4-2014

Modeling Mobile User Behavior for Anomaly Detection

Senaka Buthpitiya
Carnegie Mellon University, senaka.buthpitiya@sv.cmu.edu

Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE **Model Mobile User Behavior for Anomaly Detection**

PRESENTED BY **Senaka W. Buthpitiya**

ACCEPTED BY THE DEPARTMENT OF

Electrical and Computer Engineering

____Maritn Gris_____ ____4/15/14_____
ADVISOR, MAJOR PROFESSOR DATE

___Jelena Kovacevic_____ ___4/15/14_____
DEPARTMENT HEAD DATE

APPROVED BY THE COLLEGE COUNCIL

_____Vijayakumar Bhagavatula_____ _____5/1/14_____
DEAN DATE

Modeling Mobile User Behavior for Anomaly Detection

CMU-PDL-XX-XXX

Submitted in partial fulfillment of the requirements for
the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Senaka W. Buthpitiya

B.Sc., Computer Engineering, University of Peradeniya
M.Sc., Electrical and Computer Engineering, Carnegie Mellon University

Thesis Committee:
Prof. Martin Griss, Chair
Prof. Anind K. Dey
Prof. Ying Zhang
Dr. John Krumm

Department of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

March 2014

Keywords: Behavior Modeling, Anomaly Detection, Context, Context-Modeling

This thesis is dedicated to my parents.

Ammi for teaching me patience and persevere, through thick and thin.

Thaththi for instilling in me a compulsive need for routine and order.

Acknowledgments

It really does take a village to raise a child - I could never have completed this, particularly the last push towards finishing up, without the support and encouragement of a lot of people.

Foremost, I would like to express my deepest gratitude to my advisors Professor Martin Griss and Professor Anind K. Dey for being tremendous mentors, for encouraging my research, and for allowing me to grow as a researcher. Their advice on both research as well as on my career have been invaluable. I would also like to thank the rest of my thesis committee members for their helpful feedback and valuable discussions. Professor Ying Joy Zhang helped immensely in developing the research ideas for this thesis very early on and steadfastly provided support, which was greatly needed and deeply appreciated. Dr. John Krumm provided steering at critical points to avoid quite a few pit-falls.

I gratefully acknowledge the financial support from Ericsson Research Silicon Valley for the first half of my graduate studies at Carnegie Mellon University. Professor Ian Lane and Dr. Jike Chong for the opportunity to contribute to some very interesting research work. Professor Pei Zhang and Professor Patrick Tague for their guidance and advice early in the program. I am grateful to group mates Faisal Luqman, Heng-Tze Cheng, and Feng-Tso Sun for their support and encouragement. I thank my cubicle-mates Rahul Rajan and Aveek Purohit for always being good sounding boards for ideas and for tolerating my constant complaining and distractions. Vishwanath Raman, Jung-Suk Kim, and Avneesh Saluja were always their to provide a breath of fresh air when work was just a tad too stressful. I also thank Yu-Seung Kim, Bruce DeBruhl, Zheng Sun, Chihiro Suga, Le Nguyen, and the rest of my fellow doctoral students - those who have moved on, those in the quagmire, and those just beginning - for making my time at Carnegie Mellon memorable. I am

also very grateful to all the staff at Carnegie Mellon who made life so much easier for us.

I would like to show my gratitude to Dr. Manjula Sandirigama, Dr. S. Devapriya Dewasurendra, Dr. Roshan Ragel, and the rest of the faculty of the Department of Computer Engineering at the University of Peradeniya for their tremendous impact on my decision to pursue my graduate studies. And a very special thanks to my teachers and friends at my first alma-mater, Trinity College Kandy, for a positive imprint which echoes within me to this day - *We've lost the old delight of her, We keep her honour yet.*

I thank all of my friends for being there for me during the ups and downs for graduate school life. I appreciate and am grateful to my best friend, Warshini, for her understanding, constant encouragement and patient support. Finally, I would like to dedicate this work to my parents for all their hard work and sacrifices throughout the years to help me.

Abstract

As ubiquitous computing (ubicomp) technologies reach maturity, smart phones and context-based services are gaining mainstream popularity. A smart phone accompanies its user throughout (nearly) all aspects of his life, becoming an indispensable assistant the busy user relies on to help navigate his life, using map applications to navigate the physical world, email and instant messaging applications to keep in touch, media player applications to be entertained, etc. As a smart phone is capable of sensing the physical and virtual context of the user with an array of "hard" sensors (e.g., GPS, accelerometer) and "soft" sensors (e.g., email, social network, calendar), it is well-equipped to tailor the assistance it provides to the user. Over the life of a smart phone, it is entrusted with an enormous amount of personal information, everything from context-information sensed by the phone to contact lists to call-logs to passwords.

Based on this rich set of information it is possible to model the behavior of the user, and use the models to detect anomalies (i.e., significant variations) in the user's behavior. Anomaly detection capabilities enable a variety of application domains such as device theft detection, improved authentication mechanisms, impersonation prevention, physical emergency detection, remote elder-care monitoring, and other proactive services.

There has been extensive prior research on anomaly detection in various application domain areas (e.g., fraud detection, intrusion detection). Yet these approaches cannot be used in ubicomp environments as 1) they are very application-specific and not versatile enough to learn complex day to day behavior of users, 2) they work with a very small number of information sources with a relatively uniform stream of information (unlike sensor data from mobile devices), and 3) most approaches require labeled or semi-labeled data about anomalies (in ubicomp environments, it

is very costly to create labeled datasets). Existing work in the field of anomaly detection in ubicomp environments is quite sparse. Most of the existing work focuses on using a single sensor information stream (GPS in most cases) to detect anomalies in the user's behavior. However there exists a somewhat richer vein of prior work in modeling user behavior with the goal of behavior prediction; this is again limited mostly to a single sensor stream or single type of prediction (mostly location).

This dissertation presents the notion of modeling mobile user behavior as an collection of models each capturing an aspect of the user's behavior such as indoor mobility, typing patterns, calling patterns. A novel mechanism is developed for combining these models (*i.e.,* CobLE), which operate on asynchronous information sources from the mobile device, taking into consider how well each model is estimated to perform in the current context. These ideas are concretely implemented in an extensible framework, McFAD. Evaluations carried out using real-world datasets on this framework in contrast to prior work show that the framework for detecting anomalous behavior, 1) vastly reduces the training data requirement, 2) increases coverage, and 3) dramatically increases performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As ubiquitous computing (ubicomp) technologies reach maturity, smart phones and context-aware services are gaining mainstream popularity. A smart phone accompanies its user throughout (nearly) all aspects of the user's life, becoming an indispensable assistant the busy user relies on to help navigate their life, using map applications to navigate the physical world, email and instant messaging applications to keep in touch, media player applications to be entertained, etc. As a smart phone becomes more capable of sensing the physical and virtual context of the user with an array of "hard" sensors (e.g., GPS, accelerometer) and "soft" sensors (e.g., email, social network, calendar), it becomes better equipped to tailor the assistance it provides to the user. As a smart phone is used over a period of time, the user will trust the device with an enormous amount of personal information, everything from relatively static information such as passwords and contact lists to streams of information such as context-information sensed by the phone and call-logs.

Based on this rich set of information it is possible to model the behavior of the user, and use the models to detect anomalies (i.e., significant variations) in the user's behavior. Variations in data streams could be caused by noise, novel behavior or anomalous behavior. The difference between noise and the other variations is that noise is a phenomenon in data which is not of interest to a human-analyst and acts as a hindrance to data analysis. The distinction between novel

patterns and anomalies is that the novel patterns are typically incorporated into the normal model over time [11]. The ability to recognize and distinguish between novel and anomalous behavior enables a variety of application domains such as device theft detection, improved authentication mechanisms, impersonation prevention, physical emergency detection, remote elder-care monitoring, and other proactive applications. Following is this dissertation's definition of anomalous behavior.

> *An anomaly in a user's behavior is a significant variation from the user regular routine for a sustained period of time. The significance and sustaining period of a variation is dependent on the application for which anomalies are being detected.*

There has been extensive research in anomaly detection in a wide variety of application domains such as fraud detection for credit cards and insurance, intrusion detection for cyber-security, fault detection in critical systems, and security surveillance. Over time, a variety of anomaly detection techniques have been developed in several research communities. Many of these techniques are application domain dependent [11] while a lesser number of more generic approaches have also been developed [37]. These approaches cannot be used in ubicomp environments because,

- they are very application-specific and not versatile enough to learn complex day-to-day behavior of mobile users,
- they are designed to work with a very small number of information sources with a relatively uniform stream of information (unlike soft- and hard-sensors from mobile computing environments), and
- most approaches require labeled or semi-labeled data of anomalies (in ubicomp environments, it is expensive, time-consuming and/or nearly impossible to get labeled anomaly data).

Existing work in the field of anomaly detection in ubicomp environments is quite sparse. Most of the existing work focuses on using a single sensor information stream (GPS in most cases) to detect anomalies in the user's behavior. However there exists a somewhat richer vein of prior

work in modeling user behavior with the goal of behavior prediction, this is again limited mostly to a single sensor stream or single type of prediction (mostly location) [2, 43, 77].

1.1 Challenges in Mobile User Behavior Anomaly Detection

In previous work on using multiple information sources to model user behavior, for anomaly detection or behavior prediction, the style of modeling has been to create a single holistic model to capture all facets of the user's behavior that are of interest [55, 64]. That is, to create a single model where preprocessed information from each sensor is fed directly into a single model. The model is trained (*i.e.,* parameters of the model are estimated) using logged sensor data of the user (*i.e.,* training data). The biggest hurdles faced in creating a holistic model of a mobile user's behavior for anomaly detection can be broadly categorized as:

1. granularities of user routines,
2. variety and number of information sources,
3. training data limitations, and
4. continuous evolution of user routines.

These three hurdles are not disconnected from each other, but each plays a role in helping to create or increase the other. Each of these are described in more detail in the following subsections.

1.1.1 Granularities of User Routines

A user has very complicated behavior patterns at various granularities of time - and different applications are interested in anomalies at different granularities of behavior. For example, a user's word usage when instant messaging is a pattern that operates on a completely different timescale and granularity from the same user's patterns of mobility. Even within the same type of patterns there are a large number of possible levels of granularities. For example, a user's mobility patterns when the user is walking happens on the scale of seconds to minutes, while the

same user's mobility patterns when the user is driving happens on the scale of minutes to hours.

It is not possible to restrict the modeling of user behavior to one (or a few) of these levels of granularity since even a single application of anomaly detection, such as device theft detection, would be interested anomalies detected at any of these levels of granularity.

1.1.2 Variety and Number of Information Sources

A mobile device has multiple sensors dedicated to sensing its and the user's surroundings and situation (*e.g.,* GPS, accelerometer, gyroscope). These sensors on a mobile device, which sense the physical world around the device and the user, are termed *hard-sensors* in this dissertation. A mobile device also possesses many more "sensors" which are based on the device's usage (*e.g.,* call-logs, application behavior) which provide hints as the user's (and of course the devices) surroundings and situation (*i.e.,* context). In this dissertation, such sensors logging data from virtual sources are, as a group, termed *soft-sensors*. In addition to hard- and soft-sensors a mobile device has multiple radios each providing information about other visible devices and/or base-stations, signal strengths, data-rates, etc. Radios on the mobile device providing information are, as a group, termed termed *radio-channel-sensors*. Together, this dissertation terms the dedicated and inadvertent "sensors" with the more apt collective name of *information sources*. Hard-sensors, soft-sensors, and radio-channel-sensors are also labeled as *hard information sources*, *soft information sources*, and *radio channel information sources* respectively (for a detailed description of information source types see Section 2.2).

The number of information sources available for user behavior modeling on a mobile computing platform are immense, also the best sources for modeling behavior are user dependent, which creates the problem of selecting the best set of sources to utilize for the task of anomaly detection. Using many of these sources leads to the problem described in the next subsection, namely that of training data.

1.1.3 Training Data Limitations

As more complex models (encompassing various levels of granularity described previously) are created using an increasing number of information sources available on a mobile device, the anomaly detection process in theory should become more reliable. The drawback of more complex models using many sources, and large volumes of data is that the training data requirements increase exponentially with the complexity and the number of information sources used. As a user's behavior routines are an ever evolving phenomena, gradually changing over time, it is not feasible to expect large amounts of training data for a user's routines at any given point in time.

A further complication is that it is a very difficult to collect positive examples of anomalous behavior of a user. For example, in the case of using anomaly detection for theft detection of a mobile device, it is possible to get many negative examples of anomalies by sensing the user's routine behavior, yet it is nearly impossible to collect positive examples of anomalies.

1.1.4 Continuous Evolution of User Routines

The evolution of a user's routine happens with both, gradual changes to routine (*e.g.,* the user gradually adjusting a meal time), and abrupt changes (*e.g.,* the first day of a new semester). Ideally an anomaly detection system should not detect either of these types of chance in routine as anomalies. In either case, an anomaly detection system needs the capability of differentiating novel behavior from anomalous behavior.

1.2 Thesis Statement

This dissertation validates the following hypothesis:

> *Detection of anomalous behavior in the routines of mobile users is best achieved by modeling different simple aspects of user behavior and combining the outputs of these models in a systematic manner.*

We define "aspect of user behavior" as follows:

> *An aspect of a user's behavior is a smaller component of the user behavior which has a capture-able (quantifiable) regularity, and where deviations from this regularity are indicators of anomalous behavior.*

This dissertation presents a holistic methodology for :

- modeling different aspects of mobile user behavior, locally on a mobile device based on the rich set of information sources present on the mobile device,
- evaluating the performance of models,
- creating an holistic model of mobile user behavior by combining the behavior aspect models in a sophisticated manner, and
- applying the combined models to detect anomalies (i.e., significant variations) in the user's behavior.

Specifically by modeling different aspects of mobile user behavior separately, creating multiple models of smaller parts of a user's routine should make information source selection, model complexity, and training data requirements more manageable. By modeling different behavior aspects, each model should be more robust against detection of evolving user behavior as anomalous. To create a more holistic model of the user's routine, while keeping the model complexity and data requirements low, this dissertation examines how best to combine the models of different aspects of the user's routine. Using this complete model (*i.e.,* ensemble of models), this dissertation presents an effective approach to perform anomaly detection on the user's routine.

This dissertation further contributes a Mobile Computing Framework for Anomaly Detection in User Behavior (or McFAD for short). McFAD was designed and implemented using this methodology to assess its feasibility and practicality.

Goals. The primary goals of this dissertation are:

- Improved accuracy in the detection of anomalous mobile user behavior.
- Improved coverage throughout the day for anomaly detection.

- Anomaly detection in the absence of labeled training data.
- Reduced training data requirement.

Non-goals. This research does not address the following:

- Determining the course of action in response to detected anomalous behavior.
- Determining the type of anomaly detected.

Assumptions. This research is performed under the following assumptions:

- A user's behavior routines are sufficiently stable to remain substantially unchanged over a span of a few (at most 4) weeks.
- It is not possible (or at the very least very expensive) to collect training data of real anomalies.
- During the model training phase, the data collected is from *normal* routine behavior of the user.
- For testing purposes, anomalous behavior (*i.e.,* significant variations) of one user is sufficiently different from the user's normal routine to mirror the routine of another user.

1.3 Thesis Map

This dissertation addresses the detection of anomalous behavior, caused by various reasons, in a mobile device user's routine. The anomaly detection mechanisms described are developed and tested using two real-world datasets. While mechanisms described here are implemented as an Android platform service for utilization and extension by application developers, it is not limited to this platform.

To address the challenges described in Section 1.1, the notion of breaking down a user's behavior patterns to aspects of behavior is developed. In Chapter 2, this concept is described in detail. Chapter 2 also discusses the various information sources available on a mobile device, and introduces the datasets and metrics used in the majority of experiments presented in this

dissertation.

Chapters 3, 4, and 5 present examples of modeling different aspects of a user's routine using hard sensor information sources, soft sensor information sources, and radio channel information, respectively, available on a mobile device. While Chapters 4 and 5 present two models each, along with an analysis of the models. Chapter 3 presents a single model but delves into a very detailed analysis.

Chapter 6 explores the idea of combining individual models, each modeling a separate aspect of a user's behavior. The results in Chapter 6 show that combining models provides better reliability in the anomaly detection process, while motivating the need for a more systematic method of combining the models.

Next Chapter 7 describes a novel approach for combining classifiers to form an ensemble of classifiers, used to combine models of aspects of user behavior. This chapter presents formal proofs on the workings of this combining approach. It also presents experimental results showing that this approach is best suited for combining models each modeling a different aspect of a user's behavior.

The following chapter discusses how the anomaly detection mechanisms, described in the previous chapters, are implemented on an Android platform as the McFAD framework. In Chapter 9, an evaluation of these anomaly detection mechanism are presented, contrasting them with the traditional idea of creating a single model for behavior modeling. Finally, Chapter 10 summarizes and identifies future research opportunities.

1.4 Contributions

This dissertation presents a methodology for adapting information streams available on a mobile device to model a user's routine to detect anomalous behavior. The contributions of this work are:

Concept of modeling behavior aspects : As opposed to the traditional idea of modeling

user behavior with a single monolithic model [55], the idea of modeling smaller aspects of a user's routine in separate models, and then finally combining these models to gain a holistic model of the user's behavior routines is presented. Modeling simpler aspects of a user's routine allows each model to be trained with limited data. Combining these aspect models to create a holistic model of a user's routine, prevents the training data requirement from exponentially increasing unlike with a monolithic model. Since the number of information sources used by a aspect-based model is generally far smaller than that of a monolithic model, the likelihood that at least one aspect model will have reliable information on all (or most) of its sources is far higher than that for a monolithic model.

An ensemble learning approach for combining classifiers operating on unreliable and asynchronous information sources : The ensemble learning approach is able to combine any type of classifier, and account for missing feature values (or sets) presented to some classifiers. The ensemble learning approach *learns* the areas of a feature space where each classifier excels or fails, and based on this knowledge and the number of missing features presented to a classifier, it dynamically weights the output of classifiers for aggregation. While this ensemble learning approach is presented as a method of combining aspect models of user behavior, it can be used to combine learners for other applications, especially when there is asynchronousness or unreliability in the features being provided to the classifiers.

An anomaly detection system for user behavior, which can be trained on purely positive examples of routine behavior : The models developed in this research can be trained by providing only positive examples of the user's routine behavior. This removes the need for synthetic anomalies to be created and presented to the model at training, making the data collection process to bootstrap a model for a new user extremely straightforward.

