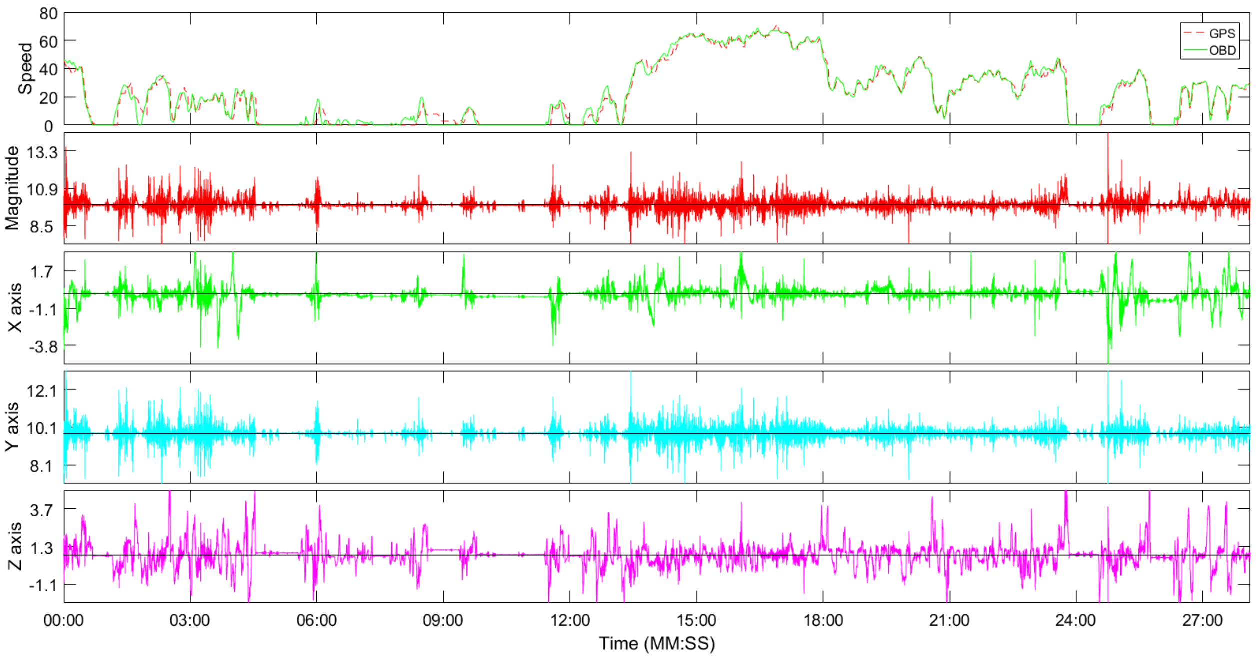


Fig. 57. Some large variation in accelerometer data when the vehicle is stopping.

