# Determining Transportation Mode through Cellphone Sensor Fusion

by

Maria Frendberg

Submitted to the Department of Electrical Engineering
and Computer Science
in partial fulfillment of the requirements for the degree of

Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011



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

Certified by .....
Professor Seth Teller
Electrical Engineering and Computer Science Department
Thesis Supervisor

Accepted by .....
Christopher J. Terman
Chairman, Undergraduate Thesis Committee

# Determining Transportation Mode through Cellphone Sensor Fusion

by

Maria Frendberg

Submitted to the Department of Electrical Engineering
and Computer Science
on May 18, 2011, in partial fulfillment of the
requirements for the degree of
Bachelor of Science in Computer Science and Engineering

## Abstract

Contextual awareness is a field that allows computer systems to provide users and their applications with valuable information about the world around us, such as the 'Mode of Transportation' of a given user. This data can be used in many different ways, for example, the analysis of traffic patterns and $CO_2$ emissions.

In this thesis, I designed and implemented a mobile application that determines the user's current mode of transportation. This is done by applying a Boosted Naïve Bayes classifier to data collected from the sensors of the mobile device.

Generating this Boosted Naïve Bayes classifier required me to first develop an application to collect training data. I then applied several Naïve Bayes classifiers to these data. Finally, I applied the AdaBoost algorithm to these classifiers to obtain the boosted classifier.

The Boosted Naïve Bayes classifier performs better than its unboosted counterparts. Boosting can be a useful tool to apply to context recognition problems.

Thesis Supervisor: Professor Seth Teller
Title: Electrical Engineering and Computer Science Department

# Acknowledgments

I would like to thank Professor Seth Teller, Dorothy Curtis, Jonathan Ledlie, Jungeun Park and Ami Patel for their thoughts and assistance over the course of my work. Their input was invaluable and greatly contributed to the success of my project.

I would also like to thank Josh Siegel for his assistance in developing the original concept of my project and for his input over the course of my work.

Finally, I would like to thank my friends and family for their support during the course of my thesis work and during my time at MIT.

# Contents

# Chapter 1

# Introduction

Context-aware systems have inspired new interfaces and applications that allow for advances in many fields including smart environments, surveillance, emergency response, healthcare, and military missions. Contextual awareness allows these systems to make more educated predictions about the environment around them and act accordingly. For example, such a system implemented in an automobile could detect various properties about a crash, such as its direction and severity, and could react accordingly by sending Emergency Service to a severe crash, while not sending them to a simple fender-bender. This is seen in the services provided by OnStar and others [7].

Mode of transportation data can also provide valuable information and services. For example, due to the recent legislation regarding texting while driving, a contextually aware mobile application could restrict a user's texting ability when it determined the user was driving, rather than simply a passenger in, a moving vehicle. Mode of transportation data could also be used in traffic analysis. If a user was determined to be in traffic and stop-and-go motion was detected, the system could assume the user was in traffic and potentially reroute other drivers to more evenly distribute traffic. Another application of this data includes the determination of the $CO_2$ footprint of a user and it could perhaps act as an exercise log.

While there exist many different sensing devices that are used to determine some sort of contextual awareness, the use of mobile phones, specifically smart phones, as

sensing devices are becoming evermore common. This is due to the recent surge in the popularity of smartphones. In fact, Nielsen predicts that by the end of 2011, half of all cell phone users will own a smartphone [2]. This provides a unique opportunity for contextually aware systems by providing them with a large user base that already carries the necessary hardware around with them on a daily basis.

Contextual data can be extremely valuable, and mobile phones now make it relatively easy to obtain. Due to these factors, I decided to develop a project that uses data obtained from a mobile phone to predict the mode of transportation of a user. This system will exist as an application on the Android platform and the results gathered from it can be used by other applications to customize their interfaces and activities to the user's context and activities.

# Chapter 2

# Theory

A brief background of the theory behind the methods used in my project is presented here for the reader.

## 2.1 Machine Learning

Machine learning is a branch of artificial intelligence that includes the development of algorithms to analyze and classify data. There are two types of machine learning algorithms: unsupervised and supervised [14].

### 2.1.1 Unsupervised Learning

Unsupervised machine learning consists of trying to find a hidden, underlying structure in a collection of data. The most common example of unsupervised learning is clustering in which data points are 'clustered' into sets with other data points to which they are the most similar.

### 2.1.2 Supervised Learning

Supervised machine learning is concerned with classification, the development of functions that map each input to the correct output. In contrast to unsupervised learning algorithms, which are able to determine their own groupings of the data, supervised

learning algorithms are given these groupings. This supervision allows the user to specify groupings which are more useful for his purposes. This function is developed through the use of training data, which 'teaches' the machine the correct output for a handful of inputs. This method of teaching varies depending on the specific algorithm used. While there exist many supervised machine learning algorithms, only two need to be discussed for the purposes of this project.