An extensible anomaly detection framework : The anomaly detection framework presented in this dissertation can be easily extended to incorporate new behavior aspect models. Due to the novel ensemble algorithm, new models can be incorporated without any changes be-

ing made to the existing models (nor requiring the existing models to be retrained). Since, new (or already modeled) aspects of behavior can be modeled and added for anomaly detection without additional training being performed, it allows new information sources to be integrated for anomaly detection simply by adding in models that use the new information sources. The implementation also identifies and addresses the engineering issues in implementing an extensible anomaly detection system on a mobile device.

A comparison of performance : The anomaly detection approach presented in this dissertation, based on behavior aspect models combined using the ensemble learning approach, is throughly evaluated and contrasted against the concept of using a monolithic learner. The evaluation address not only the performance of the different approaches, but looks also at the bootstrapping data requirements (*i.e.,* training data requirements) and evolution of performance over time.

Behavior aspect models : This dissertation presents multiple behavior aspect models using a variety of information sources available on a mobile device. The models focus on each using a single information source, which fall into the categories of hard, soft, or radio channel information sources. Each model is thoroughly evaluated using an evaluation methodology developed in this dissertation, in order to compare user anomaly detection approaches.

Chapter 2

Modeling Aspects of User Behavior

This chapter first introduces the notion of modeling aspects of behavior as an alternative approach to modeling behavior. Next is presented an overview of the data sources presently available on mobile devices and discusses aspects of user behavior that can be modeled using these information sources. A description of the datasets used in evaluations in this dissertation is also provided. A model is described to illustrate the notion behavior modeling as aspects of behavior models. The chapter concludes with a introduction of evaluation metrics used throughout this dissertation.

2.1 Aspects of Behavior

When modeling the behavior of individuals (or users), prior work usually employs a single model that encompasses the entire behavior of the user [7, 54, 55]; only when the explicit target of modeling was just one specific part of a user's behavior, such as mobility patterns [29, 47, 61], or gait modeling [19, 26], is a simpler model used. For the rest of this dissertation, a single model used learn or capture a user's entire behavior is termed a "monolithic model". The notable exception to this trend is Shi *et al.* [64] where the idea of combining three separate models is employed.

In this section we formalize this idea, *i.e.,* modeling behavior of a user as a collection of

models, where each model captures just a single aspect of the user's behavior. An "aspect of behavior" is defined as (reiterating from the previous chapter):

> *An aspect of a user's behavior is a smaller component of the user behavior which has a capture-able (quantifiable) regularity, and where deviations from this regularity are indicators of anomalous behavior.*

A user's mobility patterns (indoor and outdoor separately), activities at different times of the day, people seen through the course of the day, vocabulary usage, typing patterns, messaging patterns, social media posting patterns, and application usage on a mobile device are just a few examples from a extremely large set of behavior aspects. Then a user's behavior model could be built up by separately modeling a selection of these aspects, for example, the user's mobility, gait, and application usage patterns. It is worth noting that, in this approach, while a single model captures just a single aspect of the user's behavior patterns, multiple models may capture the same aspect of behavior (possibly based on different information sources).

The main advantages of modeling user behavior as a composition of smaller behavior aspect models, rather than using a monolithic model on fused data[1], are,

- **Reduced training data requirements :** Unlike monolithic models, each behavior aspect model operates on a (relatively) much smaller feature space. The "Curse of Dimensionality" problem introduced in [4] which directly applies to training of classifiers [24] dictates that increasing the feature space dimensions increases the training data requirements exponentially. Thus, splitting a set of information sources into groups to model separate aspects a of user's routine will require a linearly increasing amount of training data, while using the same information sources to create a single monolithic model exponentially increases the training data requirement.
- **Likelihood of data being available to detect anomalies :** The smaller behavior aspect models each use just a subset of the information sources used by a monolithic model.

Therefore in situations where one or more information sources are unavailable (*e.g.,* GPS signal is unavailable when the user is indoors), the smaller models in combination have a much better likelihood than the monolithic model that at least one of the smaller models will have a full feature set of information to make an anomaly detection decision.

- **Opportunity for greater accuracy :** While at least one of the smaller models having a full feature set to make a decision is likely, the overall accuracy of the combined models over the monolithic model in this situation depends on the combining algorithm. It should combine the decisions of the behavior aspect models properly weighting the decision by the ability of each of the models in the current situation. In mobile user behavior modeling, it is extremely common to have more than one information source unavailable at any given point in time. Therefore, if a proper combining approach is employed, the behavior aspect models should provide better accuracy at anomaly detection than a monolithic model, different models compensating for each other.
- **Opportunity for easier extensibility :** If a model combination approach which properly isolates individual models is employed, it is possible to add new models (modeling new or already modeled aspects of behavior) in to the system while involving minimal or even zero overhead in terms of retraining existing models.

2.2 Information Sources

This section introduces the information sources available on a typical consumer mobile device and presents some of the aspects of behavior that may be modeled using them. The information streams available from a mobile phone can be broadly categorized into three types based on how they are generated and sensed. This division has somewhat blurred boundaries, but helps in devising or identifying aspects of behavior to models. The categories are as follows,

[1] See Section 7.1 for a detailed description on different fusion techniques.

1. **Hard sensor information sources :** The first category is sensor data from sensing the physical world around the user and device. The data is either about the environment or about how the phone is being used. Examples of this type of sensor streams and a few aspects which can be modeled are presented in Table 2.1.

Table 2.1: Hard-sensor information sources and sample aspects of user routines that can be modeled from them.

Information Source	**Aspects to model**
GPS	Sequences of locations
Accelerometer	Gait detection models Activity and location relationship
Gyroscope	Angles at which the phone is held when in use
Compass	Orientation and heading
Light sensors	Ambient lighting profiles for locations
Microphone	Acoustic profiles of routine
Camera	Floor and ceiling patters for locations
Proximity	How phone is held when in a call

2. **Soft sensor information sources :** This type of information source is generated by software sensors. The information sources can be further divided into 3 sub-categories as,

   (a) soft sensor information about the state of the mobile device (e.g., screen state, ringer states),

   (b) communication histories of the user, for example, the call logs and SMSs sent and received by the user,

   (c) information generated when the user interacts with the mobile device (e.g., key press durations)

   Examples of these three sub-type of soft sensor information sources and few aspect which can be modeled are presented in Table 2.2.

3. **Radio channel information sources :** This type of information sources consist of communication channel information such as signal strengths, visible end-points, etc. Radio channel information sources unlike hard sensor information sources carry some informa-

Table 2.2: Soft-sensor information sources and sample aspects of user routines that can be modeled from them.

Information Source	**Aspects to model**
Mobile device state information	
Battery Level	Battery drain in relation to time/location to detect abnormally high/low usage
Memory Usage	Detect abnormally high memory usage indicative of malware infections
Process List	Process list in relation to time Process list in relation to location
Communication histories	
Call Log	How often are people outside the contact list called
Short Messages	Incoming - model each contacts vocabulary usage Outgoing - model user's vocabulary usage Delay and action taken for each contact's incoming email
Browser History	Commonly visited websites and time spent
Email	Incoming - model each contacts vocabulary usage Outgoing - model user's vocabulary usage Delay and action taken for each contact's incoming email
Calendar	Using location to detect if appointments are kept
Social Network	Time-of-day based post count Location based post count
User interactions with device	
Key-press information	Time between key-presses for known words
User's Voice	Voice fingerprinting

tion about the virtual environment (such as base station IDs and data rates), while at the same time unlike soft sensor information sources carry some information about the physical environment (such as signal strengths). Because these information sources carry both hard and sort information, radio channel information sources have been separated into a category of its own. Examples of this type of information streams and few aspect which can be modeled are presented in Table 2.3.

In the next section, the datasets used for evaluations in this dissertation are described.

Table 2.3: Radio channel information sources and sample aspects of user routines that can be modeled from them.

Information Source	Aspects to model
Bluetooth	Calendar events (and proximity of people) to Bluetooth IDs
WiFi	Base station sequences throughout the day (similar to GPS but for known indoor environments)
GSM Cell ID	Cell ID sequences throughout the day (similar to GPS but works even indoors)

2.3 Experimental Data

The evaluations presented in this dissertation (aside from the experiments presented in Chapter 7) are conducted using two datasets. The first dataset is a location-only dataset. The second dataset logs data from a variety of sensor streams on mobile phones.

2.3.1 Dataset I

The first dataset consists of GPS-trace data from 10 users living in the Pittsburgh area, including a mix of students, blue-collar and white-collar workers. Each user carried a QSTARZ BT-Q1000P GPS Data Logger at all times for 4 weeks (with two users using the data-logger for only two weeks). The GPS receiver had a low logging rate of approximately 8×10^{-3} Hz (approximately once every 2 minutes) and the battery lasted about 30 hours for each charge.

The visible satellite count is also logged by the GPS data logger. This satellite count was used to filter out unreliable data points, which have less than four satellites visible, from the final dataset. Even though the Data Logger was recording coordinates every two minutes, due to loss of GPS signals (e.g., in buildings and in tunnels) and loss of power, the dataset cannot be assumed to have a constant capture rate (*i.e.,* it has varying granularity).

2.3.2 Dataset II

The second dataset consists of a variety of soft sensor information sources, hard sensor information sources, and radio channel information recorded from mobile phones. The dataset has data from 30 users living in the Pittsburgh area, and once again includes a mix of students, blue-collar and white-collar workers. Each user carried Android mobile phones with the data logging software installed for a period of 3 months. The data logger stores the data locally on the device until the user uploads the data at the end of each day.

The full list of information logged by the application and descriptions of each source are shown in Table 2.4. Since each data source has varying update rates and power costs in sensing, the rate at which each source is sensed and logged varies. The logging rates were adjusted to get usable data while allowing the phone battery to last approximately 24 hours.

The next section provides an in-depth introduction into the evaluation metrics and charts used throughout this dissertation.

2.4 Evaluation Metrics

2.4.1 Receiver Operator Characteristic (ROC) Curves

A receiver operating characteristic (ROC) curve is a graphical illustration of the performance variation of a binary classifier with the adjustment of an internal parameter. In most situations the adjusted internal parameter is the classification threshold. The curve is generated by adjusting internal parameter and recording the true-positive rate (*a.k.a.,* sensitivity, recall - see Equation 2.1) on the y-axis and the false-positive rate (*a.k.a.,* fall-out rate - see Equation 2.2) on the x-axis. The result is a 2-dimensional scatter plot, which with sufficient points can be seen as a

Table 2.4: List of information sources logged in Dataset II

Information Source	Details
Accelerometer	Logged approximately at 10Hz
Applications	Logs applications as they are installed, updated or removed
Battery	Battery charge level information as well information on charger connection or disconnection
Bluetooth	Bluetooth MAC addresses seen by the device and whether or not the device was paired with them
Calendar	Log of calendar entry additions, editing or removal
Call	Log of calls made, received and missed along with the telephone number and contact name
Carrier	The state of the call and data lines, network connection type
Locations	GPS data, some points are recorded with street addresses
Network	Data connection type and IP address
Screen	Log of when the screen was turned on or off
Settings	Log of when phone settings were changed and the new values
SMS	Numbers and sent/received value of SMS
System Activity	Operating system activities (e.g., starting up a process, receiving a message)
Weather	Weather information for current location and the weather station providing the data
WiFi Neighbors	WiFi MAC addresses and signal strengths seen by the device

single curve.

$$\begin{aligned}\text{True Positive Rate (TPR)} &= \frac{\text{Number of true positives}}{\text{Total number of positive samples}} \\ &= \frac{\text{Number of true positives (TP)}}{\text{Number of true positives (TP)} + \text{Number of false negatives (FN)}}\end{aligned} \tag{2.1}$$

$$\begin{aligned}\text{False Positive Rate (FPR)} &= \frac{\text{Number of false positives}}{\text{Total number of negative samples}} \\ &= \frac{\text{Number of false positives (FP)}}{\text{Number of false positives (FP)} + \text{Number of true negatives (TN)}}\end{aligned} \tag{2.2}$$

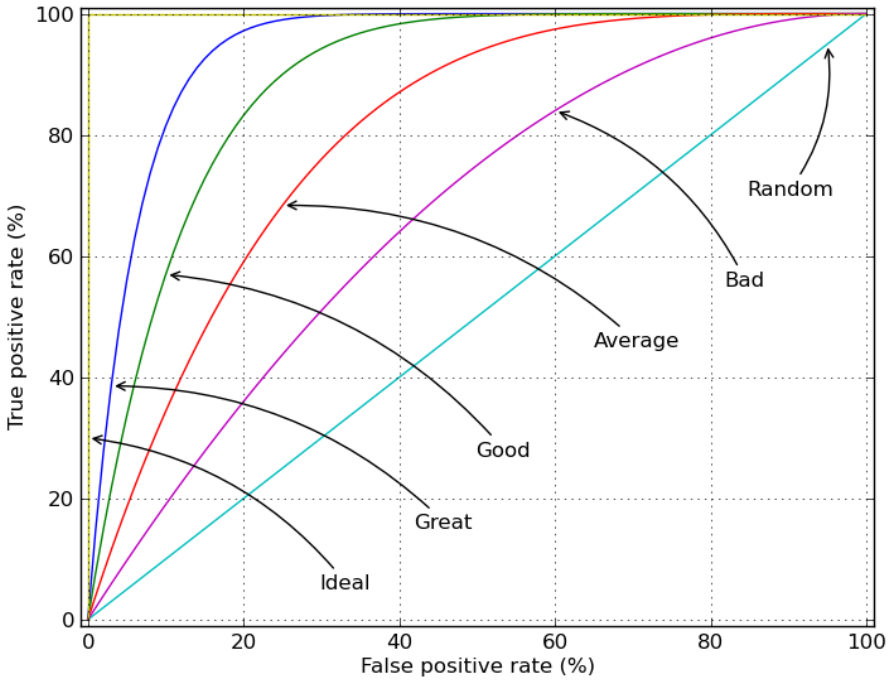


Figure 2.1: How to interpret receiver operator characteristic (ROC) curves.

For anomaly detection, the y-axis shows correct anomaly detections (as a fraction of the total number of anomalies present in the testing data), and the x-axis shows incorrect detections (as a fraction of non-anomalous test cases in the testing dataset). Points along the diagonal mean the ratio is even, and the model is performing no better than a coin flip - for every one correct detection, there is an incorrect detection. The upper left corner is a perfect anomaly detector, detecting all anomalies perfectly without any false detections. Each point on a ROC curve corresponds to a certain detection threshold. In Figure 2.1 six ROC curves are shown. The performance depicted by each of the six curves vary from random guessing with uniform probability (depicted by the blue turquoise diagonal ROC curve) to ideal anomaly detection (depicted by the yellow ROC).

By using the ROC curves, it is possible to select possibly optimal models and to discard suboptimal ones. It also allows a model's parameters to be selected for different applications, based on the penalties for false anomaly detection and missed anomalies.

2.4.2 Coverage Maps

The coverage map of a model shows the likelihoods for a model receiving sufficient data during parts of a day for the model to make an anomaly classification. Coverage maps in this dissertation are generated for 1 hour chunks, synchronized with clock-time. The use of 1 hour chunks provides sufficient granularity in showing coverage while keeping the maps readable. To generate coverage statistics for a model, for each 1 hour slot of a 24 hour period, the number of times there is sufficient data[2]in the dataset for anomaly detection by the model are counted. The counts of each slot are normalized by the number of days covered in the dataset. The normalized value is equivalent to likelihood of that one-hour slot having sufficient data for the model to reliably perform anomaly detection. *I.e.,* the coverage of each 1-hour slot is calculated as the percentage of chunks in that slot to *qualify* for classification by the model. The statistics are plotted as a bar-chart with each bar representing a one-hour slot with the height of the bar representing the likelihood of the model having sufficient data for anomaly detection. Since the coverage is calculated per-user in an evaluation dataset and the values for each time slot are then averaged across all the users in the dataset, each time slot's bar also shows the 25^{th} percentile value (below) and the 75^{th} percentile value (above) for that time slot across all the users in the dataset. See Figure 2.2 for an example of a coverage map.

The following section presents an example of modeling a single aspect of a user's routine using a few of the information sources described in Section 2.2

2.5 An Example of Modeling an Aspect of User Behavior

This section shows a *people-seen-at-recurring-events* model (or *people-seen* model for brevity) as an example of utilizing one or more information sources to model an aspect of a user's routine.

[2] Sufficiency of data depends on the model being evaluated. For example in a messaging model sufficient data could be a minimum number of messaging interactions within a time window, for a GPS-based mobility model this could be accurate GPS locations for at least a minimum amount of time into the past.

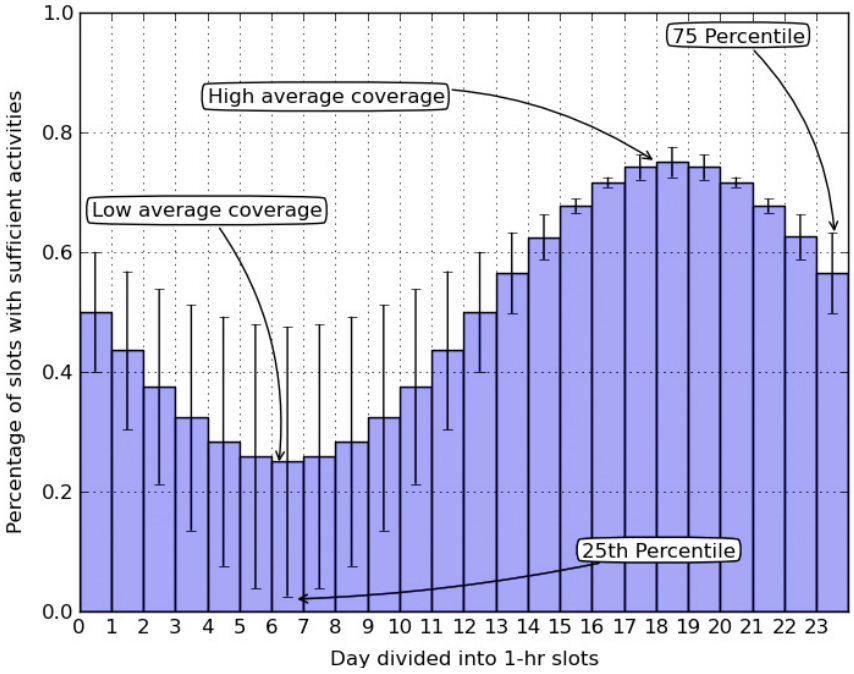


Figure 2.2: How to interpret coverage maps.

Here, the aspect of the user's routine modeled is the people and devices the user (in actual fact, the user's device) sees at recurring events. The model is based on the hypothesis that during the recurrences of an event, in most cases, the user will encounter the same people and be in proximity to the same devices. For example, a graduate student will see roughly the same people (*i.e.*, the same professors and students) at the same location at his group's weekly research meeting.