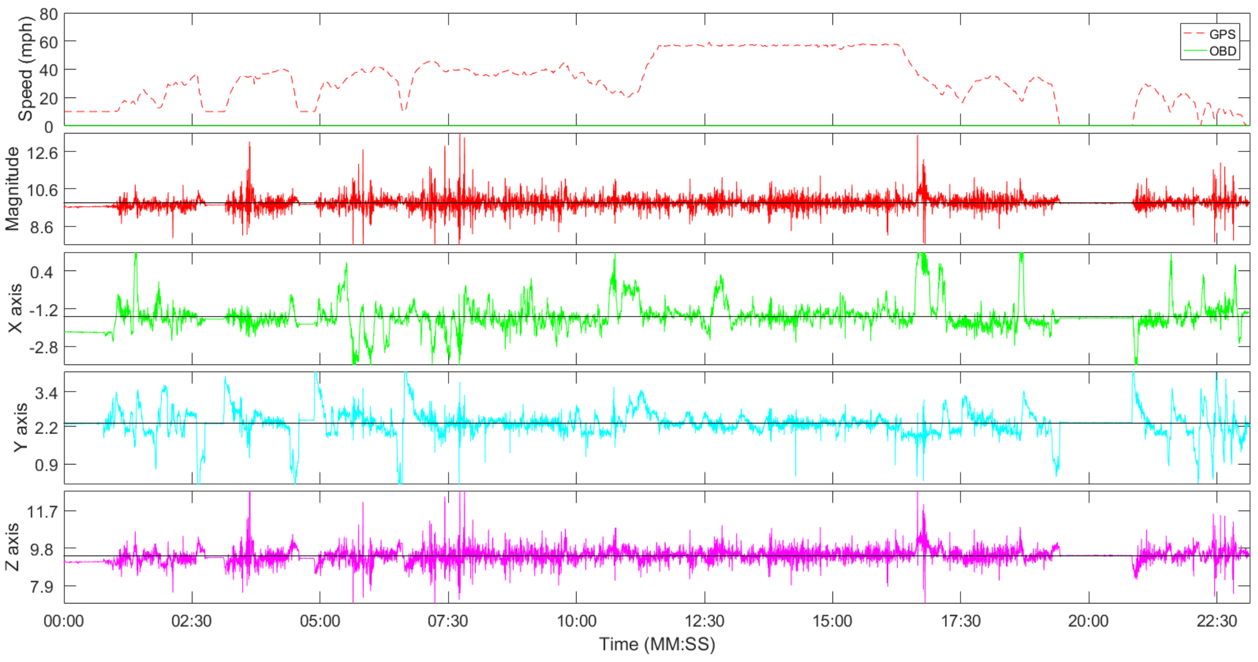


Fig. 58. GPS signal not going below 10 mph even when the car is stopping.

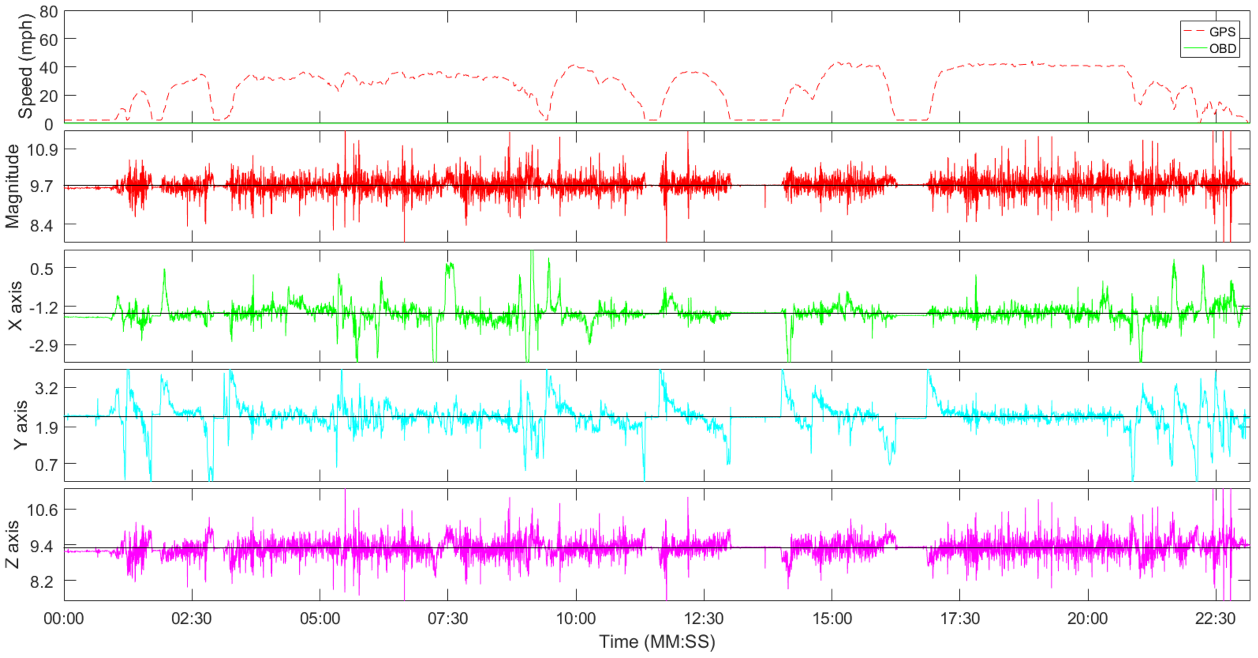


Fig. 59. GPS signal hanging around 3 mph while the car is in total standstill.