### Naïve Bayes

A Naïve Bayes classifier is a simple probabilistic classifier that is based on applying Bayes' Theorem with strong independence assumptions. Given a training data set with two possible classifications and datapoints with one value, a Naïve Bayes classifier is trained by learning the mean and standard deviation of values of each classification of the training data. Once the classifier is trained, it can then be given an input point to classify. The classifier determines which distribution the point is more likely to fall in, and returns that as the classification. This classifier can also be applied to datapoints with multiple values by constructing distributions for each individual value. Unknown datapoints are then classified by determining the product of the likelihoods of each value.

### Boosting

Boosting is a machine learning meta-algorithm that constructs a strong classifier (classifiers that produce classifications that are fairly well-correlated with the actual classifications) from multiple weak classifiers (classifiers that do barely better than random guessing). There are multiple ways to implement boosting, but we will use the AdaBoost algorithm, due to Schapire[10]. The idea behind this algorithm is similar to that of a consensus, where the classification is that chosen by the majority of the classifiers. The major difference between boosting and consensus is that each classifier is given a different weight in the boosted classifier. Classifiers that produce more accurate predictions are weighted more heavily than those which produce less accurate predictions. This allows the most accurate classifier to make the initial

classification of each datapoint, but it can be overruled if enough of the remaining classifiers disagree. A rough sketch of this algorithm is presented here, with a brief explanation below.

Given: $(x_1, y_1), ..., (x_m, y_m)$ where $x_i \in X, y_i \in Y = -1, +1$, Initialize $D_1(i) = \frac{1}{m}, i = 1, ..., m$. For $t = 1, ..., T$:

- Train base learner using distribution $D_t$.
- Update $D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$ where $Z_t$ is a normalization factor (chosen so that $D_{t+1}$ will be a distribution).

Output the final classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$.

In this algorithm, we see that all the classifiers have the same initial weight and, through each iteration, these weights are revised to more accurately classify the data. The algorithm is run for $T$ iterations. We hope that the values of the weights have converged after $T$ iterations, but there exist classifiers for which the weights of the boosted classifier never converge.

# Chapter 3

# Previous Work

Much work has been done in the field of context awareness. Context awareness can provide us with valuable information about the world around us. It is commonly used to provide users with valuable services, such as services that change their actions based on the user's current activity or services that listen for and react to certain situations. For example, context awareness is used in devices given to elderly users to determine if the user has fallen. This allows the device to make the necessary calls to emergency services, etc.

Determining the Mode of Transportation (MOT) of a user is an important subfield of context awareness. MOT information has many uses, such as determining whether a user is walking or driving to determine their $CO_2$ output [5] or collecting traffic data. Many different approaches have been taken to MOT determination, several of which are discussed below.

## 3.1 Use of Mobile Devices in Context Awareness

The use of mobile devices in both context awareness [8, 15, 1, 3] and more specifically MOT awareness [9, 12, 4, 11, 5] is well documented. Mobile devices are often a logical hardware choice due to the sensors they provide, their cost-effectiveness, and their high presence amongst users. However, they also introduce limitations in the choice of sensors, the quality of these sensors, and in the memory and computing power

available for analysis of sensor data.

## 3.2 Use of Sensors in Context Awareness

The most commonly used sensor in context awareness is the accelerometer [9, 8, 15, 1, 12, 3, 4, 5] due to the fact that it is able to detect the fine-grained movements of a user in three axes. In fact, many approaches to context awareness use only accelerometer data as their input.

In MOT awareness, location data from both the GPS [9, 1, 4] and network towers [11] are frequently used. This allows the recognizer to estimate the speed of a user, which can be a useful measurement in MOT awareness. However, GPS sensors are notorious for their high power usage and should thus be avoided in systems which run continuously or frequently. An interesting approach to this issue was taken by Want, et. al [11]. Instead of using GPS data, they instead read location data from the network towers as provided in the user's call records. This allowed them to determine location and speed information without discharging the battery too quickly.

Various other sensors, such as light sensors, temperature sensors, etc. are used in other implementations of context awareness systems [1]. Depending on the device used, not all of these sensors may be available.

## 3.3 Use of AI in Context Awareness

AI algorithms are another tool frequently employed in context awareness. Most users decide to apply a supervised learning algorithm, including, but not limited to:

- Naïve Bayes [3]
- Hidden Markov Models [9]
- Neural Nets [15]
- Decision Trees [12, 5]
- k Nearest Neighbors [12]
- Support Vector Machines [12]
- Boosting [1]

Work presented by Ravi et. al. compares the effectiveness of many of these algorithms when applied to a context awareness problem [8]. They applied these algorithms to four different types of datasets:

1. Data collected from a single subject, mixed and cross-validated
2. Data collected from multiple subjects, mixed and cross-validated
3. Training data collected from a single subject used for classifying the data from the same subject
4. Training data collected from a single subject used for classifying the data from a different subject

They found that plurality voting was the most effective approach when applied to the first three datasets, with accuracies of 99.57%, 99.82%, and 90.61% respectively. However, when they examined the fourth, where training data was collected from one subject and used to classify the actions of another, they found that a Boosted SVM was the most effective approach, with an accuracy of 73.33%. Boosted Naïve Bayes was not far behind, with the third-highest score on the fourth dataset and higher scores than a Boosted SVM on the first three. This, included with the relative simplicity of its implementation, led me to use a Boosted Naïve Bayes classifier in my algorithm.