In this model, soft-sensor calendar entries and radio information of Bluetooth scan results are used to identify occurrences of recurring events and people and devices seen during event instances respectively. Recurring event times and their details are obtained from the calendar synchronized and maintained on the device. From the calendar, the times and titles of events are extracted. To detect who and what (*i.e.*, people and devices) were seen by the user, the Bluetooth MAC addresses seen by the mobile device are used. By using Bluetooth scan results as a method of detecting who was seen by the user, it is assumed that a certain percentage of the people that the user meets will carry their Bluetooth-capable mobile devices with the Bluetooth turned on. The mobile device's periodic scan results are used to gather lists of MAC addresses seen during

each scan.

To create this people-seen model the following likelihoods are calculated and utilized. For each recurring event:

1. likelihood of seeing a particular known Bluetooth device,
2. likelihood of seeing each known device during each Bluetooth scan during the event (this estimates the length of time during the event for which the known Bluetooth device was close to the user), and
3. likelihood of seeing unknown devices.

The likelihoods of seeing a known Bluetooth device and seeing an unknown device are estimated using maximum likelihood estimation (MLE) on a training dataset. The number of times a device is seen during an event instance is modeled using a Gaussian mixture model (GMM), trained and tuned using expectation maximization (EM) and Bayes' information criterion (BIC) over the training data[3].

Each of these likelihoods are modeled, per recurring event, as follows:

1. **Seeing a known Bluetooth device :** A known Bluetooth device is a device seen in training data. Likelihood of seeing a known Bluetooth device, $\mathcal{L}$, is estimated using MLE as,

$$\mathcal{L} = \frac{\text{the number of event instances where the device is seen}}{\text{the total number of event instances of that type}} \tag{2.3}$$

2. **Number of times a known device is seen during each event instance :**

   This value is calculated per recurring calendar event and per-device from training data using MLE. For each event instance, the number of times a known device is seen during all of the Bluetooth scans during the event instance are counted. For each event-device

[3] BIC is a criterion for model selection from a finite set or selecting the optimal number of components (again from a finite set) in a model. In this case BIC is used to identify the optimal number of Gaussian distributions to include in the mixture models. Adding more components to a model increases the model's ability to fit the training data better, but excessive addition of components leads to over-fitting. BIC uses the likelihood estimate of a model on the training data along with a penalty for each additional component in the model to identify the optimal number of components.

pair, the counts for that device from all of the event instances are modeled using a GMM comprised of a variable number of one-dimensional Gaussian distributions.

3. **Seeing an unknown device during an event :** To estimate this likelihood the number of Bluetooth devices that were seen during one and only one event instance of this event are counted. The event instance count is used to normalize this value. Likelihood of seeing an unknown Bluetooth device, $\mathcal{L}$, is estimated as,

$$\mathcal{L} = \min\left[\frac{\text{the number devices seen at only one event}}{\text{the total number of event instances of that type}}, 1.0\right] \tag{2.4}$$

To detect anomalous behavior using this model, during an event, E, this approach takes into account all the devices who it sees during that event ($B = b_1, b_2, \ldots, b_N$). Each b_j, may be seen 1 or more times during the event (c_{b_j}). The event E can be thought of as being comprised of a set of Bluetooth devices that are expected to be present ($E = e_1, e_2, \ldots, e_K$). For simplicity we assume $e_1, e_2, \ldots, e_K$ are independent of each other and $b_1, b_2, \ldots, b_N$ are independent of each other, even though this may not strictly be the case (*e.g.*, with a single user carrying two Bluetooth devices). The people-seen model, M, can be used to estimate the likelihood, $\mathcal{L}$, of the

devices seen while the user participates in the scheduled event.

$$\begin{aligned}\mathcal{L} &= P_M(B, E) \\ &= P_M(e_1, e_2, \ldots, e_K, b_1, b_2, \ldots, b_N) \\ &= \prod_{e_i = b_j} P_M(e_i, b_j) \cdot \prod_{e_i \notin B} P_M(e_i) \cdot \prod_{b_j \notin E} P_M(b_j)\end{aligned}$$

where,

$P_M(\ldots)$ are probabilities estimated using the trained people-seen model, and,

for $e_i = b_j$,

$$P_M(e_i, b_j) = P_M(\text{seeing } e_i) \times P_M(\text{seeing } e_i \; C_{b_j} \text{ times})$$

for $e_i \notin B$,

$$P_M(e_i) = 1 - P_M(\text{seeing } e_i)$$

for $b_i \notin E$,

$$P_M(b_j) = P_M(\text{seeing an unknown device})$$

This likelihood can then be compared to an *anomaly threshold* and if the likelihood is lower, an anomaly is flagged. Note that the time windows with less than an *activity threshold* number of activities are disregarded as not having sufficient information for classification.

2.5.1 Evaluation

The evaluations on the individual models were performed using the 30 user, 3 month Dataset II described in Subsection 2.3.2. A 3-fold validation approach for each user is used, where one fold (*i.e.*, one month of data) is used to train a model to be tested. The other two-thirds of the user's data is used as testing data of *normal behavior*. The common practice throughout this dissertation to simulate *anomalous behavior* was to use the complete data sets of the other users in the dataset. The only exception to this evaluation routine was for testing the people-seen

model. Since Bluetooth MACs seen during one user's day are extremely unlikely to be seen by other users in the dataset, using the Bluetooth records from other user to simulate anomalous behavior are trivial for the model to detect. Therefore to simulate anomalies during an event, the same user's Bluetooth records during other calendar events are used. To complete the other two folds of the 3-fold validation, this process is repeated twice more using each of the other two one-month periods of the user's data as training data in turn. This entire process is repeated for each fold of the 30 users in the dataset and the results presented here are the averaged results from these experiments. This experimental procedure assumes that during the data collection period each user did not have any substantial anomalies in their own behavior.

The first evaluation performed was to generate a Receiver Operating Characteristic (ROC) curve for the model to show how changing each model's anomaly threshold would affect the true positive rate (TPR) and false positive rate (FPR) for anomaly detection. For a full description of the evaluation metrics, refer Section 2.4. Figure 2.3 shows the ROC curve for the event-driven people-seen model. The ROC curve for this model indicates it is very effective at authenticating the user (with a 100% TPR at a FPR of less than 20%) given an entire event's worth of Bluetooth scan results. The average calendar event in this data set lasts between 30-60 minutes.

The second set of statistics generated are for coverage - the likelihoods for a user having sufficient data (during each hour of a day) for a model to perform anomaly detection. The number of users in the dataset that had Bluetooth scanning enabled was just 8 out of 30, and the coverage map shown in Figure 2.4 is the average across these 8 users. For a full description of the coverage maps see Section 2.4.

In summary, even though the people-seen model is excellent at detecting anomalies in user behavior, these results indicate that there is limited data during the day, and extremely sparse data during the the evening and early morning hours of the day. Therefore it would be ideal for this model to be combined with other models of different aspects of the user's behavior to give more complete coverage of the day.

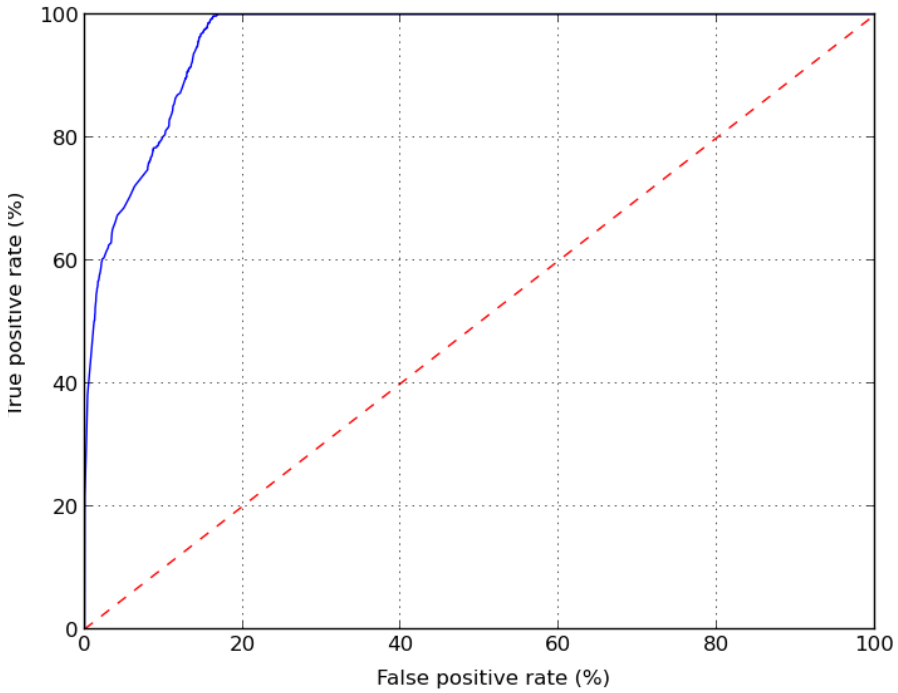


Figure 2.3: Receiver operator characteristic (ROC) curve of the event-driven people-seen model.

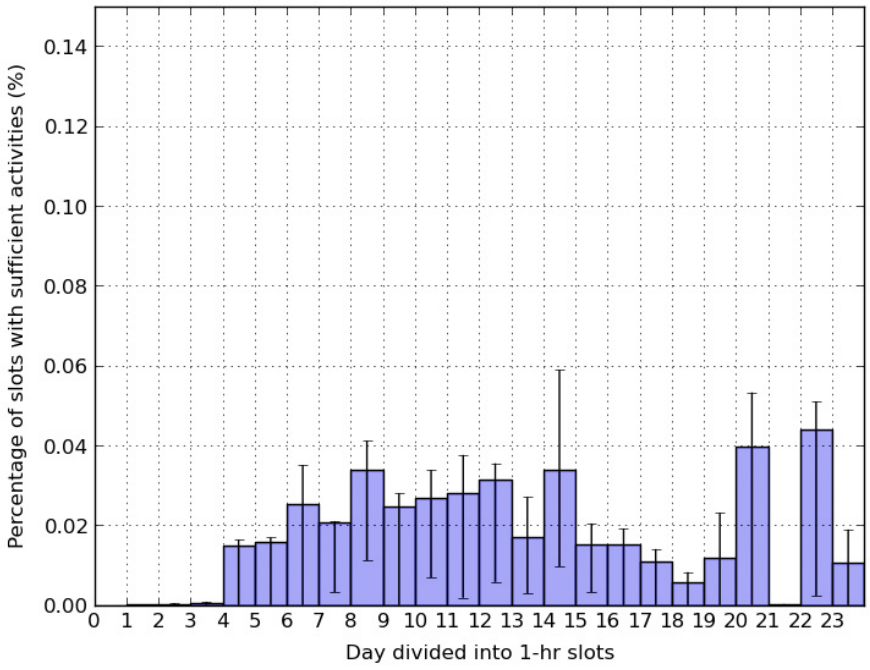


Figure 2.4: Coverage map of the people-seen model.

In the next chapter an example of a behavior aspect model using a hard sensor information source is discussed, along with a thorough evaluation of the model.

Chapter 3

Modeling Aspects of User Behavior using Hard Sensor Information

This chapter discusses an example of modeling an aspect of user behavior using hard sensor information sources. The model presented is an n-gram based model for modeling a user's mobility patterns, where the model is capable of detecting anomalous activities performed by the user and predicting the user's future locations. Experimental results comparing this model to alternate approaches for modeling human mobility are presented. Extensive experimental evaluations of the model performance show the versatility, and conversely the fine-tuning effort required, even in models that capture just a single aspect of the user's behavior. Also discussed is a hierarchical location partitioning system used by the model that is capable of obscuring a user's exact location, to protect privacy, while still allowing applications to utilize the obscured location data for modeling anomalies effectively. If the reader's interest lies primarily in the overall anomaly detection methodology presented in this dissertation Chapters 3, 4, and 5 can be skimmed on a first reading as each chapter presents detailed examples of creating behavior aspect models using each source of information category described in Section 2.2 (*i.e.,* hard sensor information sources, soft sensor information sources, and radio channel information sources).

3.1 User Mobility Modeling

Over the last decade, with smart-phone and ubiquitous computing technologies maturing, the ability to locate a user accurately has become a reality especially in outdoor environments using the Global Positioning System (GPS). With the ability to accurately track a user's location, it is theoretically possible to create a comprehensive model for that user's movement pattern [31].

Such a model, combined with location-based services (and context-aware services in general), enables a wide array of services using the model's ability to detect variations from a regular routine, and its ability to predict location and variations. A comprehensive model of a user's movement patterns would be able to detect if a user is doing something out of the ordinary. This capability of the model can support context-aware applications for,

1. caregivers to monitor elderly people unobtrusively (especially those suffering from memory ailments such as Alzheimer's Disease who are likely to wander),
2. monitoring of young children, and
3. theft detection systems for mobile phones and cars.

At an individual level, the predictive power of a user's movement models would allow applications such as pre-heating a house only when the user is going to arrive home soon [33]. Applications such intelligent call routing on cellular and other wide area networks with mobile users [46], [5] become possible at a community scale with the predictive capabilities of a user mobility model.

The model presented in this chapter is an n-gram [63] [1] based model for learning a user's movement pattern using historical geo-traces (GPS tracks) [10, 14]. Each person has a set of unique geo-locations and transitions (i.e., movements between those geo-locations) which act as triggers or qualifiers for the person's future locations. n-gram models are very robust at modeling relationships between a sequence of observations and outcomes dependent on the sequences. Therefore an n-gram model is used to learn a person's geo-patterns (significant and recurring

[1] See Subsection 3.3.1 for a detailed description of the n-gram model.

geo-traces). For example, if a person visits a cafe everyday on the way to work and the person is seen at the cafe on a workday, it can be reasonably assumed that the person is on his way to work. From the raw geo-trace reading from GPS, we cluster GPS coordinates into geo-labels and sequences of geo-labels are used to train n-gram models such as standard n-gram models for abnormal activity detection and skipped n-grams for future location prediction.

Using probabilistic methods for modeling the movement patterns of users has been attempted in previous work. These approaches have focused predominantly on using probabilistic approaches such as Markov models [2, 41, 44, 77] and T-Pattern based models [29, 47].

The model discussed in this chapter approaches the problem of grouping individual entries in geo-traces with a method that is distinct from previous attempts. It utilizes previous states (or locations) to detect anomalies in daily routines similar to using a T-Pattern or Markovian model. However, unlike those models, the n-gram based model presented is able to skip over previous locations which are detractors (or non-contributors) to the current prediction. By skipping over location entries which are deemed to be detractors or non-contributors, the n-gram model is made more robust to noise in data caused by either GPS resolution or minor variations in a user's movements. Furthermore skipping detracting n-grams also reduces the size of the model in terms of computational time and storage size. Therefore the model presented in this chapter is able to outperform models presented in previous work at detecting anomalies.

The next section presents the feature extraction processed used by this model.

3.2 Quantizing Geo-trace Data

Raw geo-trace information can be recorded from a user's mobile phone (using GPS) or specialized GPS hardware. While some specialized geo-tracking devices are capable of providing a greater amount of information about the user's movements (such as velocity), it is assumed that the only information available to the model is raw, timestamped GPS coordinates (longitude and latitude) along with the number of visible GPS satellites. The satellite count is used to clean

the data, rather than as a feature for the model [2]. It is assumed that the GPS data capture rate is at least 8×10^{-3}Hz (approximately once every 2 minutes), but a constant capture rate is not assumed.

Here, the term "geo-tag" is used to describe a tuple of raw readings consisting of a longitude-latitude pair, time-stamp, and visible satellite count, and the term "geo-label" is used to describe a tuple of quantized geo-tag data. The geo-trace models are built upon these geo-labels.

3.2.1 Location Quantization through Partition

Raw GPS readings are not directly used as a part of geo-labels, since the precision of the raw GPS readings is too high to generalize to a meaningful geo-trace model (i.e., the geo-trace models will attempt to learn with an unrealistic degree of spatial granularity, resulting in many false positives/false anomaly detections). For example, knowing that the user is driving on highway I-75 close to exit 187 heading north is enough to detect his future location and it is unnecessary to know which lane he is in to achieve such a prediction. Essentially, in this example, the left and right lanes are treated as one location rather than two. Additionally, users may have privacy concerns over sharing precise location information with an external application (such as a server/cloud-based application utilizing geo-trace models) [6, 40, 41]. Therefore the raw GPS readings are abstracted to reduce their precision and to generalize them to a useful geo-trace model. Two methods to perform this conversion are, 1) Equal-sized partitioning, and 2) Density-driven hierarchical partitioning.

Equal-sized Partition

The first method is to partition the entire surface of the earth into equal sized rectangular segments, each with a unique label. Then, all points from a geo-trace falling within a certain segment are replaced by the label of that segment. This method of partitioning is very straightforward to implement and makes comparisons between multiple users' geo-traces extremely

simple. But the method also has the drawback of over- and under-granularization. In the case of over-granularization, in areas where a user rarely travels, the few geo-tags that exist (over a relatively large geographic area) are split into a large number of partitions. In the case of under-granularization, areas regularly frequented by the user will have a large number of places of interest (e.g., campus premises with different buildings of interest) and a low granularity of partitioning could lose important information regarding visits to each individual place of interest within the partition.

Density-driven Hierarchical Partition

To counter the deficiencies of the first partitioning approach, a density-driven partitioning method was developed where the granularity of partitioning varies from area to area. The granularity of partitioning for an area is determined by the frequency of visits to that area, based on observed geo-traces of a user or a group of users. The process begins by treating the entire world surface as a single partition and recursively dividing it up into child partitions by splitting it into four quadrants with a north-south line and an east-west line. Further division of a partition is halted when the number of geo-tags recorded within that partition falls below a density threshold. The resulting partition is a quad-tree. A label is assigned to each leaf partition which denotes the path to the leaf node from the root node in the partitioning quad-tree. Each child partition's label is comprised of the parent partition's label as a prefix appended with a designator as to which quadrant of the parent it belongs. See Figure 3.1 for an example of a partitioned area on a user's route. The pseudo-code for the partitioning algorithm shown in Listing 3.1.

While there are partitioning schemes such as Hierarchical Triangular Meshes [21] which would result in a more even spatial partitioning, these schemes require greater computational cost at run-time. Furthermore such partitioning schemes do not provide significant performance improvements.