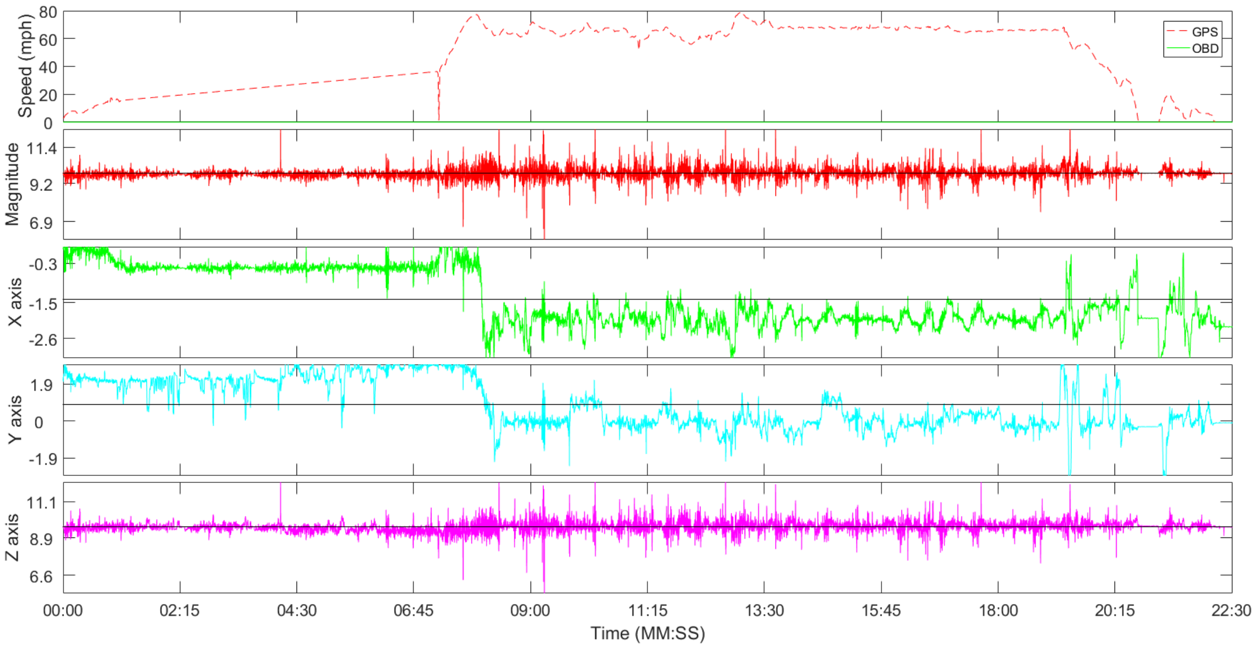


Fig. 60. Change of orientation around 8 min mark, and lack of GPS at the beginning.

APPENDIX C: HMM VALIDATION

In this appendix, the transition matrices of the HMM models, and the plots of the validations results of the best models presented in Section 5.2.3 are given. Fig. 61 to Fig. 63 show the state predictions of the best HMM models, where the best models are selected based on the CPDPM. In the plots, green lines represent the observed true states of the vehicle, and the red lines represent the predicted states. Each of the ten trips in the validation sets are drawn separately in their respective subplots, enumerated 1 to 10. Since there are two states, the upper segments of the lines show vehicle motion periods, while the lower segments of the lines show the standstill periods. The vertical lines represent the change points. The x axis shows the time in Min:Sec format. Since some trips were much longer than the rest, to keep it consistent, each trip was truncated at 20 minutes.

The trained transition probability matrices for the best performing models of the three vehicles are shown in Table 42. It can be seen that the transition probabilities between different states are very low.