## 3.4 Related Work

The work most closely related to my proposed project is the Mobile Sensing Platform (MSP) [1]. MSP is a small, wearable device that is able to recognize the activity of a user, from a developer-defined set, with up to 93.8% accuracy. To do this, data is processed from the following sensors: electret microphone, visible light photo-transistor, 3-axis digital accelerometer, digital barometer and temperature sensor, digital IR and visible+IR light sensor, digital humidity/temperature sensor, and a digital compass. As in my approach, various classifiers were then applied to these raw data, and finally a boosting algorithm was applied to produce a single strong classifier. Although the MSP project is focused on activity awareness, whereas I am

focusing on Mode of Transportation determination, the project offered me valuable insight into the various classifiers that I could use on my raw data and into the implementation of the boosting algorithm. The most significant difference between MSP and my project is MSP's use of custom hardware. This allows the developers of the MSP to choose exactly which sensors to use and does not limit them to those provided by a mobile phone. This could cause a higher accuracy in activity prediction. However, the MSP also requires the user to wear or carry an extra device, which could be inconvenient to many users. My approach is limited by the fact that it uses only the sensors provided by a mobile device. However, my approach benefits from the large percentage of the population that carries a cell phone with them on a regular basis.

While many of the projects discussed above exhibit similarities to my project, my work is unique due my choice of sensors and my choice of classification approach. Very few of the systems that I found used data from sensors other than accelerometer and location sensors. In contrast, my approach uses data from the orientation sensor, the magnetic sensor and the GPS in addition to the accelerometer. The data provided by these additional sensors allows me to calculate more accurate results by providing me with data which allows the algorithm to make finer distinctions between various MOTs. Also, aside from the Mobile Sensing Platform, I was unable to find work that used a boosting algorithm on their classifiers. We have seen that boosting is an effective approach in classifier development [8], and therefore I determined that it would be an effective approach to this problem. I expect that the combination of these two improvements on previous work will provide me with a highly effective approach for determining mode of transportation.

# Chapter 4

# Methods

Here we describe the systems and methods used to develop the Mode of Transportation classifier.

## 4.1 Test Bed

The Android platform was chosen to be the test bed for this analysis. Specifically a Motorola Droid X smartphone running Version 2.2 of the Android Operating system. This device comes equipped with the following sensors:

- 3-Axis Accelerometer
- 3-Axis Magnetic Field Sensor
- Orientation Sensor
- Light Sensor
- Proximity Sensor
- Temperature Sensor
- GPS
- Network

This choice was made for three reasons. First, the Android platform offers a flexible, easy-to-use API that allows a developer to conveniently access and manipulate data generated by the sensors. Second, the sensors available on the Droid X are representative of what can be found on many new smart-devices and thus allow us to incorporate data from many sensors that had previously not been explored on

a mobile device. Finally, this testbed was chosen due to convenience. The author owned a Motorola Droid X before the start of this project and was already familiar with programming Android applications, thus the use of this platform was a logical choice.

All data analysis and classifier generation was done using MATLAB [6] due to ease of use in processing and visualizing this type of data. MATLAB is a commonly used software tool, thus the code developed during this project could potentially be used by other developers as they approach similar projects.

## 4.2 Data Collection

The first stage of this project consisted of collecting data to be used as training data. This was done through an Android application developed specifically for this project.

### 4.2.1 Data Collection Application

The data collection application consisted of an interface with three tabs and was designed to carry out two main tasks. The first task was to read the values at all available sensors and write to file. The real-time values of each sensor are displayed in the third tab ( Figure 4-1). The second task was to provide an interface that the user could utilize to specify their current mode of transportation. This interface allows a user to actively, proactively, and retroactively indicate their mode of transportation for a given time range (Figure 4-2). The active mode indication occurred in the first tab in which the user can select various modes of transportation from a drop down menu. The proactive and retroactive mode indication occurred on the second tab (Figure 4-3 and Figure 4-4) where the user can view and edit previous time frames and create their own future time frames.