Figure 3.1: The map on the left displays a partition based on the frequency of visits by the user. The image on the right shows an example of how labels are assigned to partitions in a density based partitioning scheme.

```
Create root Partition with root->region = entire world;
Assign all geo-tags to root;
PartitionSimple( root );

PartitionSimple( Partition cp ) {
    IF( cp->geo-tags <= THRESHOLD ) THEN return;

    Add 4 child partitions to cp;

    i = 1;
    FOR EACH child partition p IN cp
        Set p->region = quadrant i of cp->region;
        Set p->label = Concatenate( cp->label, i );
        FOR EACH geo-tag gt IN cp
            IF (gt lies within p) THEN Assign gt to p;
        END
        PartitionSimple( p );
        i = i + 1;
    END
}
```

Listing 3.1: Hierarchical partitioning algorithm in pseudo-code.

3.2.2 Time and Direction Features

Other features, excluding location information, extracted for use in geo-trace modeling are:

- **direction of displacement**. The direction of displacement of a geo-tag g_i is the angle of the directional vector between the raw GPS coordinates of g_{i-1} and g_i, i.e., the direction the user moves from his previous location to the current location. The direction is 0 if $g_i - g_{i-1}$ points to North and 90 if East. Direction of displacement can also be quantized into B sectors where $B << 360$ (typically around 8).
- **time of day**. The time of day feature is extracted from the geo-tag's time-stamp. The 24 hour period of a day is divided into segments of equal duration (e.g., in experiments one-hour segments were used) and the actual hour:minute:second time-stamp is converted to a discrete label.
- **time spent at a location**. Consecutive sequences of identical geo-tags are replaced with a single geo-tag which indicates that the user is not moving or only moving inside the current partition. The time spent at that location is converted to a discrete value similar to the "time of day" feature (*e.g.,* using the 15 minute segmentation used in the experiments, three hours spent at a location translates into the 12^{th} discrete label assigned to the segments).

3.2.3 Geo-label

Different types of features are concatenated in various combinations to form a single geo-label. The combination of needed features depends on the application and the amount of training data available. For example, the location label can be combined with the time-of-day label as depicted in Figure 3.2 in order to model the user's mobility as being dependent on the user's previous locations as well as on the time of day.

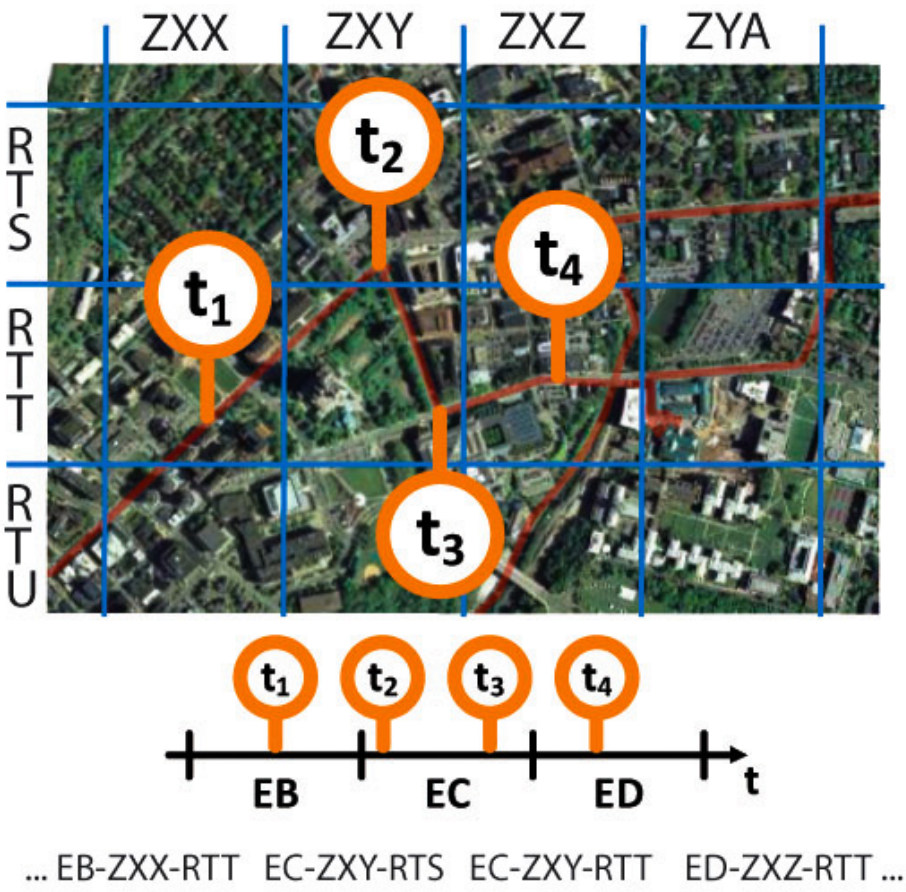


Figure 3.2: Creation of geo-labels using equal-sized partition location labels and time-of-day labels. In this example, "time spent at location" such as EB, EC, ED, are concatenated with the location labels such as "ZXX-RTT" and "ZXY-RTS".

3.3 Geo-trace Modeling

This section describes the n-gram model along with two alternative methods for geo-trace modeling. The alternative geo-trace modeling methods are used for performance comparison against the n-gram model.

3.3.1 n-gram Model

Shannon established that a language could be approximated by an n-th order Markov model [62], where n may extend up to infinity. n-gram[2]models have proven to be very robust in modeling sequences of data. Using an n-gram model trained on English text, it can estimated whether

"United" or "house" is more likely to follow the phrase "the president of the" by comparing the probability $P(\text{"United"} \mid \text{"the president of the"})$ and $P(\text{"house"} \mid \text{"the president of the"})$.

In this chapter, a user's geo-trace information is modeled using an n-gram model assuming that the sequence of locations of a person can also be approximated by n consecutive locations from the past (in essence, as a higher order Markov model). A geo-label is considered as a "word" in the language and an n-gram geo-label language model is trained on a user's geo-trace data. The model can then be used to estimate the next geo-label g_i given the previous $n-1$ geo-labels from the user's geo-trace as $P(g_i|g_{i-n+1}, g_{i-n+2}, \ldots, g_{i-1})$ or in short $P(g_i|g_{i-n+1}^{i-1})$. It is also possible to estimate the probability of a geo-trace $g_1, g_2, \ldots, g_N$ as:

$$P(g_1, g_2, \ldots, g_N) = \prod_{i=1}^{N} P(g_i|g_{i-n+1}^{i-1}) \tag{3.1}$$

Similar to the n-gram language model, the n-gram geo-trace model is based on the Markovian assumption that a user's next location depends solely on his/her previous $n-1$ locations. This assumption is not always true as there are many cases in which one's future location depends on locations that happened a long time ago (while the intermediate locations have little influence on the present location). For example, consider a user who goes to the gym everyday, either directly before or directly after work, when the user is at work the next location is independent of the user's current location, but is dependent on the user's previous location.

Training an n-gram Model

The model probabilities $P(g_i|g_{i-n+1}^{i-1})$ can be estimated using the Maximum Likelihood Estimation (MLE) from the training data by counting the occurrences of geo-labels:

$$P_{\text{MLE}}(g_i|g_{i-n+1}^{i-1}) = \frac{C(g_{i-n+1}, \ldots, g_{i-1}, g_i)}{C(g_{i-n+1}, \ldots, g_{i-1})} \tag{3.2}$$

[2] A *gram* in this context is a single token (*e.g.,* a word, a label, a letter). An n-gram is a sequence of n consecutive tokens.

where $C(g_i \ldots g_j)$ is number times the sequence $g_i \ldots g_j$ was seen in the training dataset. MLE is problematic when a geo-trace contains n-grams that have never occurred in the training data before, since MLE assigns probability zero to any unseen n-grams. Much of the prior work in GPS-based user mobility modeling centers around and is limited by this assumption, that the user is highly unlikely to visit locations that he has not visited previously (referred to as the "closed-world" assumption by Krumm and Horvitz [43]). To reject the closed-world assumption and to address this issue, Good-Turing discounting and Katz backoff smoothing [75] which were developed for language modeling, is applied in the geo-trace model. The key idea of smoothing (also known as discounting) is to discount the MLE probability for each observed n-gram in the training data to reserve some probability mass for unseen events.

3.3.2 Collapsing Recurring Geo-labels

In addition to quantizing GPS coordinates to labels, sequentially re-occurring geo-labels are collapsed into a single geo-label. This is to ensure that the n-gram models capture transition patterns of different locations rather than being dominated by a few locations that users spend a lot of time at. Instead, "time spent at a location" is modeled using the time-at-location feature (described in Section 3.2). The collapsing process is applied on both training and testing data for consistency.

Trigger Bi-grams

In order to predict a user's future location, discontinuous geo-label pairs from the original data are extracted to train trigger bi-gram models. A "trigger" is a geo-label pair (g_1, g_2) where g_1 and g_2 are usually not adjacent but have strong correlations. In other words, the occurrence of g_1 *triggers* the occurrence of g_2 in the future. For example, knowing that a user is at the entrance ramp of highway A close to his/her office, we can predict he will be at the exit of highway A close to his/her house, in the future. From the geo-trace data, pairs of geo-labels that are distance

d apart are extracted to train the trigger model for distance d. Conditional probabilities of seeing a "future" label given the current geo-label are estimated from the extracted pairs (see Section 3.4.7).

The trained n-gram model is used for anomaly detection by continuously feeding the model geo-labels in real-time. The n-gram model then outputs a probability estimate for the current geo-label being part of the user's learned geo-trace model, given the previously seen real-time geo-labels. The probability estimate is compared against a heuristically decided threshold (see Section 3.4.4), and if the estimate falls below the threshold it is considered an anomaly.

3.3.3 T-Patterns

The T-Pattern model identifies "statistically significantly" related geo-label pairs as signatures to characterize a user's movement model [29, 47]. A T-Pattern is a triple of *(starting location, time interval, ending location)*, where the "starting" geo-label is followed by the "ending" geo-label within the time interval $\langle t + d_1, t + d_2 \rangle$ $(d_2 > d_1 > 0)$. The T-Pattern extraction algorithm searches through the geo-labels for the most significant pair of recurring geo-labels. Statistical significance in this case is calculated using the Critical Interval Test described by [47] (See Figure 3.3). This pair is collapsed as a new label and the search is repeated on the modified data until no more pairs of statistical significance can be found. The result of the iterative search is a collection of hierarchical pairs and unpaired geo-labels (similar to binary trees).

The T-patterns extracted from the testing data can be compared against those extracted from the testing data to detect anomalies. In addition, when the "starting" label of a particular T-Pattern is observed in the testing data, it can be predicted that the user is likely to visit the corresponding "ending" location within the time interval specified by the T-Pattern.

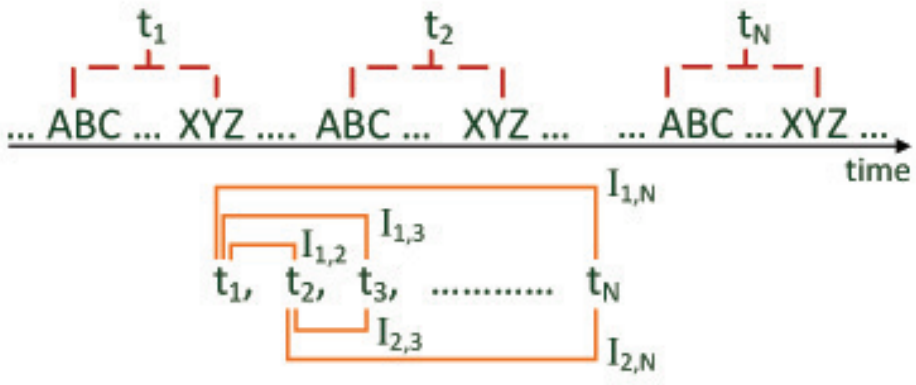


Figure 3.3: Extracting a T-pattern for the geo-label combination ABC and XYZ. The interval for the ABC-XYZ T-pattern is the largest interval $I_{i,j}$ satisfying the critical interval test [47] and containing more than a threshold amount of XYZ labels in the training data.

3.3.4 Decaying Probabilistic Transition Network

Alternatively, a Decaying Probabilistic Transition Network (DPTN) similar to [59] is proposed for modeling geo-traces. In DPTN, each distinct geo-label is represented as a node in a network and the network is fully connected. While traversing the DPTN following the sequence of geo-labels from a trace, the model calculates the probability of the present geo-label g_t given the preceding two geo-labels (g_{t-1}, g_{t-2}). A penalty value e_t is assigned for the present geo-label which is proportionally inverse to $P(g_t|g_{t-1}, g_{t-2})$. Denote E_t as an accumulated penalty up to time t and $E_t = e_t + \lambda E_{t-1}$, where λ is an empirically determined decay factor. When E_t exceeds an empirical threshold, the geo-trace is considered to be abnormal. The decay factor is introduced to 1) prevent the system from accumulating errors over a large amount of time, and 2) to make the model robust against sensor noise which would cause high penalties (by preventing these penalties from accumulating). The choices for the threshold value and decay factor are measures of the system's sensitivity to anomalies.

The performance of the three models described in this section are tested using real-world in the next section.

3.4 Experiments and Results

The dataset used to evaluate the models in this chapter is the 10 user, 4 week Dataset I described in Subsection 2.3.1. Further experimental results on the n-gram Geo-Trace model are presented in Chapter 6 using a much larger dataset.

3.4.1 Anomaly Detection

The following sub-sections discuss how well the n-gram model performed anomaly detection on this data. First, experimental results of anomaly detection accuracy of the n-gram model in comparison to other models are presented. Next the impact of using different feature sets with the n-gram model is analyzed. Then a discussion of alternatives for tuning the parameters of the n-gram model (i.e., size of n and the model's temporal detection threshold) is presented, followed by a discussion on the effect of location partitioning granularity on anomaly detection. Finally, detection accuracy variation with the amount of anomalous data seen is discussed.

The first set of experiments compares the three models described in Section 3.3 and a n^{th} order Markovian model (similar to that described by Ashbrook and Starner [2]) in an anomaly detection application.

In this experiment, each model was trained using a week of a user's data. Four separately trained models (of each model type) for each user were trained. (In the case of the n-gram models, n was set to 10.) As there are 10 users, this results in 40 n-gram models for all users, 40 T-Pattern models for all users, etc. The testing phase was conducted by taking a week long geo-trace of a user and dividing it up into segments of six hours and providing each six hour segment to all trained models except the model trained on the originating geo-trace. Each model provides a probability representing how likely it is that the geo-trace data is generated by the model. The owner of the model (i.e., user) that generates the highest probability is predicted to be the user that created the testing data. Table 3.1 shows the average accuracy of user identification using the three models described in Section 3.3. The n-gram model significantly outperforms the

Table 3.1: Comparison of owner-predicting accuracy given a segment of a geo-trace.

T-Pattern	Markovian model	Decaying Probabilistic Transition Network	n-gram model
72.6%	76.8%	79.2%	86.6%

T-Pattern and both Markovian models (*i.e.,* the DPTN and the traditional Markovian models). In this experiment the anomaly that is simulated is whether the phone is being carried by any person other than the owner (equivalent to determining the owner of a geo-trace segment), but the anomaly detection process is capable of detecting other types of anomalous activities (such as a phone owner wandering off on unfamiliar routes).

3.4.2 Geo-tag feature selection

The second anomaly detection experiment was to evaluate the use of the n-gram model with varying combinations of input features,

1. partitioning scheme
2. time of day
3. time at location
4. direction of travel

The n-gram model is set to $n = 10$, while using 4 hours' testing and training data segments. The cell size for the equally spaced partitioning scheme is set to be 40 meters by 40 meters. For each of the feature combinations, the experimental procedure described in the previous experiment was applied. The various feature combinations and results from this experiment are presented in Table 3.2.

The results indicate that both partitioning methods have very similar impact on the performance of the model's anomaly detection. Adding additional information such as time-at-location, time-of-day, or direction-of-travel, decreases the accuracy. This is due to the fact that the model is trained on only one week of data and adding additional information increases the

Table 3.2: Prediction accuracy of the n-gram model with various input feature combinations

Feature Combination	Accuracy
Location label from quad-tree partitioning (density threshold = 50)	86.0%
Location label from equally spaced partitioning	86.5%
Location label from equally spaced partitioning, time of day label	73.1%
Location label from equally spaced partitioning, time at location label	57.1%
Location label from equally spaced partitioning, direction of travel label	81.6%

dimensionality of the input, resulting in severe data sparseness.

3.4.3 Impact of history length n

This experiment was used to search for the optimal value of n in the n-gram model. Increasing the order of n captures more context dependency in the n-gram model and usually increases the accuracy of the model. On the other hand, the size of the resulting n-gram model grows fast when n increases which makes training and testing computationally expensive. Figure 3.4 shows anomaly detection accuracy vs the order of n-gram. This experiment was the same as described in Sub-section 3.4.2 except for the varying values of n. Here all experiments are based on 40m-by-40m equal partitioning.

The three curves in Figure 3.4 represent results for the experiment repeated with testing and training data segments of 2, 4, and 6 hours respectively. The results converge for $n \geq 6$ suggesting that the current location of the user is, in the majority of the cases, dependent only on his last five locations (*i.e.,* at $n = 6$ the model esimates the likelihood of the j^{th} element based on the $(j-1)^{\text{th}}$ to $(j-5)^{\text{th}}$ elements).

3.4.4 ROC curves of the n-gram model

Receiver Operating Characteristic (ROC) curves for the n-gram model are generated to show how changing the model's anomaly detection threshold would affect the true positive and false positive rates for anomaly detection. The curves are generated with $n = 10$, using equally

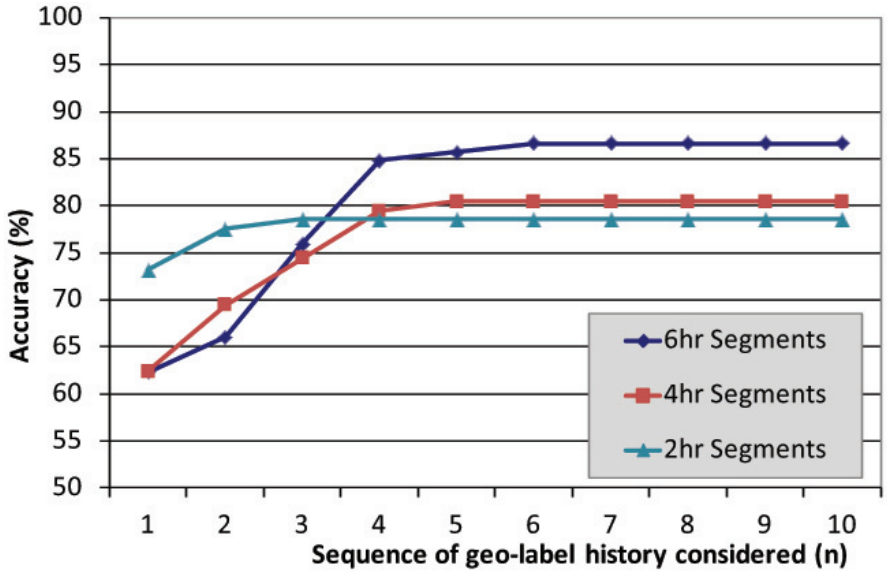


Figure 3.4: Variation of anomaly detection accuracy vs. length of geo-location sequence (n)

spaced partitions with granularity of 40m-by-40m. The five ROC curves displayed in Figure 3.5 are generated for testing segment lengths of 6 hours, 1 hour, 45 minutes, 30 minutes and 15 minutes.