Table 42. HMM Trained State Transition Matrices

	Prius		Camry		Mazda	
State	0	1	0	1	0	1
0	0.99563	0.00437	0.99611	0.00389	0.99646	0.00354
1	0.00065	0.99935	0.00070	0.99930	0.00097	0.99903

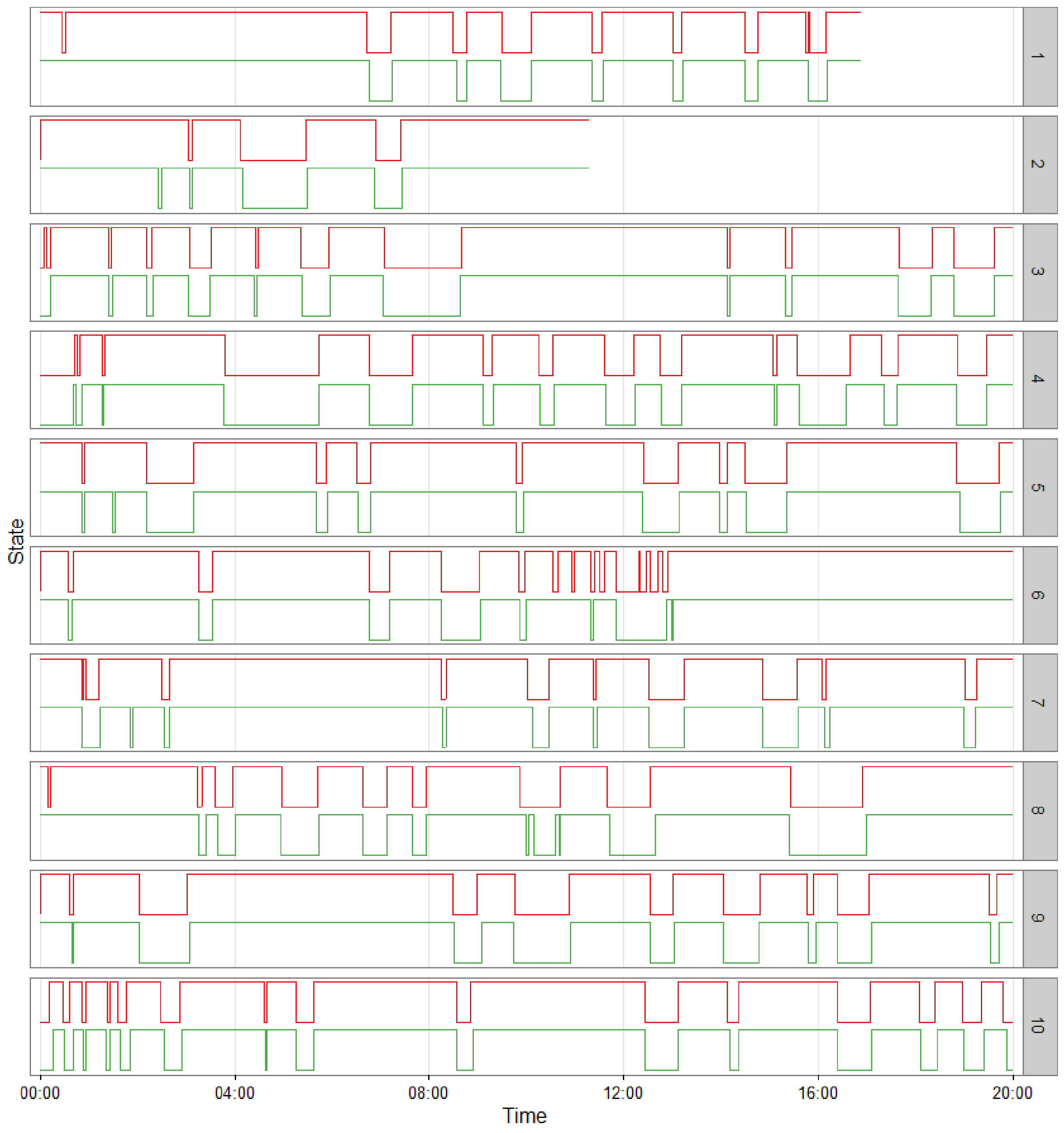


Fig. 61. Prius state estimation of validation trips. Red: Prediction, Green: Observed.