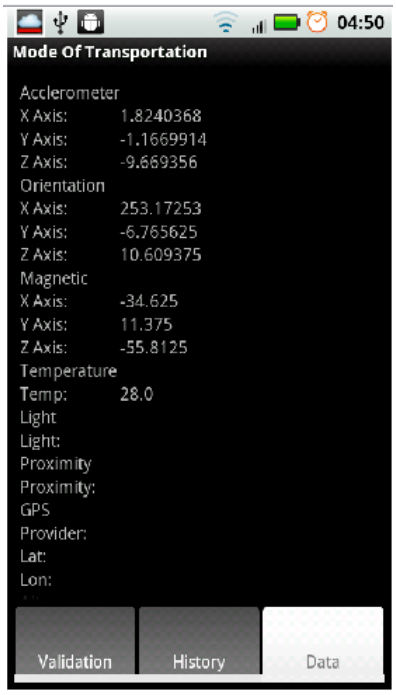


Figure 4-1: Sensor Data Interface

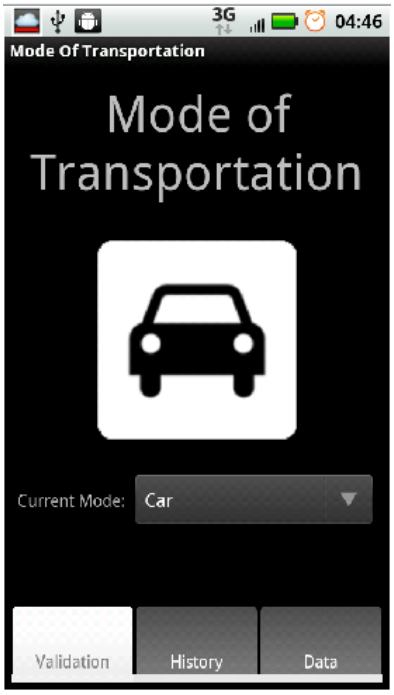


Figure 4-2: Current MOT Interface

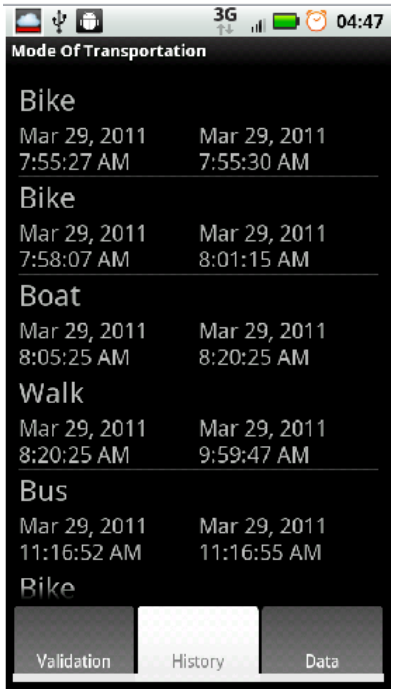


Figure 4-3: Previous Timeframe Interface

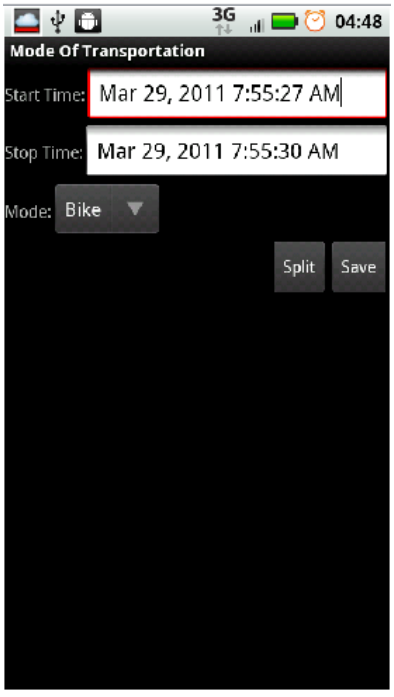


Figure 4-4: Timeframe Editing Interface

### 4.2.2 Output

The output of this application consists of three text files that were saved to the SD card of the device. The first file, 'data.mot', consists of a list of data samples and a collection timestamp for each data sample. The second file, 'ranges.mot', consists of a list of time ranges and the mode of transportation associated with each range. The final file, 'log.mot', is a log showing the changes the user made to 'ranges.mot'. These files can then be easily imported into MATLAB for data analysis.

## 4.3 Data Analysis

The data were partitioned into timeframes of one second. These timeframes were initially analyzed by developing a set of Naïve Bayes classifiers, using various preprocessing techniques, that classified the raw data from the accelerometer, magenetic sensor, and orientation sensor. The data from the location sensors (GPS and network) were first used in conjunction with the timeframe data to determine a speed at each datapoint. For any given datapoint, this was done by determining the distance between, and the time elapsed between, the location measurements at the datapoints directly preceding and following the specified datapoint in the timeframe. This distance was calculated using the Haversine Formula [13]. The distance was then divided by the elapsed time to produce an estimate of the speed of the user over the specified timeframe. This speed data was then also used to train a set of Naïve Bayes classifiers that used the same preprocessing techniques as mentioned above.