On the figure, the y-axis shows correct anomaly detections (as a fraction of the total number of anomalies present in the testing data), and the x-axis shows incorrect detections (as a fraction of non-anomalous test cases in the testing dataset). This graph makes it clear the trade-off between segment length, true positive rates and false positive rates (refer to Subsection 2.4.1 for a detailed description of ROC curves). This is further discussed in Section 3.5.

3.4.5 Partitioning granularity

The fifth experiment carried out for anomaly detection was to identify the effects of geo- and temporal-partitioning granularity on the accuracy of the n-gram model's anomaly detection rate. For this experiment, the geo-partition scheme was used to partition the surface of the globe into cells of equal size, with the partition granularity varying from 4m-by-4m regions to 4km-by-4km regions. Again, every person-week geo-trace was used to train separate models and the models

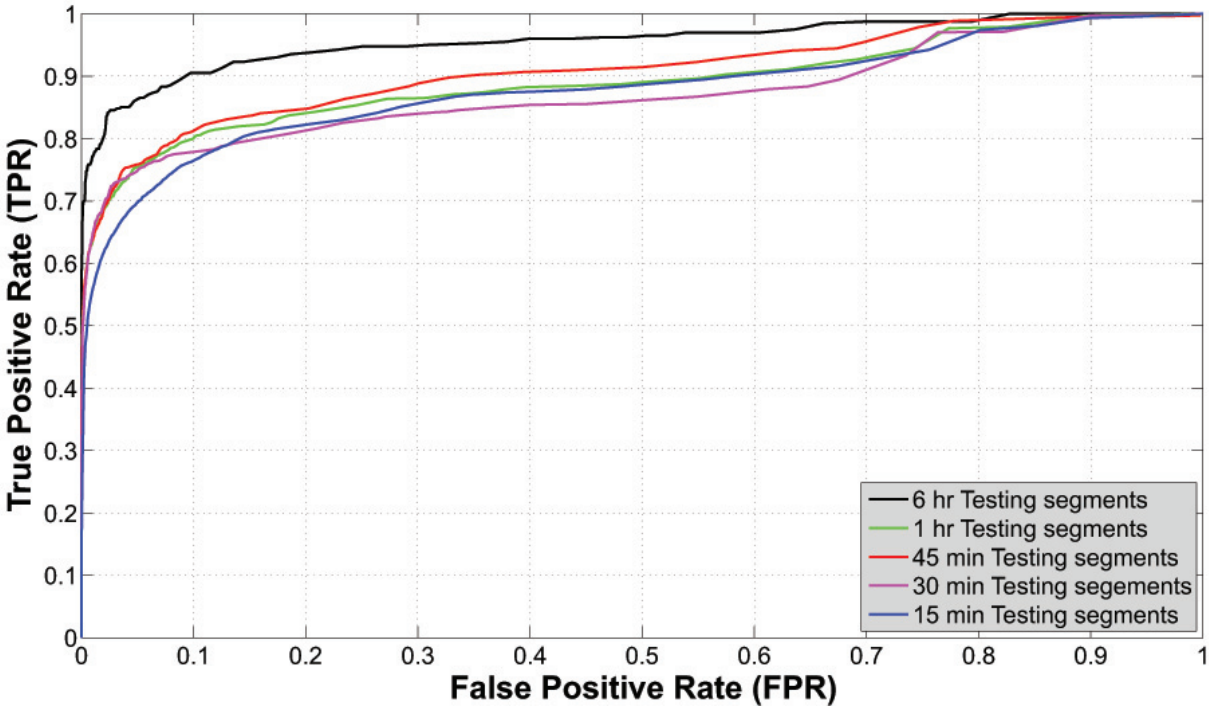


Figure 3.5: Receiver Operating Characteristic (ROC) curves of the n-gram model for testing segment lengths of 4 hours, 1 hour, 45 minutes, 30 minutes and 15 minutes.

were used to classify the owner of a segment of a geo-trace. The n-gram model uses $n = 10$. The results for this experiment are shown in Figure 3.6.

The three curves in Figure 3.6 represent results for the experiment repeated with testing and training data segments of 2, 4, and 6 hours respectively. The results indicate that the system performs best with cell sizes in the range of 40m to 80m.

3.4.6 Evaluating Variation of Accuracy vs. Anomalous Data Seen

An experiment was performed to estimate the amount of anomalous data that has to be seen by the n-gram model in order to achieve accurate detection. The same methodology as in the previous experiments was employed; only the length of the testing data segments was varied from 15 minutes to 12 hours. Partitioning for the experiment was performed using a 40m-by-40m cell size, while n for the n-gram model is set to 10. While accuracy in detection increases with amount of anomalous data seen, only 1 hour of data is required to achieve almost 80%

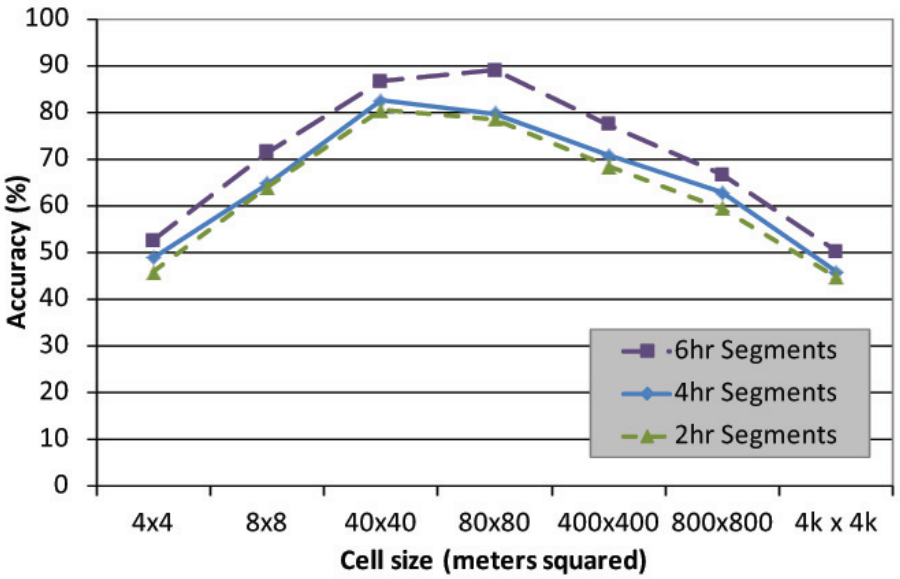


Figure 3.6: Variation of anomaly detection accuracy vs. location label cell size

accuracy (see Figure 3.7). These results are further discussed in Section 3.5.

3.4.7 Location prediction accuracy

While not the focus of this work, the geo-trace model created for anomaly detection performed with considerable accuracy in predicting a user's future location. The n-gram model is capable of predicting the future location of the person after a given interval of time. Such a prediction model enables a whole new class of applications. On an individual scale, the predictive power of user movement models enables the development of applications such as:

1. Pre-heating a house only when a resident is homeward bound [33, 61].
2. Intelligent context-aware meeting/schedule organizers [18].
3. Intelligent automotive navigation systems that use prediction when the user deviates from a given route [43].

When applied to a community, the predictive power of user movement models enables applications such as:

1. Intelligent call routing on cellular and other wide area networks with mobile users [5, 46].

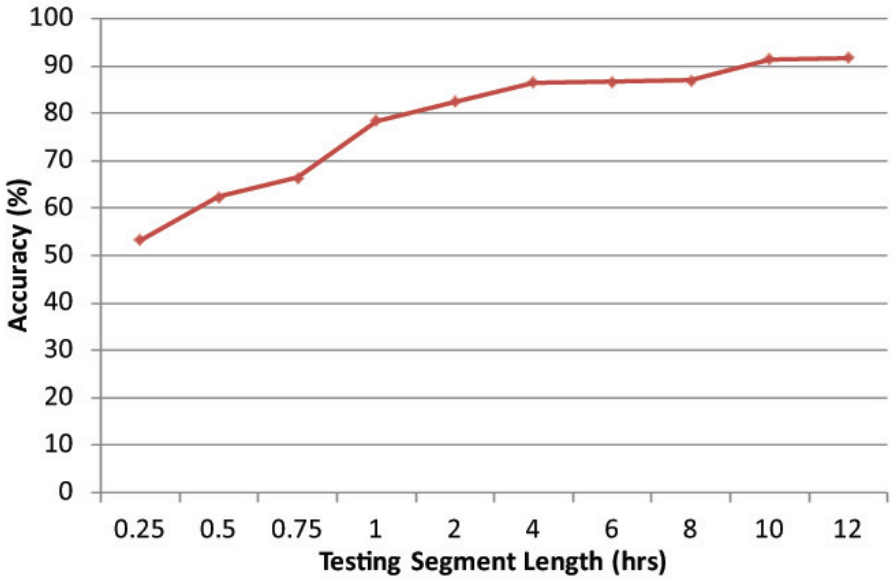


Figure 3.7: Variation of anomaly detection accuracy vs. the testing segment length

2. Energy management in green buildings [1].

3. City wide traffic routing [27].

4. Emergency message targeting [48].

To assess the n-gram model's predictive capabilities a week's worth of data was used to train a model for each person, to predict the user's location for the remaining 3 weeks using the gathered data as input data and validation information. The predictions compared are for where the user will be in the future after he travels for 5km. The reason for using distance instead of time as a way to measure the "future" was because time depends on many other factors besides a user's geo-trace pattern. To answer the question such as "where the user is going to be in 20 minutes?" depends on his/her commuting method (walking, running, biking, driving etc.) and traffic situation. A user's next location prediction output by the model is the centroid of a circle of 500m radius. The results are shown for 8 individuals in Table 3.3.

As can be seen from the results, the model is able to predict the future locations of most people quite accurately. The prediction accuracy may drop in situations where the user's mobility patterns are quite irregular or different between the training and testing data, as is the case for

Table 3.3: Percentage of correct predictions on a user's future movements based on a week of training data and the user's previous locations

Person ID	1	2	3	4	5	6	7	8
Correct predictions	61.9%	82.8%	96.0%	55.8%	31.9%	83.7%	82.1%	93.6%

the results for persons 4 and 5 in this experiment.

An entropy map was also created for each person using all four weeks of the person's data. The entropy maps show the degree of predictability. The highest entropy partitions, from where the user's next locations are most unpredictable, are in red. The descending order of entropy after red regions are pink, turquoise, green, blue, and black regions (a section of an entropy map of a user is shown in Figure 3.8). For future location prediction, it was found through observation of these maps that the equally spaced geo-partitioning method does not perform well, therefore for prediction, the model should use the quad-tree partitioning approach.

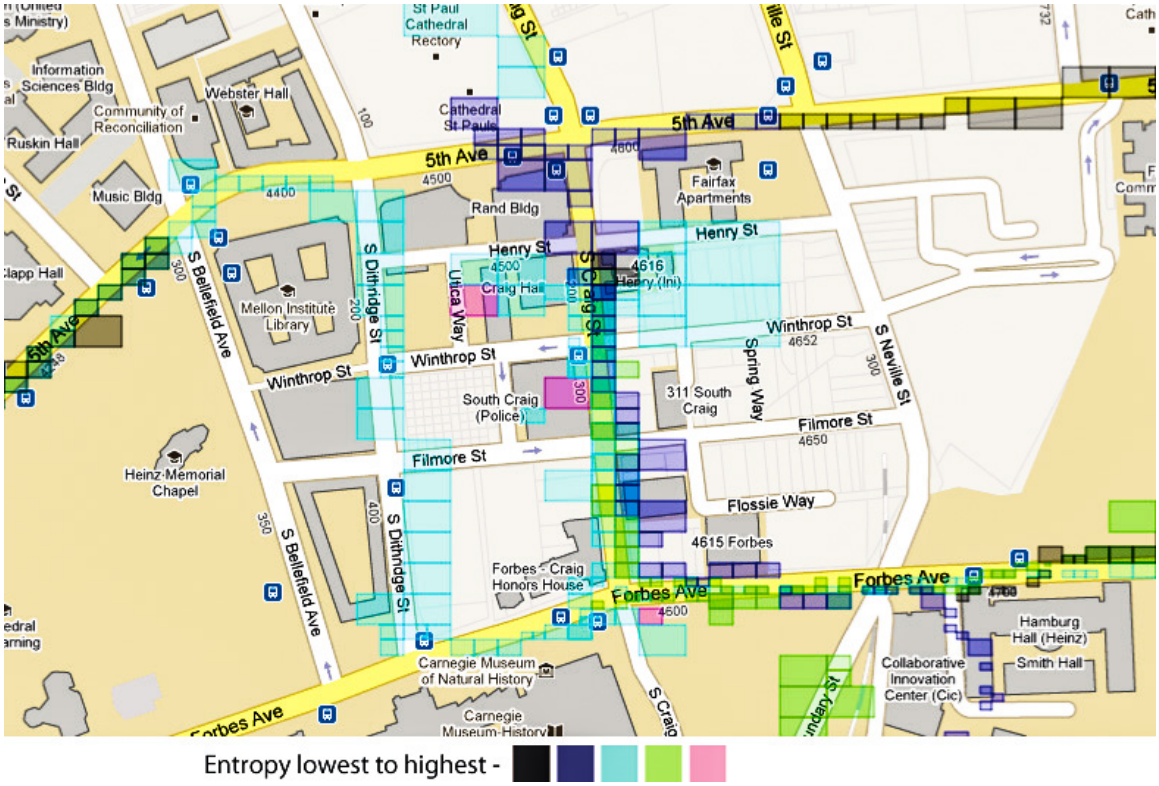


Figure 3.8: Entropy of locations frequented by user when considering locations 500m away as predictors of next location

3.5 Discussion

From the experimental results in the previous section it can be seen that the n-gram model is able to perform with high accuracy even in conditions of sparse training data. In all of the above experiments the models were trained on only a week's worth of user's data, which in terms of geo-traces is a small amount of training data (e.g., a user's Saturday routine has only one example in the training set). However, with more training data, it is expected that the n-gram model will show further improvements in anomaly detection accuracy (see Chapter 6 for experiments on the geo-trace model using Dataset II with four weeks of training data). With larger training datasets it is possible that incorporating additional features (such as time-of-day and time-at-location) into the model will present further performance gains.

As pointed out previously, each point on the ROC curves (Figure 3.5) corresponds to a certain detection threshold. Applications utilizing the n-gram model for anomaly detection will have their own unique costs for false-positives and false-negatives, and should customize the n-gram model's detection threshold so as to optimize the trade-off between false-positive and false-negative rates and overall accuracy.

The n-gram approach was evaluated with varying partition granularity size, and it was observed that it achieved optimal performance when the size was between 40x40m and 80x80m. This observation indicates that it is possible to abstract a user's location from a very specific GPS coordinate to a much larger cell (up to 6400 sq. meters), and still achieve high performance in detecting anomalies. For applications or users that require high levels of privacy, this feature provides a significant benefit. It was also observed that using hierarchical partitioning instead of equal-sized partitioning provides approximately similar results in detecting anomalies. Using hierarchical partitioning, the vocabulary size can be kept far smaller than with equal-sized partitions, resulting in a small n-gram model size. Also, the hierarchical partitioning approach can encode the user's location using a key (*i.e.,* the partitioning map) known only by the device to help preserve a user's privacy while retaining enough information for anomaly detection.

3.5.1 Applications of Geo-trace Modeling

Briefly presented here are geo-trace modeling applications enabled by the ability to detect anomalies in users' mobility routines. Anomaly detection schemes using geo-traces are useful in applications such as theft detection systems for cars and mobile phones, and elder-care and child-care monitoring systems. In these applications, the user carries a phone or GPS tracking device which continuously tracks the user's location and reports the location to the anomaly detection program running on the mobile device or servers in the cloud. The model is initialized in the "observing" mode where the system only learns the geo-trace of the user and does not set off any alarms. The "observing" time was empirically set to one week. After one week the anomaly detection function can be activated and the alarm will go off if the system believes that the mobile device is traveling on a strange route. Data points collected after the observing phase are continuously added to the training data to update the trained model.

Consider the theft detection application. The variation of accuracy of the anomaly detection system as a function of the testing data segment length is shown in Figure 3.7. The results indicate that the model is able to detect that the phone is stolen at an accuracy of close to 80% within an hour (approximately 30 GPS data points). The model is able to achieve an accuracy of over 60% after only 30 minutes (15 GPS readings). Alternatively, by examining the ROC curves, if a user is willing to put up with a 20% false positive rate, the model can achieve an 82% true positive rate after only 30 minutes. This is a good indicator that the n-gram model can be used in this real-world application.

The next section discusses previous work related to n-gram Geo-Trace modeling.

3.6 Related Work

Tracking the location of mobile phone users and using tracking information to enhance user experiences have been areas of extensive research over the past decade in both industry and

academia [13]. Most of these location-based systems, services and applications focus on utilizing the present location and most recent historical location information of the user [32]. There has been some research into more complex approaches that learn and model user mobility patterns. To the best of the author's knowledge there has been limited previous work in detecting anomalies in a user's behavior using learned mobility models. Shi et al. [64] use GPS readings in conjunction with call-logs and mobile phone browser history to detect anomalies and test their approach with a large amount of user data, but the models used are less adaptable as they are not conditioned over prior events.