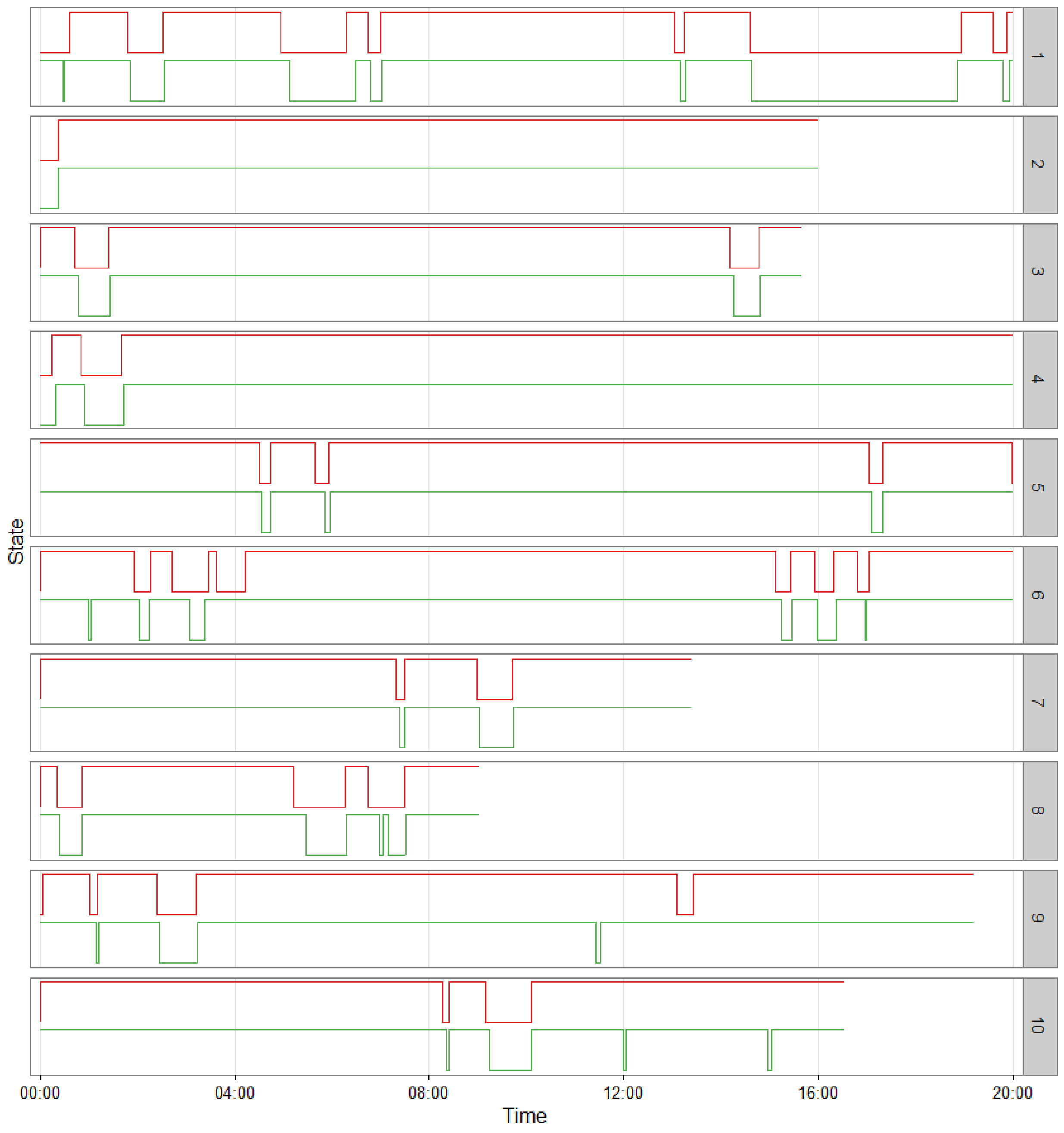


Fig. 62. Camry state estimation of validation trips. Red: Prediction, Green: Observed.