### 4.3.1 Preprocessing Techniques

The preprocessing techniques used in the development of the classifiers were inspired by work from Figo et. al.[3]. Only classifiers which Figo determined to be useful on a mobile device were implemented. We define a classifier as being useful if it has reasonable computational costs and storage requirements and it is at least 50% accurate.

#### Time Domain Preprocessing Techniques

Time domain preprocessing techniques consist of both a set of mathematical and statistical functions and a set of other functions. These functions are applied to each value over all the datapoints in the time frame. The mathematical and statistical functions used as features include the following:
**Mean** The average of each value
**Median** The median of each value
**Variance** The variance of each value
**Standard Deviation** The standard deviation of each value
**Minimum** The minimum of each value
**Maximum** The maximum of each value
**Range** The magnitude of the range of each value

The other functions used as features include the following:
**Differences** The average difference between neighboring datapoints
**Angular Velocity** The angular velocity about an axis between neighboring datapoints
**Zero Crossings** The number of times a value crosses through its mean
**Signal Vector Magnitude** The average signal vector magnitude, calculated as $\frac{1}{n}\sum_{i=1}^{n}\sqrt{x_i^2+y_i^2+z_i^2}$, of each sensor

### 4.3.2 Bayesian Analysis

Once each preprocessing technique was applied to both the driving and walking data, a Naïve Bayes classifier was then used to determine a threshold value for each classifier to identify between the two states. This was done by determining where the fitted normal distributions intersected. Above this intersection point, one state is more likely to occur than the other whereas, below this intersection point, the opposite is true. We see an example of this in Figure 4-5. This figure shows the minimum value of the x-axis of the accelerometer over datasets 'Driving_Long' and 'Walking_Long', described in Chapter 5. Here, we can see that the threshold exists at 0.41. Thus any datapoint that has a minimum accelerometer x-axis value greater than this threshold will be classified as a walking point and any point that has a minimum accelerometer x-axis value less than this threshold will be classified as a driving point.

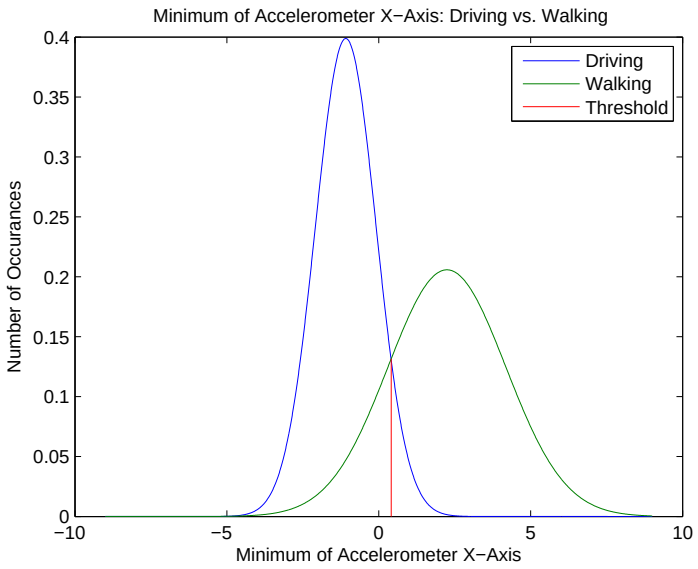


Figure 4-5: Minimum of Accelerometer X-Axis: Walking vs. Driving

## 4.4 Final Classifier Development

Once these classifiers were developed and trained, the AdaBoost algorithm, described in Chapter 2, was implemented in MATLAB and applied to these classifiers over the raw accelerometer data, the raw orientation data, the raw magnetic sensor data, and the speed data.

## 4.5 Implementation

The final classifier application is based on the application developed in the data collection phase of the project, with one change: the classifier that the AdaBoost algorithm produced is now integrated. When the classifier detects a change in the current mode of transportation, it automatically updates this value in the application. The user can still manually change the mode of transportation listings, both actively and retroactively, if the application has made an incorrect prediction. These discrepancies are logged with the expectation that they can be used to recalibrate the classifier.

# Chapter 5

# Results and Conclusion

Here we present our datasets and results. We then discuss the meaning of these results and suggest further research.

## 5.1 Datasets

Four datasets were collected and used for analysis. They consist of two sets of driving data and two sets of walking data. There is a long and short version of each (Table 5.1). Details of these datasets are presented in Table 5.2.

| Dataset Name | Number of Samples | Filesize |
|---|---|---|
| Driving_Short | 65,238 | 13MB |
| Driving_Long | 188,478 | 37MB |
| Walking_Short | 12,927 | 3MB |
| Walking_Long | 23,815 | 5MB |

Table 5.1: Dataset Sizes

## 5.2 Results

We begin by giving the thresholds used for each individual Naive Bayes classifier. We then examine the accuracy of each classifier and see how they compare. Finally, we examine the results given by the final boosted classifier.