The approaches in the previous work in user mobility modeling were used in experiments presented in this chaper for comparison against the n-gram model. The approach employed by Salvador et al. [59] uses a state machine based probabilistic approach for anomaly detection (equivalent to the Decaying Probabilistic Transition Network in the presented experiments) with a different type of data which also has an important time- and sequence-dependent nature. The trajectory pattern (T-pattern) mining approach by Giannotti et al. [29] presents the notion of time dependent sequences (mined from historic data) for modeling user mobility. Ashbrook and Starner use n^{th} order Markov models to predict the next building a user will visit when observing the previous building visited by the user [2]. This model is designed to work with relatively clean location data as the preprocessing step discards or aggregates the majority of data into a few clusters. The Predestination system by Krumm and Horvitz [43] uses a partitioning method where the world is divided into equal sized cells (similar to the equal-sized partition method presented in this chapter). They observe that a partially traveled route is usually an efficient path to the final destination. Ziebart et al. [77] and Krumm [42] also use Markovian models for destination prediction as well as shorter term predictions of the route to be traveled. The focus of the work presented in this chapter is not location prediction, but since the prediction results of the n-gram Geo-Trace are promising, it is a worth-while avenue of investigation.

While there has been little previous work in detecting anomalies in user mobility patterns,

there exists some previous work in detecting anomalies in other forms of context information. For example, Duong et al. use a variant of Hidden Markov Models (HMMs), named Switching Hidden Semi-Markov Models for recognizing and detecting anomalies in human activities of daily living in smart environments [20]. There has been much work in other areas for anomaly detection, especially for security applications. An extensive survey of anomaly detection methods using machine learning approaches is provided by Hodge and Austin [37]. Xiang and Gong develop a framework for automatic behavior profiling and abnormality detection in surveillance video streams [73]. In [22], Eskin et al. describe an unsupervised anomaly detection framework for intrusion detection in network systems. A survey of work about anomaly detection in the field of intrusion detection in network security is provided by Patcha and Park [53]. Chandola et al. provide a broad survey of various anomaly detection schemes and their application domains in [11].

This chapter presented a novel n-gram based method for modeling a mobility aspect of a user behavior using a hard-sensor information source. Simple as they may be, n-gram models perform surprisingly well compared to more complicated probabilistic approaches such as T-Pattern and Markovian models for anomaly detection tasks. This chapter also presented a geo-partitioning method which can help preserve a user's privacy by blurring GPS coordinates, while retaining enough information for applications to model the user's behavior and perform anomaly detection with great accuracy. In the following chapter modeling aspects of user behavior using soft sensor information sources are discussed along with two example models.

Chapter 4

Modeling Aspects of User Behavior using Soft Sensor Information

This chapter describes two models that utilize soft sensor information to capture aspects of the user's behavior [9]. These models present an alternative to modeling purely based on hard-sensor information and act as a contrast to the geo-trace model presented previously. A thorough evaluation of the performance and coverage of each model are presented. Also presented are an evaluation of the effect of internal model parameters on the performance of the models.

4.1 Message Response Patterns Model

The message response patterns model captures the behavior aspect of how the user responds to incoming messages (*e.g.,* SMS, instant messages) and how and when the user initiates a new message thread. This message response model is based on the hypothesis that a delay in a user responding to a received message will depend heavily on the significance the user places on the sender (or contact) of the original message. Though the content of a message may also affect the response delay, it is assumed, for this analysis, that the criticality of message content will not vary significantly for a single contact and the delay in the device owner checking a

message is still governed by the importance the owner places on the contact. Therefore the message response model consists of a collection of smaller models, one for each contact the user communicates with. While only SMS messages are considered in the evaluation of this model, it is straightforward to extend this approach to emails and instant messages as well. This model is especially well suited for detecting authentication anomalies, *i.e.,* if the device is stolen the thief would not respond to messages for fear of discovery, or if the thief were attempting to gain personal information about the device owner via the contacts the response delay to messages would be anomalous. The intuition behind using this model for stress or health-care related anomaly detection is that a user breaking from routine would have a different delay in responding to received message as the user is more stressed or anxious [16], and the user would possibly message numbers with unusual frequencies (*e.g.,* to alert family/co-workers of change of plans [17]).

To create the message response model, in addition to the response delays, the following likelihoods are modeled for the user for the user:

1. Initiating a messaging conversation with each contact.
2. Responding to a message from each contact.
3. Initiating a messaging conversation with an unknown number.
4. Responding to a message from an unknown number.

All likelihoods are calculated per-contact (all unknown numbers are treated as a single contact) using Maximum Likelihood Estimation (MLE) on the training data. Each of these actions are modeled as follows:

$$\mathcal{L}\left[\text{Initiating a conversation with a contact}\right] = \frac{\text{number of times the user initiated a conversation with the contact}}{\text{number of times the user messaged the contact}} \quad (4.1)$$

$$\mathcal{L}\,[\text{Responding to a message from a contact}] = \frac{\text{number of times the user has responded to messages from the contact}}{\text{number of times the contact has messaged the user}} \quad (4.2)$$

For each contact, the response delay is modeled as a distribution using a Gaussian Mixture Model (GMM), where a GMM is comprised of a variable number of one-dimensional Gaussian distributions. To train the GMM, the delays in responding to messages from the contact are compiled and used with the Expectation-Maximization (EM) algorithm and the Bayesian information criterion (BIC) to decide on the number of Gaussian distributions to use in the mixture model and to train the final GMM (see Section 2.5 for more details on using EM and BIC). In this approach we have not incorporated features such as time-of-day, which could possibly increase the accuracy of the model, at the cost of a drastic increase in training data requirements.

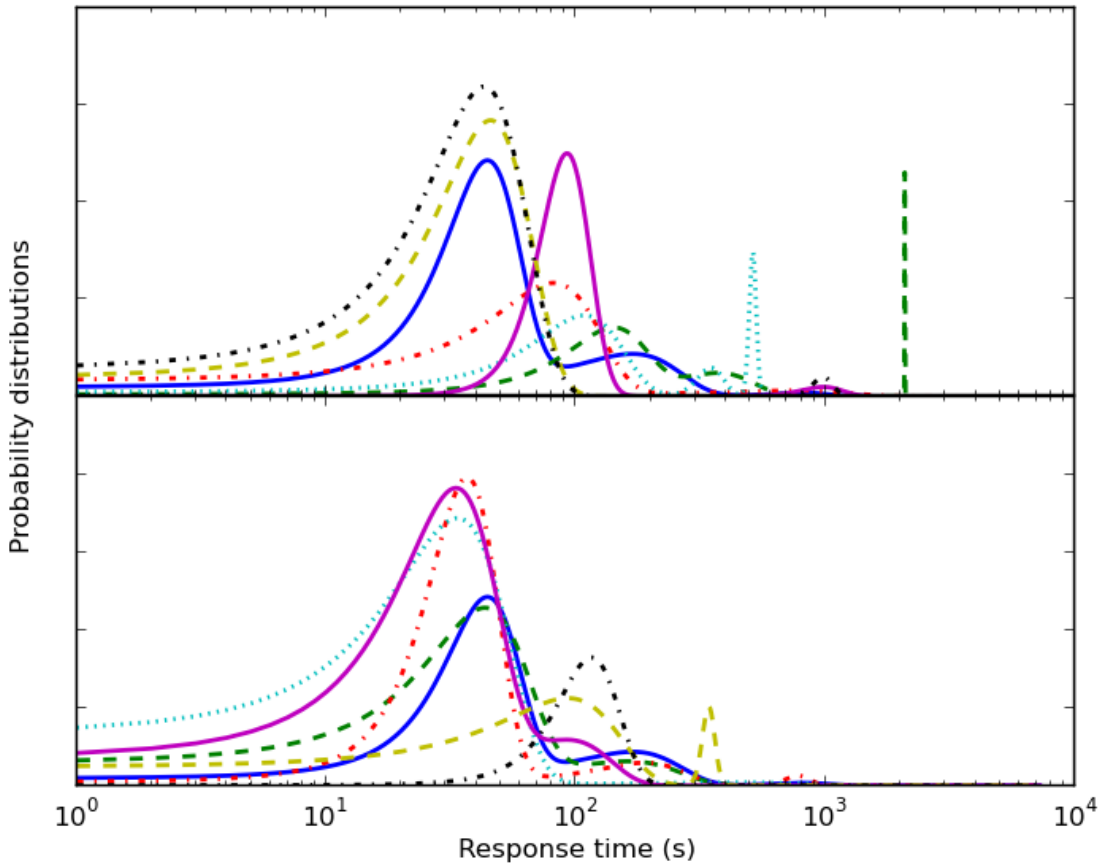


Figure 4.1: Visualization of response delays modeled as GMMs. *Top:* 6 most-interacted-with contacts of a user. *Bottom:* most-interacted-with contact of 6 different users.

The top plate of Figure 4.1 shows the Gaussian mixture models representing the response delay distributions for the top 6 contacts for a single user, in terms of number of interactions. The varying number of peaks and peak locations for each GMM shows the difference in urgency in responding to messages from various contacts by the same user. The bottom plate of Figure 4.1 show the response delay distributions of the top-most contact of 6 different users. Once again the varying number of peaks, peak heights, and peak locations for each GMM shows the difference in urgency, of different users, in responding to messages from their top contact.

For anomaly detection, this approach takes into account all messaging activity, A, in the past t time period. A is comprised of individual actions $(a_1, a_2, \ldots, a_N)$, such as initiating a conversation, responding to or ignoring a received message for the duration of t. For simplicity it is assumed that $a_1, a_2, \ldots, a_N$ are independent of each other. The trained messaging response patterns model, M, can be used to estimate the likelihood $\mathcal{L}$ of the activity A being generated by normal behavior of the device owner as,

$$
\begin{aligned}
\mathcal{L}\,[\text{Normal behavior}] &= P_M\,(a_1, a_2, \ldots, a_N) \\
&= P_M\,(a_1 | a_2, \ldots, a_N) \cdot P_M\,(a_2, \ldots, a_N) \\
&= P_M\,(a_1) \cdot P_M\,(a_2, \ldots, a_N) \\
&\qquad \{\text{assuming } a_1 \perp a_2 \ldots a_N \,(\textit{i.e.,}\ a_1 \text{ is independent of } a_2 \ldots a_N)\} \\
&= P_M\,(a_1) \cdot P_M\,(a_2) \cdot \ldots \cdot P_M\,(a_N) \\
&= \prod_{i=1}^{N} P_M\,(a_i) \qquad (4.3)
\end{aligned}
$$

where $P_M(\ldots)$ are probabilities estimated using the trained messaging response patterns model,

and,

$$P_M(A_i) = \begin{cases} P_M(\text{Initiating a conversation with a contact}) \\ P_M(\text{Initiating a conversation with an unknown number}) \\ P_M(\text{Responding to a message from an unknown number}) \\ P_M(\text{Responding to a message from a contact}) \\ \quad = P_M(\text{responding}) \cdot P_M(\text{delay}) \\ P_M(\text{Unreplied message from a contact}) \\ \quad = 1 - P_M(\text{responding}) \end{cases} \tag{4.4}$$

This likelihood is then compared to an *anomaly threshold* (see Figure 4.2 and description) and if the likelihood is lower, an anomaly is flagged. Time windows with less than a pre-selected *activity threshold* number of activities are disregarded as not having sufficient information for classification, thus impacting coverage of this approach.

4.1.1 Model Evaluation

The evaluations on this model was performed using the Dataset II (see subsection 2.3.2) using all 3 months of data from all 30 users. A 3-fold validation approach was used for each user, where one fold (*i.e.*, one month of data) is used to train a model to be tested. The other two-thirds of the user's data are used as testing data of *normal behavior*. To simulate *anomalous behavior* the complete data sets of the other 29 users in the dataset are used. This entire process is repeated for each fold of the 30 users in the dataset and the results presented here are the averaged results from these experiments. This experimental procedure assumes that 1) during the data collection period each user did not have any substantial anomalies in their own behavior, and 2) the anomalies that are of interest are significant to the extent of mirroring the behavior of a different user.

To simulate anomalies for the messaging model the numbers in each user's logs are replaced with numbers from a shared list. The process for this number substitution is to first sort the numbers in a user's logs according to the number of interactions the user has with each number

in descending order. Then replace each number with the number in the corresponding position in the shared list. The shared list is a list of randomly generated non-repeating numbers. In the case where a device is stolen and used to message numbers outside of the owner's contact list and messages from known contacts are ignored, the model will flag anomalies quickly. These experiments simulate harder-to-detect situations where the user's behavior has changed significantly or the device is stolen and being used to either phish for information about the device owner through contacts or impersonate the user with malicious intent.

Receiver Operating Characteristic (ROC) curves

The first evaluation performed was to generate Receiver Operating Characteristic (ROC) curves for each of the models to show how changing each model's anomaly threshold would affect the true positive rate (TPR) and false positive rate (FPR) for anomaly detection (see Subsection 2.4.1 for a thorough description of ROC curves). For the message response patterns model the five ROC curves displayed in Figure 4.2 are generated with activity thresholds of 2, 5, 10, 15 and 25 and time windows of 1 hour for the first three thresholds and 4 hours for the last two thresholds (with higher activity thresholds and smaller time windows, the number of testing data points is too low for meaningful results).

The ROC curves for the message response model (see Figure 4.2) show that the message response model is effective at detecting anomalies when it has seen at least 4 hours of anomalous messaging logs, but the effectiveness of the model drops (to around 75% TPR at a 20% FPR) when the model is forced to classify with just one hour of data.

Model Parameter Tuning

The experiment presented was used to identify the effect of the testing data window size and the event threshold on the messaging response patterns accuracy of detecting anomalies. Increasing the activity threshold increases the information the model receives about the activities, therefore

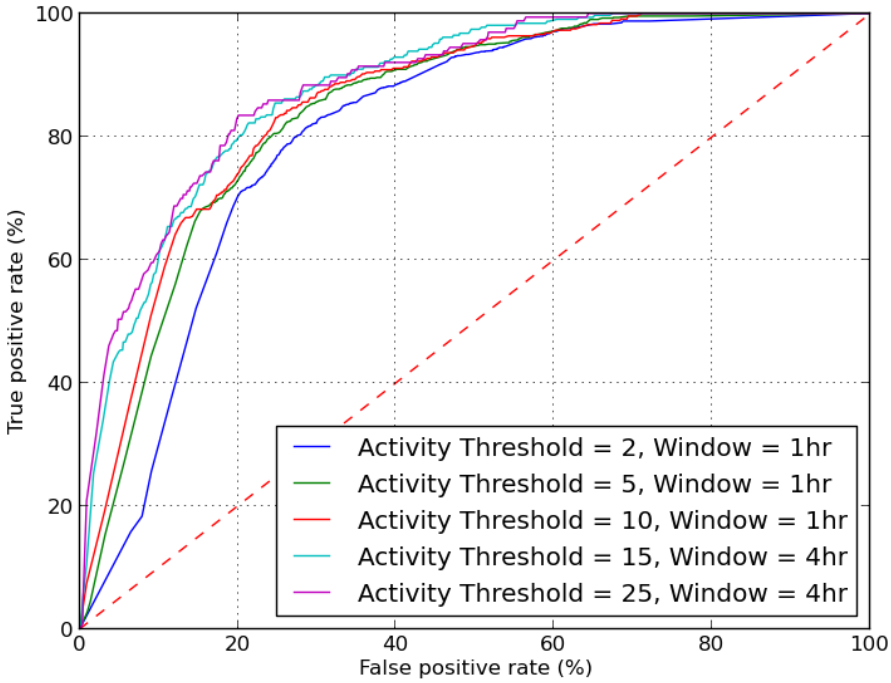


Figure 4.2: ROC curves for message response patterns model.

increasing *classification accuracy*. On the other hand, increasing the activity threshold for a particular window size reduces the probability of a window containing the sufficient activities to qualify for anomaly detection, reducing both *accuracy* and *coverage*. The results from these experiments are shown in Figure 4.3.

All of the curves in Figure 4.3 show an initial increase in accuracy, as expected, when the activity threshold is increased. As the activity threshold is increased further the curves representing the smaller time windows show a decrease in accuracy. This accuracy drop is due to only a small amount of data from the testing set qualifying for classification and creating a bias in the results.

Model Coverage

In this experiment, the likelihoods for a user generating sufficient data (during each hour of a day) for a model to make an anomaly classification was explored. See Subsection 2.4.2 for a detailed description of the coverage calculation. The coverage likelihoods averaged across all 30 users for the message response patterns model are shown in Figure 4.4.

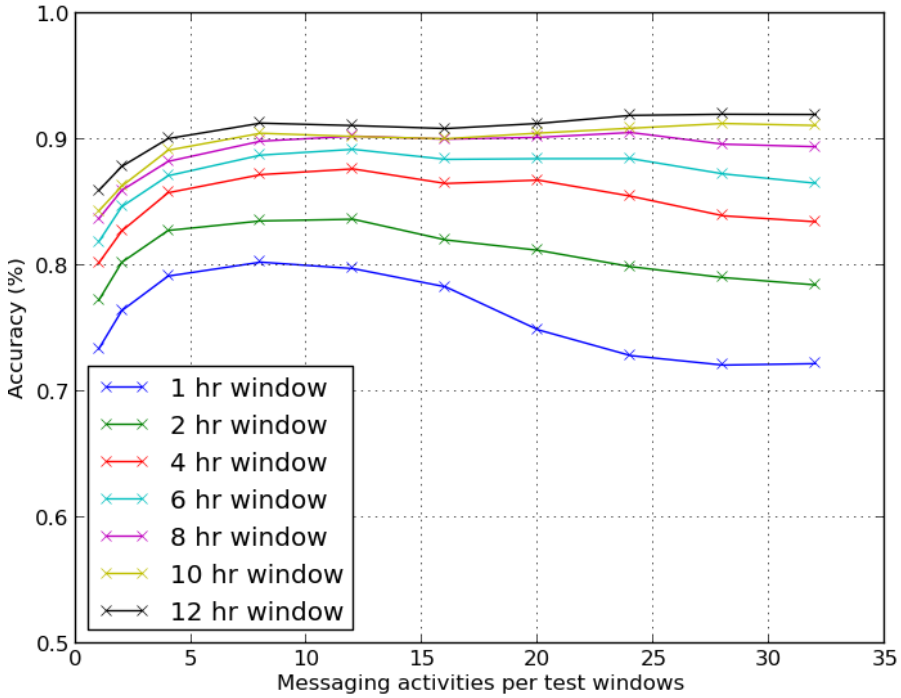


Figure 4.3: Variation of accuracy of message response patterns model with activity threshold (for different time window sizes).

In summary, these experiments indicate that the amount of time during each day where there is sufficient data is somewhat low for the message response patterns model. Also it is worth noting that the variance of coverage between the user's (especially during the day) is high. For these reasons, it would be ideal for this model to be combined to give more complete coverage of the day.

4.2 Call Patterns Model

The approach taken to build the call patterns model, as with the message response patterns model, is to model how users respond to and make calls from different contacts. Despite the many factors (*e.g.*, reason for the call, time of day, owner's activity) that may impact calling behavior, it is assumed that call duration and delays in responding to missed calls depend primarily on who the caller is. This model consists of a collection of smaller ones, one for each contact.