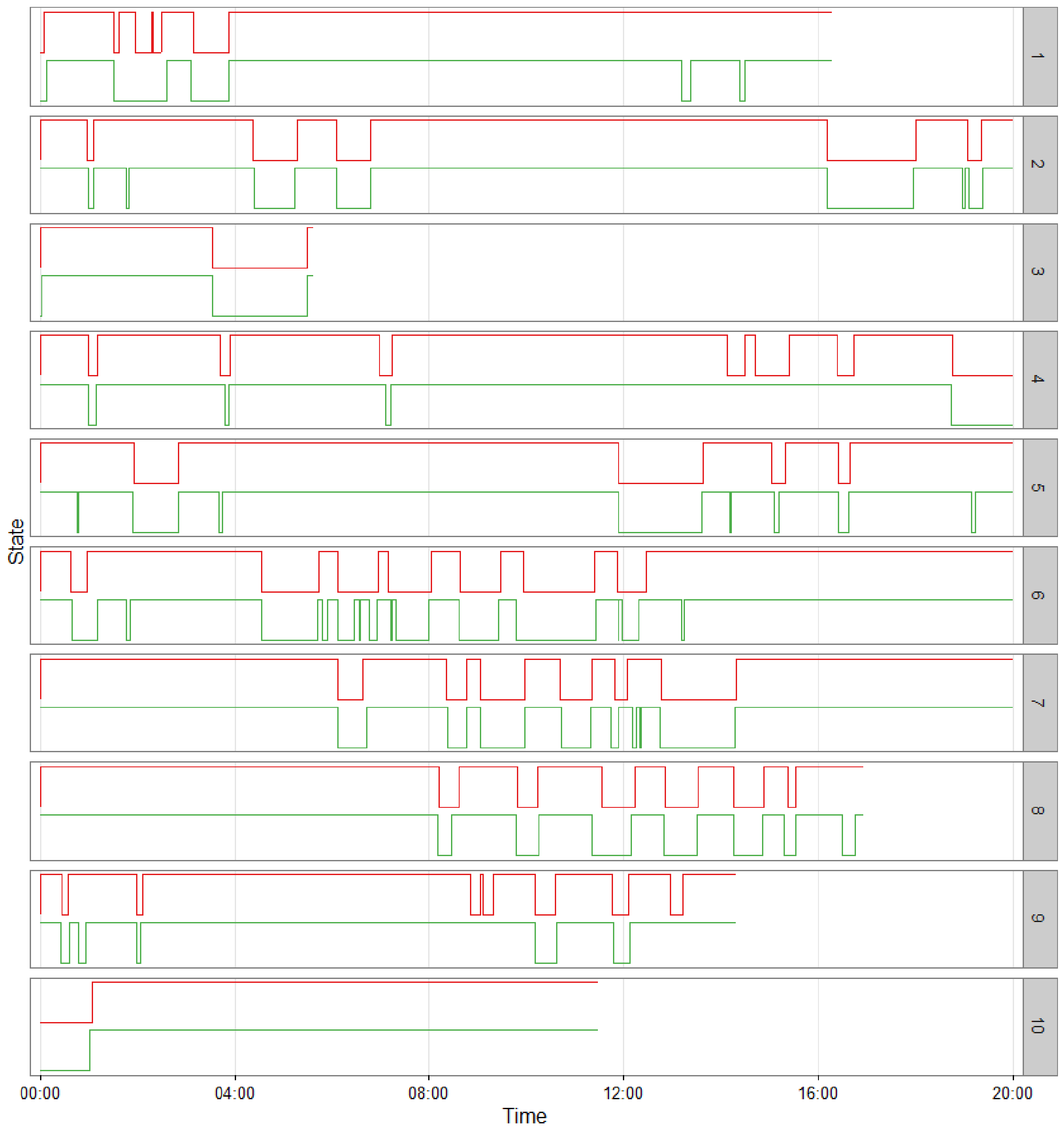


Fig. 63. Mazda state estimation of validation trips. Red: Prediction, Green: Observed.

VITA

Ilyas Ustun earned his B.S. degree with summa cum laude from Fatih University in Istanbul in 2008. After working a year at Fatih University as a graduate research assistant, he was admitted to the PhD program in the department of Modeling, Simulation, and Visualization Engineering at Old Dominion University. Later, he became a graduate research assistant at the Transportation Research Institute. He served as the President of the International Student Advisory Board at Old Dominion University in 2015. He was the founding president of the Transportation Engineering Students Organization in 2015. His research interests include data mining, data analysis, machine learning techniques, descriptive and predictive analytics, modeling and simulation of transportation systems, and working with data collected from various sources such as probe vehicles, weigh-in-motion scales, and smartphone sensors. He is very enthusiastic about data science, and publishes some of his work on his blog site datacademy.wordpress.com. For more information about Ilyas, please check his LinkedIn page: linkedin.com/in/ilyasustun.