### 5.2.1 Naive Bayes Results

The classifiers described in Chapter 4 were used to classify the data. The training data, 'Driving_Short' and 'Walking_Short', were used to generate a threshold for each sensor/classifier pair (Table 5.3). These thresholds were then used on non-training data, 'Driving_Long' and 'Walking_Long', to generate the accuracy of each classifier over both the training data (Table 5.4) and the non-training data (Table 5.5). Due to the inaccurate nature of the location data, some classifiers were ignored with respect to speed because they caused an overflow in the system.

One feature to notice in these results is similarity in the accuracies produced by the Variance, Standard Deviation, Range, Differences, Angular Velocity, and Zero Crossings in the Orientation and Magnetic Sensors. This is due to that fact that, during training data collection, the phone laid on the seat of the car and was not disturbed. This caused the values at the orientation and magnetic sensors to stay quite constant, giving the distribution of these values a very small variance. As all of the mentioned classifiers measure some aspect of the variance of the value, they also all had a narrow distribution of values centered around zero. In contrast, the walking data had a wider distribution of the values at these sensors. Thus, the accuracy percentages represent the portion of the time that the phone was still while the user was driving or in motion while the user was walking. This explains the correlation of these accuracies.

### 5.2.2 Boosting Results

The thresholds generated by the Naive Bayes classifiers were then used in the AdaBoost algorithm, described in Chapter 2, to produce the strong classifier described in Table 5.6. The final strong classifier is determined by the sign of the sum of the products of the classifiers and their respective coeffieicnts: $H(x) = sign(\sum c_i * h_i(x))$.

As we can see, this classifier converges quickly. This classifier had an accuracy of 92.13% on training data and of 81.06% on non-training data. We can see that this is an improvement over our best bayesian classifiers which had accuracies of 90.65% on

| Name | Duration | Channel | Frequency (Hz) | Total Samples |
|---|---|---|---|---|
| Driving_Short | 0:07:34 | Accelerometer | 123.5 | 56,065 |
| | | Orientation | 9.8 | 4,449 |
| | | Magnetic | 9.6 | 4,349 |
| | | GPS | 0.9 | 422 |
| Driving_Long | 0:22:01 | Accelerometer | 122.3 | 161,585 |
| | | Orientation | 9.8 | 12,975 |
| | | Magnetic | 9.6 | 12,736 |
| | | GPS | 1.0 | 1,272 |
| Walking_Short | 0:08:01 | Accelerometer | 20.3 | 9,766 |
| | | Orientation | 2.9 | 1,375 |
| | | Magnetic | 2.8 | 1,368 |
| | | GPS | 0.8 | 404 |
| Walking_Long | 0:18:29 | Accelerometer | 19.3 | 21,441 |
| | | Orientation | 0.6 | 670 |
| | | Magnetic | 0.6 | 649 |
| | | GPS | 0.5 | 585 |

Table 5.2: Summary of Datasets

| | **Acceleration** | | | **Orientation** | | | **Magnetic** | | | **Speed** |
|---|---|---|---|---|---|---|---|---|---|---|
| **Classifier** | **X** | **Y** | **Z** | **X** | **Y** | **Z** | **X** | **Y** | **Z** | |
| Mean | 0.3 | 1.8 | 9.2 | 1.9 | -11.1 | 128.2 | -3.7 | -6.6 | 16.0 | 0.9 |
| Median | 0.3 | 1.8 | 9.2 | 2.0 | -11.1 | 128.3 | -3.7 | -6.6 | 17.0 | 0.9 |
| Variance | 0.3 | 0.2 | 1.5 | 5.6 | 4.8 | 339.6 | 1.4 | 1.9 | 2.0 | |
| Standard Deviation | 0.3 | 0.2 | 0.8 | 1.3 | 1.1 | 3.9 | 0.6 | 0.6 | 0.6 | 0.0 |
| Minimum | -0.1 | 1.4 | 7.1 | 0.2 | -14.5 | 122.6 | -4.7 | -7.5 | 14.8 | 2.0 |
| Maximum | 1.6 | 3.0 | 10.5 | 5.6 | -9.0 | 133.6 | -7.5 | -5.7 | 18.2 | 2.0 |
| Range | 1.6 | 1.1 | 3.5 | 4.4 | 3.6 | 11.0 | 1.9 | 1.8 | 1.8 | 0.3 |
| Differences | 2.2 | 1.4 | 4.3 | 1.6 | 1.0 | 1.6 | 0.4 | 0.3 | 0.3 | 0.13 |
| Angular Velocity | 0.4 | 0.4 | 0.2 | 0.2 | 0.2 | 0.1 | 0.0 | 0.1 | 0.1 | |
| Zero Crossings | 60.4 | 57.1 | 0.1 | 60.9 | 45.0 | 31.1 | 47.6 | 76.7 | 41.9 | |
| Signal Vector Magnitude | 131.6 | | | 10.0 | | | 32.2 | | | |