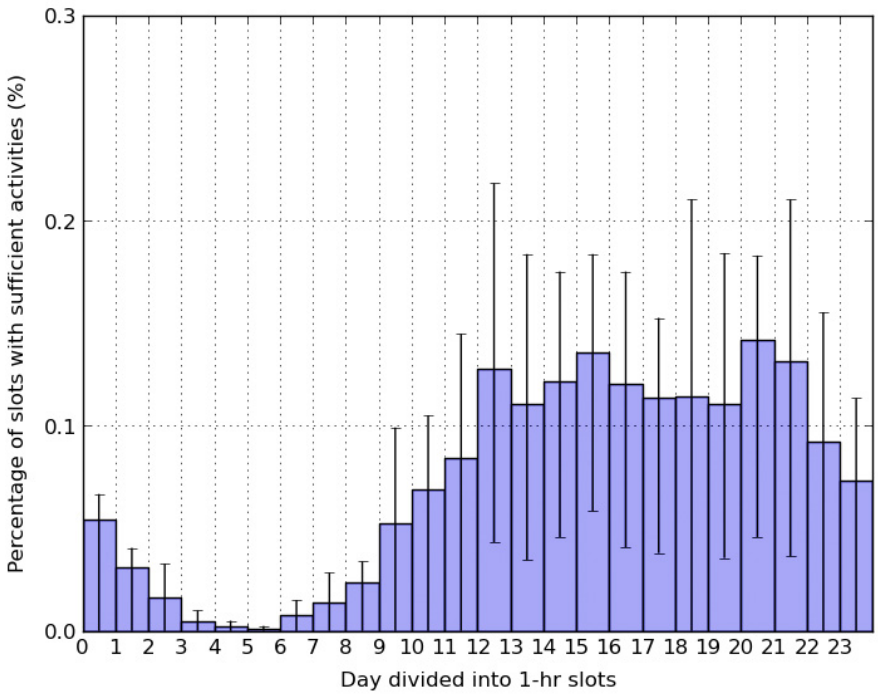


Figure 4.4: Coverage map for message response patterns model

This model is especially well suited for detecting authentication anomalies, *i.e.,* if the device is stolen the thief would not answer incoming calls and would not respond to these missed calls, while the thief could possibly call numbers that the device owner would not, all of which would be detected as anomalous. The intuition behind using this model for stress or health-care related anomaly detection is that a user breaking from routine may ignore incoming calls and would have a different delay in responding to missed calls when the user is more stressed or anxious [16], and the user would possibly call numbers with unusual frequencies (*e.g.,* to alert family/co-workers of change of plans [17]).

To create the voice call patterns model, likelihoods are modeled for each of the following factors per-contact (unknown numbers are considered as a single contact):

1. Duration of an incoming call.
2. Duration of an outgoing call.
3. Responding to a missed call.

4. Delay in responding to a missed call.

Note that all unknown numbers are modeled as a single combined contact (i.e., statistics from all numbers that are not in the contact list are aggregated). The likelihoods of responding to a missed call are estimated using MLE from training data. For each contact, the outgoing / incoming call durations and missed call response delay distributions are modeled as three separate Gaussian mixture models (GMMs). The GMMs are trained and tuned, as with the message response patterns model, using the Expectation-Maximization (EM) algorithm and the Bayesian information criterion (BIC[1]). Features such as time-of-day are not incorporated in this model due to training data limitations even though such features could possibly increase the accuracy of the model.

For anomaly detection, this model takes into account all voice call behavior, V, in the past t time period. V is comprised of individual actions ($v_1, v_2, \ldots, v_N$), such as responding to a missed call (with a delay of d_{v_i}), making an outgoing call (with a duration of d_{v_i}), or answering an incoming call (with a duration of d_{v_i}). For simplicity we assume $v_1, v_2, \ldots, v_N$ are independent of each other. The voice call patterns model can be used to estimate the likelihood $\mathcal{L}$ of the behavior V being generated by normal behavior of the device owner, where,

$$
\begin{aligned}
\mathcal{L}\left[\text{Normal behavior}\right] &= P_M\left(V\right) \\
&\quad \{\text{where } P_M(\ldots) \text{ are probabilities estimated using the} \\
&\quad\quad \text{trained call patterns model.}\} \\
&= P_M\left(v_1, v_2, \ldots, v_N\right) \\
&= \prod_{i=1}^{N} P_M\left(v_i\right) \qquad (4.5) \\
&\quad \{\text{assuming } v_i \perp v_j \forall i, j \in [1, N], i \neq j \text{ (\textit{i.e.,} all } v_i \text{ are independent)} \\
&\quad\quad \text{and applying the chain rule.}\}
\end{aligned}
$$

[1] See Footnote 3 in Section 2.5 for more details on using BIC.

and,

$$P_M(v_i) = \begin{cases} P_M(\text{Incoming call of duration } d_{v_i} \text{ from a contact}) \\ P_M(\text{Incoming call of duration } d_{v_i} \text{ from an unknown number}) \\ P_M(\text{Outgoing call of duration } d_{v_i} \text{ to a contact}) \\ P_M(\text{Outgoing call of duration } d_{v_i} \text{ to an unknown number}) \\ P_M(\text{Responding to a missed-call from a contact}) \\ \quad = P_M(\text{responding}) \cdot P_M(\text{delay}) \\ P_M(\text{Responding to a missed-call from an unknown number}) \\ \quad = P_M(\text{responding}) \cdot P_M(\text{delay}) \end{cases} \tag{4.6}$$

This likelihood is then compared to an *anomaly threshold* (see Figure 4.5 and description) and if the likelihood is lower, an anomaly is flagged. Time windows with less than an *activity threshold* number of actions are disregarded as not having sufficient information for classification, impacting coverage.

4.2.1 Model Evaluation

The evaluations on this model was performed using the Dataset II (see Subsection 2.3.2) using all 3 months of data from all 30 users. The approach used is identical to the approach (described in Subsection 4.1.1) used to evaluate the message response patterns model. To simulate anomalies for the calling pattern model the numbers in each user's logs are replaced with numbers from a shared list. The process for this number substitution is to first sort the numbers in a user's logs according to the number of calling interactions the user has with each number in descending order. Then replace each number with the number in the corresponding position in the shared list. The shared list is a list of randomly generated non-repeating numbers. In the case where a device is stolen and used to call numbers outside of the owner's contact list and calls from known contacts are ignored, the model will flag anomalies quickly. These experiments simulate harder

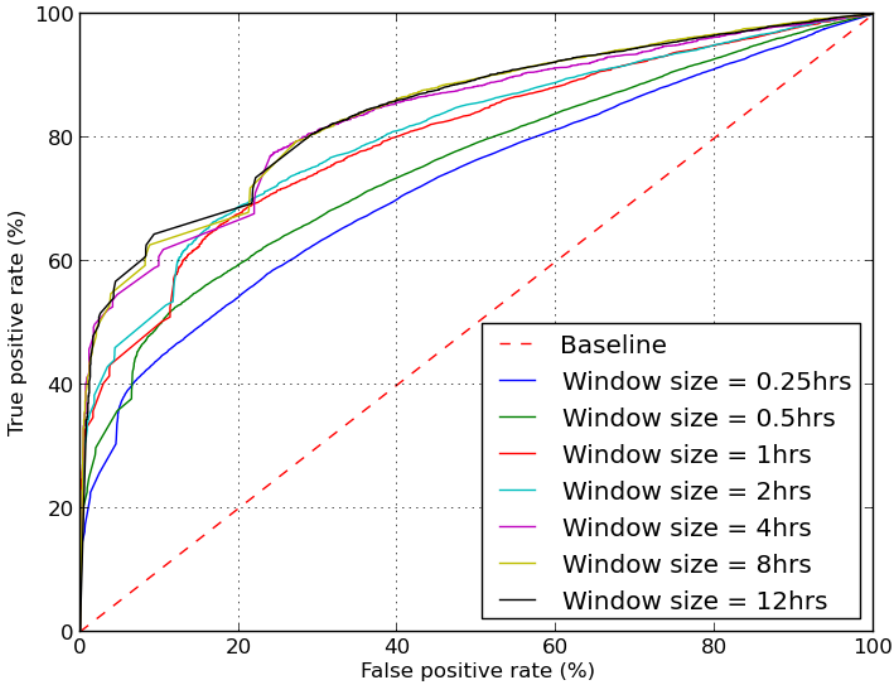


Figure 4.5: ROC curves for the call patterns model.

to detect situations where the user's behavior has changed significantly or someone impersonates the user with malicious intent.

Receiver Operating Characteristic (ROC) curves

The first evaluation performed was to generate Receiver Operating Characteristic (ROC) curves for each of the models to show how changing each model's anomaly threshold would affect the true positive rate (TPR) and false positive rate (FPR) for anomaly detection (see Subsection 2.4.1 for a thorough description of ROC curves). For the call patterns model, the seven curves displayed in Figure 4.5 are generated with varying time windows of 15 minutes, 30 minutes, 1, 2, 4, 8, and 12 hours and corresponding activity thresholds of 1, 1, 3, 3, 6, 6, 6 events.

For the call patterns model, the ROC curves (see Figure 4.5) show that the voice call patterns model can only be used with moderate effectiveness (with around a 70% TPR at a 20% FPR). The ROC curves also show how the model's classification capabilities improve as more prior data (*i.e.,* more events) are seen by the model.

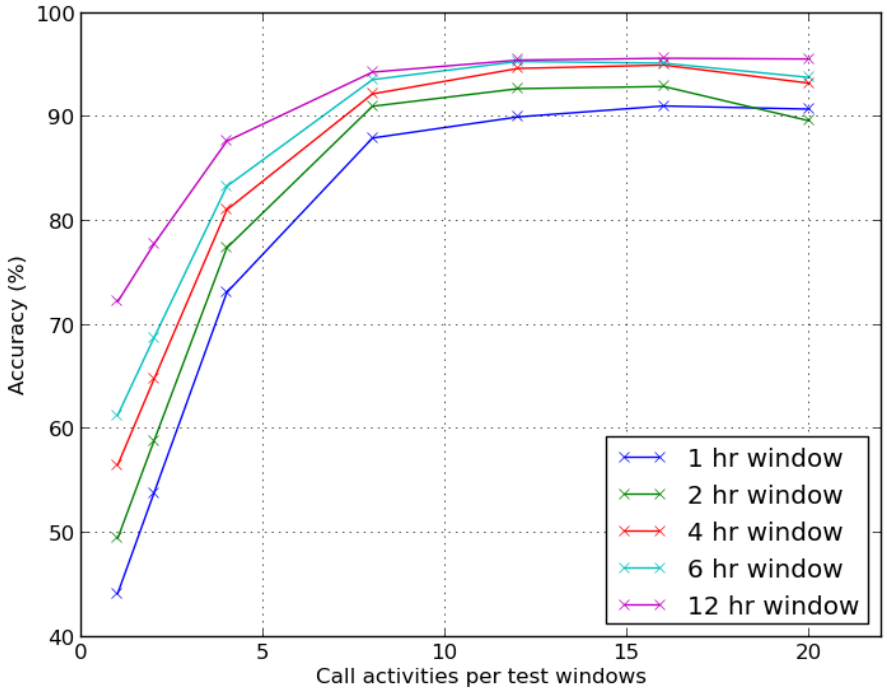


Figure 4.6: Variation of accuracy of call patterns model with activity threshold (for different time window sizes).

Model Parameter Tuning

In this experiment the effect of the testing data window size and the event threshold on the call patterns model's accuracy at detecting anomalies was investigated. Increasing the activity threshold increases the information the model receives about the activities, therefore, increasing *classification accuracy*. On the other hand, increasing the activity threshold for a particular window size reduces the probability of a window containing the sufficient activities to qualify for anomaly detection, reducing both *accuracy* and *coverage*. The results from these experiments are shown in Figure 4.6.

All of the curves in Figure 4.6 show an increase in accuracy, as expected, when the activity threshold is initially increased. As the activity threshold is increased further the curves representing the smaller time windows show a decrease in accuracy. This drop in accuracy is due to only a small amount of data points from the testing set qualifying for classification and this creates a bias in the results.

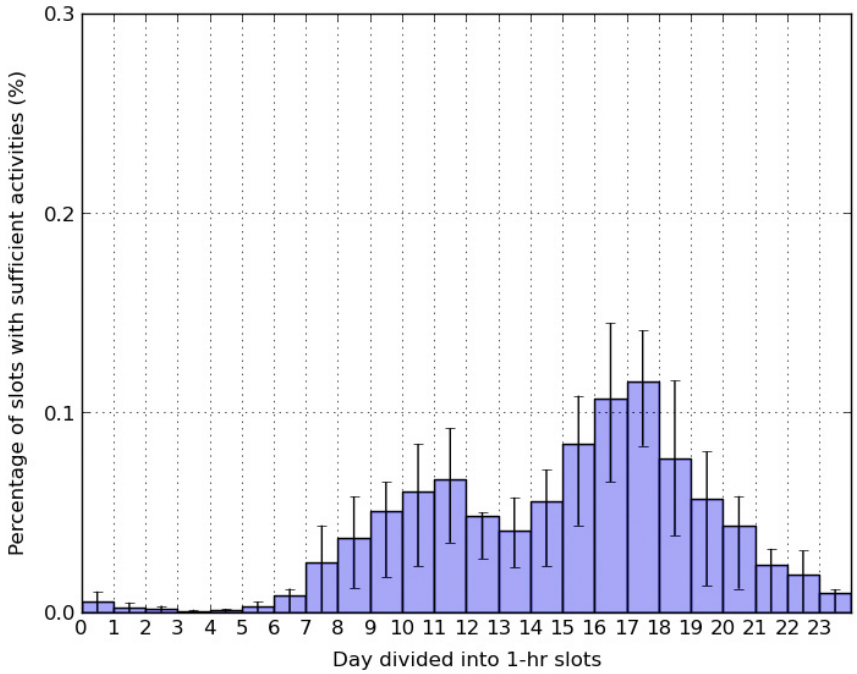


Figure 4.7: Coverage map for call patterns model.

Model Coverage

The coverage map depicts the likelihoods for a user generating sufficient data (during each hour of a day) for a model to make an anomaly classification. See Subsection 2.4.2 for a detailed description of the statistics calculation procedure. The coverage likelihoods averaged across all 30 users for the call patterns model are shown in Figure 4.7.

Once again, as with the message response patterns model, these experiments indicate that the amount of time during each day where there is sufficient data is somewhat low for the call patterns model. Therefore it would be ideal for this model to be combined with some other model to give more complete coverage of the day.

This chapter described some of the methods utilized to create models of an aspect of user behavior. From experimental results it was shown using purely soft-sensor information does not inhibit a behavior aspect models ability to effectively detect anomalies in the user's routine. The next chapter looks at radio channel information source based models.

Chapter 5

Modeling Aspects of User Behavior using Radio Information

This chapter expands on the notion of using radio channel information to model aspects of the user's behavior. It presents two models that utilize the radio channel information to model the mobility of the user [9]. This chapter also discusses experimental results on Dataset II evaluating the coverage and anomaly detection performance of these models.

5.1 Outdoor Mobility Model

In this section, an alternative model to the geo-trace model (see Chapter 3) is presented as an example of using radio information for modeling the user mobility behavior. The data source for this model is GSM cell tower information. While this model is capable of working indoors as well, as long as GSM cell reception is available, it is tagged as an outdoor mobility model. This is because the model requires the user to move a significant distance within a GSM cell to generate a new signature, which happens mostly when the user is outdoors. This model is based on the same hypothesis as the geo-trace model [10], that a user's next location is dependent largely on present location and a finite number of past locations and these sequences of dependent locations

can be combined to create a unique signature for a user's routine behavior. Unlike the geo-trace model, this model does not depend on power hungry GPS chips for modeling data: instead it uses readily available GSM tower information to localize the user. While GSM tower-based localization has a lower granularity than GPS localization, the power savings and ability to work even when the device is in covered areas offsets this disadvantage.

An n-gram model is also used to learn and monitor movement patterns as in Section 3.3. Since the n-gram model deals with labels, the GSM tower signal data are converted into discrete labels. To convert the GSM tower signal data into labels, the ID of the tower the device is currently connected to is used, and the list of towers visible to the device are discarded (experimental results show the use of the entire tower list with only a few weeks of training causes an increase in the true-positive rate at the expense of a *drastic* increase in the false-negative rate by making the location signature overly sensitive). Next, the signal strength of the connected tower is quantized, using a quantization scale with unique and arbitrary labels associated with each level in the scale. Finally the quantized signal strength's level label and the tower ID are concatenated to form a unique label indicative of the user's present location. For this model, each such label is termed as an area-label (a_i) and a sequence of labels is termed an area-trace ($A = a_1, a_2, \ldots, a_N$).

The n-gram movement model consists of probabilities for short area-traces ($a_1, a_2, \ldots, a_n$). These probabilities can be estimated using Maximum Likelihood Estimation (MLE) on the training data by counting the occurrences of area-labels and area-traces:

$$P(a_1, a_2, \ldots, a_N) = \prod_{i=1}^{N} P(a_i | a_{i-n+1}^{i-1}) \tag{5.1}$$

$$P_{\text{MLE}}(a_i | a_{i-n+1}^{i-1}) = \frac{C(a_{i-n+1}, \ldots, a_{i-1}, a_i)}{C(a_{i-n+1}, \ldots, a_{i-1})} \tag{5.2}$$

However, MLE assigns zero probabilities to all area-traces that do not occur in the training data. Again, to overcome this "closed-world" assumption [43] Good-Turing discounting and

Katz backoff smoothing [75] are applied (see Subsection 3.3.1). The trained n-gram model can then be used to estimate the likelihood of an area-label, a_i, given the previous $n-1$ area-labels from a user's context as,

$$P(a_i|a_{i-n+1}, a_{i-n+2}, \ldots, a_{i-1}) = P(a_i|a_{i-n+1}^{i-1}) \tag{5.3}$$

Again, label collapsing (at the resolution of the GSM signal strength data) described in Section 3.3 is applied to ensure location transitions are modeled (instead of time spent at locations).

The trained movement pattern is used for detecting anomalies in behavior by continuously feeding the model area-labels in real-time. The n-gram model then outputs a likelihood estimate for the current area-label being associated with the user, given the area-trace seen up to that point in time. The probability estimate is compared against a heuristically decided threshold, *anomaly threshold*, (see Figure 5.1 and description), and if the estimate is below the threshold, anomalous behavior is flagged.