Table 5.3: Thresholds of Naive Bayes Classifiers

| | Acceleration | | | Orientation | | | Magnetic | | | Speed |
|---|---|---|---|---|---|---|---|---|---|---|
| **Classifier** | **X** | **Y** | **Z** | **X** | **Y** | **Z** | **X** | **Y** | **Z** | |
| Mean | 76 | 88 | 86 | 88 | 86 | 76 | 82 | 58 | 89 | 83 |
| Median | 88 | 86 | 88 | 88 | 86 | 76 | 82 | 58 | 89 | 83 |
| Variance | 79 | 81 | 63 | 64 | 79 | 51 | 81 | 81 | 81 | 51 |
| Standard Deviation | 72 | 66 | 80 | 79 | 67 | 55 | 59 | 76 | 57 | 83 |
| Minimum | 89 | 87 | 64 | 88 | 86 | 75 | 82 | 58 | 89 | 65 |
| Maximum | 87 | 86 | 90 | 87 | 86 | 76 | 81 | 59 | 89 | 83 |
| Range | 73 | 72 | 81 | 80 | 68 | 56 | 76 | 59 | 58 | 83 |
| Differences | 81 | 68 | 89 | 77 | 69 | 59 | 62 | 63 | 64 | 83 |
| Angular Velocity | 87 | 87 | 78 | 59 | 64 | 77 | 84 | 85 | 85 | |
| Zero Crossings | 76 | 76 | 88 | 73 | 67 | 62 | 68 | 77 | 66 | 90 |
| Signal Vector Magnitude | 75 | | | 79 | | | 82 | | | |

Table 5.4: Accuracies(%) of Naive Bayes Classifiers on Training Data

| | Acceleration | | | Orientation | | | Magnetic | | | Speed |
|---|---|---|---|---|---|---|---|---|---|---|
| **Classifier** | **X** | **Y** | **Z** | **X** | **Y** | **Z** | **X** | **Y** | **Z** | |
| Mean | 91 | 82 | 86 | 91 | 81 | 80 | 68 | 56 | 53 | 11 |
| Median | 92 | 82 | 86 | 91 | 82 | 80 | 68 | 56 | 53 | 11 |
| Variance | 87 | 83 | 50 | 50 | 88 | 50 | 0 | 0 | 0 | |
| Standard Deviation | 56 | 66 | 53 | 51 | 53 | 50 | 89 | 50 | 50 | |
| Minimum | 90 | 77 | 50 | 90 | 87 | 80 | 65 | 56 | 56 | 11 |
| Maximum | 91 | 87 | 58 | 93 | 74 | 80 | 61 | 56 | 52 | 11 |
| Range | 51 | 56 | 50 | 50 | 52 | 50 | 89 | 50 | 52 | 11 |
| Differences | 63 | 63 | 67 | 57 | 63 | 63 | 83 | 16 | 83 | 69 |
| Angular Velocity | 69 | 65 | 56 | 53 | 53 | 57 | 53 | 66 | 63 | |
| Zero Crossings | 55 | 51 | 66 | 58 | 43 | 43 | 43 | 56 | 43 | |
| Signal Vector Magnitude | 56 | | | 58 | | | 53 | | | |

Table 5.5: Accuracies(%) of Naive Bayes Classifiers on Non-training Data

| Coefficient, $c_i$ | Classifier, $h_i(x)$ |
|---|---|
| 0.2993 | Maximum of the Accelerometer's Z-Axis |
| 0.2432 | Median of the Orientation Sensor's X-Axis |
| 0.1704 | Differences of the Magnetic Sensor's Y-Axis |
| 0.1531 | Minimum of the Accelerometer's X-Axis |
| 0.1340 | Minimum of the Accelerometer's Z-Axis |

Table 5.6: Strong Classifier

the training. While some of the bayesian classifiers perform better than the boosted classifier on the non-training data, our boosted classifier will produces accurate results more consistently. Additionally, our boosted classifier could be trained on a larger set of data, which would allow it to perform better. Thus, we see that applying boosting was able to increase the accuracy of our results.

Starting with the raw data, it took 134.1 seconds to train the weak classifiers. Once this was completed, the boosting loop is run 5 times and takes an average of 136.7 seconds to complete each loop. After the fifth loop, none of the classifiers can obtain more than 50% accuracy on the data with the current weights.

## 5.3 Conclusion

As is shown by the above results, implementing a boosting algorithm over a set of classifiers can generate a classifier that is stronger than any of its components. This classifier is not difficult to generate and, once generated, can easily be implemented on a mobile device. Although it has not been widely used in context recognition problems, boosting could allow context recognition applications to produce results with improved accuracy. Additionally, these methods could be further used in Mode of Transportation recognition to generate a more accurate prediction of a user's current MOT. These improved accuracies allow us to learn more from and better react to the data collected about the world around us.

## 5.4 Further Work

The results presented in this paper were only used to classify data collected from a user in an automobile and a user walking. Additionally, these data were only collected by a single user. These results could be expanded by incorporating more modes of transportation and by collecting data from more users. Additionally, these methods to be expanded to other mobile platforms. The data collection application and the MATLAB code provided for analysis can be used for this with little or no modification.

# Appendix A

# Code

Here we discuss the code developed for this thesis and give instructions for the reader to obtain the source.

## A.1 Summary of Code

The code developed in this thesis consists of three main components:

**MOT Data Collector Application** An Android application that allows the user to collect the data necessary for this thesis

**MOT Classifier Developer** A suite of MATLAB files that allows the user to analyze the collected data and generate the naïve bayes classifiers and the boosted naïve bayes classifier described in this thesis

**MOT Determination Application** An Android application that takes the boosted naïve bayes classifier generated in this thesis and uses it to predict a user's MOT based on sensor data

In addition, there is one python script, 'MOT Data Parser.py', included that parses the data collected by the 'MOT Data Collector Application' into files which can be used by the 'MOT Classifier Developer'.

## A.2 Instructions

All code is stored in the SVN repository of the RVSN group at CSAIL. You can check it out using your CSAIL identity. In the command line, run the following command:

```
svn co svn+ssh://<YOURUSERNAME>@svn.csail.mit.edu/afs/csail/group/
                rvsn/Repository/walkthru/OIL/trunk/code/MOT
```

A new directory will then be created in your current working directory. Under this directory, there are three subdirectories. A rough sketch of the structure and the locations of the necessary README files are given below.

```
MOT
|-Android
  |-MOT Data Collector Application
    |-README
  |-MOT Determination Application
    |-README
|-MATLAB
  |-README
  |-MOT Data Parser.py
|-Data
```

The first, 'Android', contains the source for two Android applications. These are the 'MOT Data Collector Application' and the 'MOT Determination Application'. The main directory of each of these projects contains a README file which describes how to build and run the code. The second subdirectory, 'MATLAB', contains all of the MATLAB source used for this project. There is also a README file in this directory explaining how to use this application. In this directory, there is also the 'MOT Data Parser.py'. Instructions on the use of this file are included in the file's comments. The final subdirectory, 'Data', contains the datasets described in this thesis.

Please direct any questions to maria.frendberg@gmail.com.

# Bibliography


[1] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway, P. Klasnja, K. Koscher, J.A. Landay, J. Lester, D. Wyatt, and D. Haehnel. The mobile sensing platform: An embedded activity recognition system. *Pervasive Computing, IEEE*, 7(2):32 –41, April-June 2008.

[2] Roger Entner. Smartphones to overtake feature phones in U.S. by 2011. http://blog.nielsen.com/nielsenwire/consumer/ smartphones-to-overtake-feature-phones-in-u-s-by-2011/, 2010.

[3] Davide Figo, Pedro C. Diniz, Diogo R. Ferreira, and João M. Cardoso. Preprocessing techniques for context recognition from accelerometer data. *Personal Ubiquitous Comput.*, 14:645–662, October 2010.

[4] Jonathan Lester, Phil Hurvitz, Rohit Chaudhri, Carl Hartung, and Gaetano Borriello. Mobilesense - sensing modes of transportation in studies of the build environment. UrbanSense08, pages 46–50, November 2008.

[5] Vincenzo Manzoni, Diego Manilo, Kristian Kloeckl, and Carlo Ratti. Transportation mode identication and real-time $CO_2$ emission estimation using smartphones. Technical report, SENSEable City Lab, Massachusetts Institute of Technology.

[6] MATLAB: Product Description. http://www.mathworks.com/products/matlab/description1.html, May 2011.

[7] OnStar: Automation Crash Response. http://www.onstar.com/web/portal/acr, May 2011.

[8] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3*, pages 1541–1546. AAAI Press, 2005.

[9] Sasank Reddy, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. Determining transportation mode on mobile phones. In *Proceedings of the 2008 12th IEEE International Symposium on Wearable Computers*, pages 25–28, Washington, DC, USA, 2008. IEEE Computer Society.

[10] Robert E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[11] H. Wang, F. Calabrese, G. DiLorenzo, and C. Ratti. Transportation mode inference from anonymized and aggregated mobile phone call detail records. In *IEEE International Conference on Intelligent Transportation Systems*, Madeira Island, Portugal, September 2010.

[12] Shuangquan Wang, Canfeng Chen, and Jian Ma. Accelerometer based transportation mode recognition on mobile phones. In *APWCS'10*, pages 44–46, 2010.

[13] Wikipedia. Haversine formula — wikipedia, the free encyclopedia, 2011. [Online; accessed 17-May-2011].

[14] Wikipedia. Machine learning — wikipedia, the free encyclopedia, 2011. [Online; accessed 17-May-2011].

[15] Jhun Y. Yang, Jeen S. Wang, and Yen P. Chen. Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern Recogn. Lett.*, 29(16):2213–2220, 2008.