5.1.1 Model Evaluation

As with the models from the previous chapter, the evaluations of the outdoor mobility model were also performed using Dataset II (see subsection 2.3.2) using all available data in the dataset. Again, a 3-fold validation approach was used for each user, where one fold (*i.e.*, one month of data) is used to train a model to be tested. The other two-thirds of the user's data are used as testing data of *normal behavior* and to simulate *anomalous behavior* the data traces from the other 29 users were used. This entire process is repeated for each fold of the 30 users in the dataset and the results presented here are the averaged results from these experiments. This experimental procedure assumes that 1) during the data collection period each user did not have any substantial anomalies in their own behavior, and 2) the anomalies that are of interest are represent a change in behavior to the extend that the movement patterns would mirror the movement of a different user.

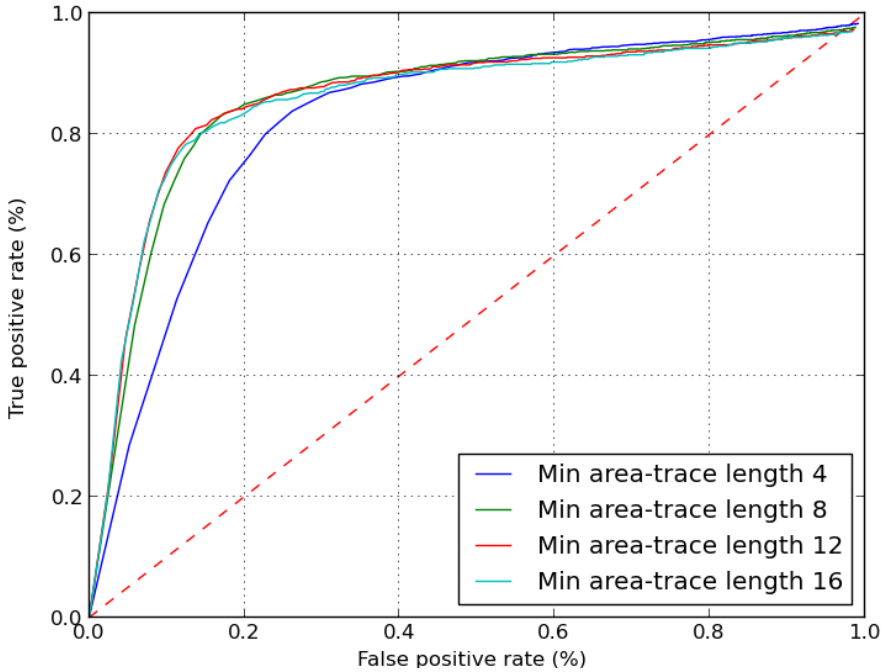


Figure 5.1: ROC curves for the outdoor mobility model.

Receiver Operating Characteristic (ROC) curves

In this experiment, the anomaly threshold (*i.e.,* the model's sensitivity) was adjusted to investigate its effect on the false-positive rate (FPR) and the true-positive rate (TPR). The results are plotted to form the receiver operating characteristic (ROC) curve for the model (see Subsection 2.4.1 for a full description of ROC curves). For the outdoor mobility model, four separate ROC curves were generated where the minimum length of an area-trace (number of area-labels in a trace) the model has to see before it performs a classification is varied from 4, 8, 12, and 16 labels (see Figure 5.1). This figure makes it clear the trade-off between activity threshold / time window size, TPR and FPR. All ROC curves are generated using 10-gram mobility models.

The ROC curves in Figure 5.1 indicate the outdoor mobility model can be used to effectively detect anomalies (with over 85% TPR at a 20% FPR). As expected, the model's performance improves as more prior data (*i.e.*, longer area-traces) are available, but the model is able to perform quite well even with short area-traces. While ROC curves for the outdoor mobility models are generated for various history lengths, the history lengths are loosely correlated with

time, as a new tag is added to a history when the user has moved sufficiently. The amount of movement required to add a new tag in the outdoor movement model is when the user moves sufficiently for the GSM cell towers signal strength to change to a different level of quantization.

Model Parameter Tuning

In this experiment, the impact of the area-trace history considered by the outdoor mobility model on the accuracy of detecting anomalies was explored. There are two aspects to the area-trace history considered by the models. First, and the more straight-forward, is the minimum number of area-labels that the model needs to see before making a classification (*i.e.*, trace-length-threshold). The second is the length of the longest segments of area-traces modeled, *i.e.*, the value of n in the n-gram model. Increasing either of these would increase the amount of information the model has and, therefore, it should increase the accuracy of the classification. While increasing the trace-length-threshold reduces the likelihood of a user having sufficient data within a time window for classification, increasing n increases the model size that has to be held on the device and the training data requirements.

The results from this experiment (see Figure 5.2) show that as the history considered by the model is increased, the accuracy increases as expected but with diminishing gains. This indicates that the importance of a prior location on a user's next location diminishes as the prior location is further back in the area-trace. It was observed that increasing n to 20 keeps the model size under 10MB. Since retaining a model of this scale on a modern mobile device is not a strain on resources, model size does not impose any major restriction on history size. On the other hand, the training data requirement increase showed that a 1 month long training dataset did not contain sufficient data to train a model with $n > 12$, causing the training process to use discounting to estimate the likelihoods of longer area-traces. Using the results of this experiment, it was concluded that $n = 10$ for the outdoor mobility model provides a reasonable level of accuracy in return for a plausible training data requirement. As the model is used in the real-world over a

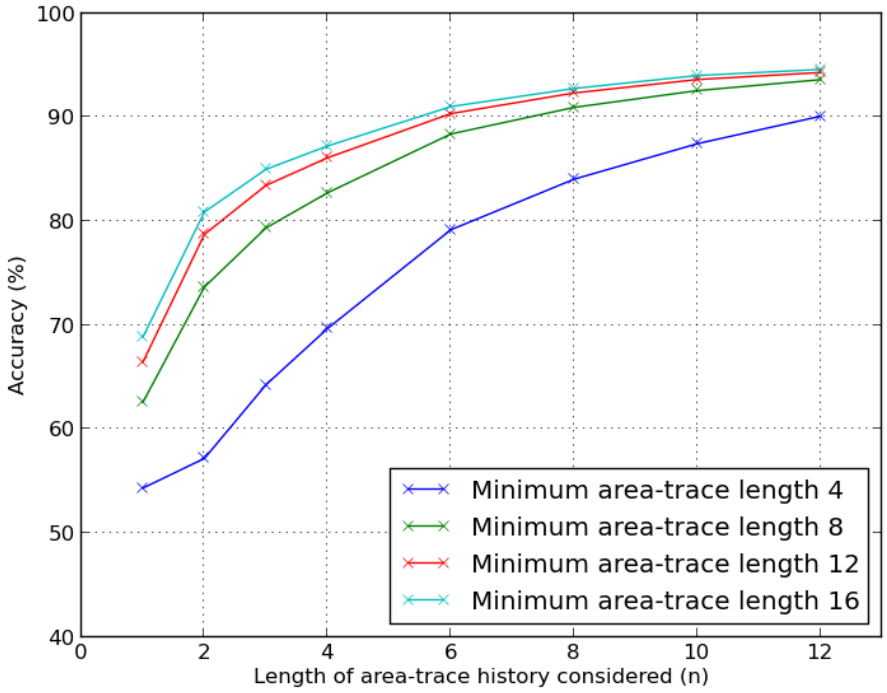


Figure 5.2: Variation of accuracies of outdoor mobility model vs. length of trace history considered (i.e., n).

period of time it can be retrained with data accumulated while in use to increase the size of the area-trace history it can consider.

Model Coverage

In this experiment, the likelihoods for a user generating sufficient data by moving around for the indoor mobility model to detect anomalies is investigated. See Subsection 2.4.2 for a detailed description of the coverage map calculation. The coverage likelihoods averaged across all 30 users for the outdoor mobility model are shown in Figure 5.3. For the outdoor mobility model a 1-hour window requires at least two labels to qualify the slot as having generated sufficient data for the a model to make an anomaly classification. The qualifying threshold for the model was selected with the aid of the ROC curve such that the model has at least 75% TPR at a 20% FPR.

The outdoor mobility model, while having greater coverage, is still not uniformly adequate through out the day as well across users. It would be ideal for this model to be combined (with

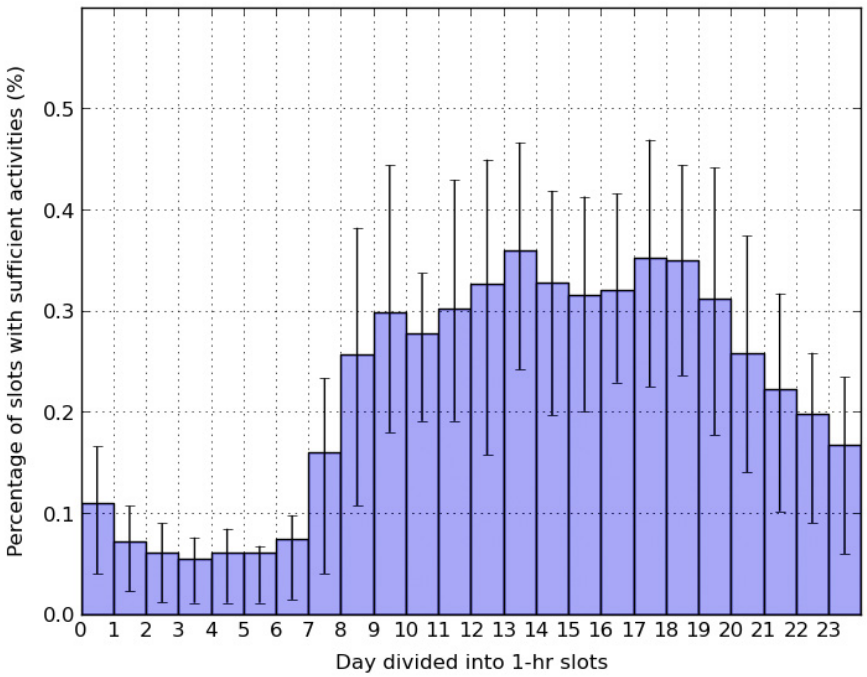


Figure 5.3: Coverage map of the outdoor mobility model.

other models) to give more complete coverage of the day.

5.2 Indoor Mobility Model

This section presents an approach to model the indoor movement patterns of a user's behavior using WiFi signal data. This model is based on the same hypothesis as the outdoor mobility model (see Section 5.1), that a user's next location is dependent largely on his present location and a finite number of past locations and these sequences of dependent locations combine to create a unique signature for each person. The model is designed to utilize WiFi information that is periodically scanned and gathered (*i.e.*, WiFi signatures) by the mobile device as part of its routine usage without incurring a sensing power penalty. Even though previous work [45] has shown that the granularity and accuracy of WiFi signature-based localization has less granularity and accuracy than GPS-based localization, it has the distinct advantage of working in WiFi-equipped indoor environments. Without performing explicit localization, a hypothesis

behind this model is that WiFi signatures are sufficient to perform anomaly detection with high accuracy. While this model is termed as an indoor mobility model, it is applicable in any area with WiFi coverage (*e.g.*, large corporate or college campuses).

This model, like the outdoor mobility model, uses an n-gram model to learn and monitor movement patterns. As the n-gram model deals with labels, the WiFi signature information has to be converted into discrete location labels. A WiFi signature consists of a list of WiFi access points (APs), the signal strength of each AP, and a list of networks to which each of the APs belong. While a straight-forward approach would be to assign a unique label to each WiFi signature, this approach would make the labels extremely susceptible to minor variations in the environment (*e.g.*, changes in signal strength caused by changes in weather, an AP disappearing). To make the label assignment more robust, a Hamming distance-based hierarchical clustering approach is used to group together signatures which are from the same or close-by locations and assign each cluster a unique label. This label creation approach is possible as the model is interested only in the user's transitions between *locations* and not in the precise location.

The Hamming distance based clustering approach works as follows. Each signature is assigned a Hamming code, a binary string where each bit represents the networks seen in the training data and the bit is set to true if the network is seen in the current signature (the signal strength data is discarded to make the labeling scheme less susceptible to noise at the expense of localization granularity). Using the Hamming distance between the signatures as the distance metric, hierarchical clustering is used to produce a binary tree of clusters where the leaf nodes are the individual WiFi signatures. Next the cluster tree is traversed depth-first and if the distance between the two child clusters is greater than a threshold (chosen based on the required size of the vocabulary), the child clusters are assigned separate unique labels. In this model, each unique label is termed a region-label and a sequence of labels a region-trace. The region-label assignment to new WiFi signatures happens by finding the cluster to which the signature's Hamming distance is closest and then assigning the cluster's label to the signature.

The n-gram movement model consists of probabilities for short region-traces $(r_1, r_2, \ldots, r_N)$ which are estimated using Maximum Likelihood Estimation (MLE) on the training data (see Section 3.3 for a full description of training n-gram models). The trained n-gram model can then be used to estimate the likelihood of the next region-label, r_i, given the previous $n-1$ region-labels from a user's context with the outdoor mobility model, the number of times the same region-label can consecutively recur is limited (which indicates that the user is stationary at least at the level this model is concerned with). This limitation is applied when the indoor mobility model attempts to model the user's location transitions, and not the amount of time the user spends at each location, since repetitive region-labels could push significant transitions past the n-gram models horizon.

The trained indoor mobility model is used for anomaly detection by continuously feeding the model region-labels in real-time. The n-gram model then outputs a probability estimate for the current region-label being part of the user's learned region-trace model, given the previously seen real-time region-labels. The probability estimate is compared against a heuristically decided threshold, *anomaly threshold*, (see Figure 5.4 and description), and if the estimate is below the threshold an anomalous behavior flag is raised.

5.2.1 Model Evaluation

As with the previous model presented in this chapter, the evaluations of the indoor mobility model were also performed using Dataset II (see subsection 2.3.2) using all available data in the dataset. As with the previous models, a 3-fold validation approach was used per-user. One month of data was used to train a user's model and the other two-thirds of the user's data (*i.e.,* 2 months of data) are used as *normal behavior* data for testing. To simulate *anomalous behavior* the data traces from the other 29 users were used. This entire process is repeated for each fold of the 30 users in the dataset and the results presented here are the averaged results from these experiments. As with the outdoor mobility model's evaluation procedure, it is assumed that 1)

during the data collection period each user did not have any substantial anomalies in their own behavior, and 2) the anomalies that are of interest are represent a change in behavior to the extend that the movement patterns would mirror the movement of a different user.

To test the WiFi signatures across users, the WiFi signature clustering and cluster labeling sections of each user are used on their own dataset to generate the sequences of cluster labels and then share these sequences across users. The labeling of clusters is done from a shared list of labels. The case of major variations from routine where the user leaves known areas is straight-forward to detect, but these experiments simulate harder to detect cases where the user has broken from routine but remains within known areas (*e.g.,* within the user's college-campus).

Receiver Operating Characteristic (ROC) curves

As with the outdoor mobility model from the previous section, the indoor mobility model's anomaly detection sensitivity was adjusted to investigate its effect on the false-positive rate (FPR) and the true-positive rate (TPR) of the model (see Subsection 2.4.1 for a more detailed description of ROC curves). Five separate ROC curves were generated where the minimum length of a region-trace is varied from 1, 2, 4, 8, and 10 using a 10-gram indoor mobility model (see Figure 5.4).

The ROC curves with minimum trace lengths of 8 and 10 in Figure 5.4 indicate that the indoor mobility model is effective at anomaly detection (with over a 90% TPR at a 20% FPR) when presented with longer histories of movement. The degradation in performance for this model is quite low (remaining over 80% TPR at 20% FPR) when the model sees at least the present and previous (*i.e.,* 2) region-tags. As with the outdoor mobility model, ROC curves for the indoor mobility models are generated for various history lengths, which lengths are loosely correlated with time. A new tag is added to a history when the user has moved sufficiently. For the indoor movement model the amount of movement required is much less than that of the outdoor mobility model, in most cases out of or in to the range of 1 or more WiFi access points.

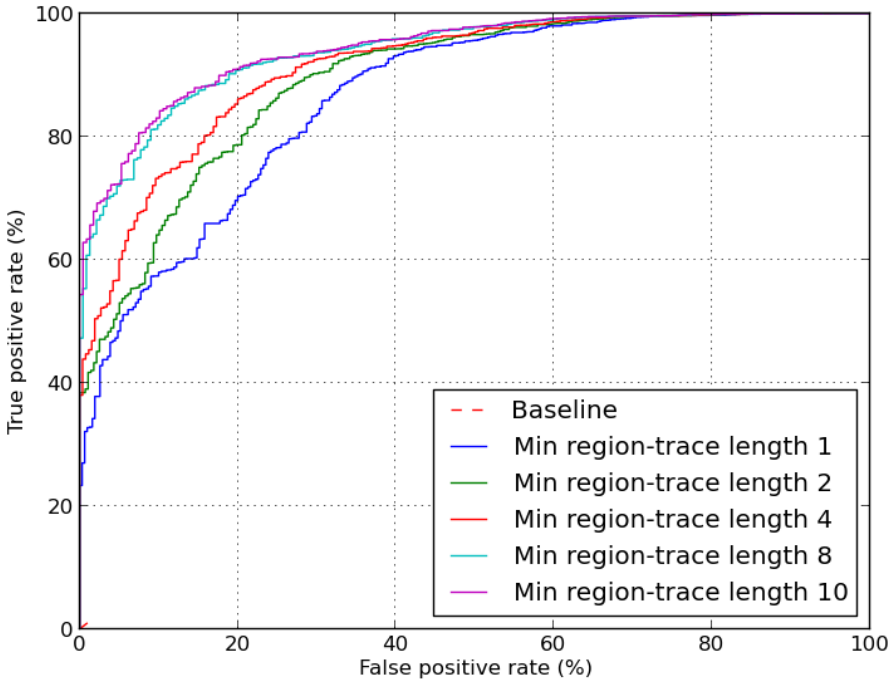


Figure 5.4: ROC curves for the indoor mobility model.

Model Parameter Tuning

In the next experiment, the impact of the area-trace history considered by the indoor mobility model on the accuracy of detecting anomalies is investigated. There are two aspects to the area-trace history considered by the model. *I.e.*:

1. The minimum number of region-labels that the model needs to see before making a classification (*i.e.*, trace-length-threshold).
2. The length of the longest segments of region-traces modeled, *i.e.*, the value of n in the n-gram model.

Increasing either of these two would increase the amount of information seen by the model and, therefore, it should increase the accuracy of the classification, however, increasing the trace-length-threshold reduces the likelihood of a user having sufficient data within a time window for classification, increasing n increases the model size that has to be held on the device and the training data requirements.

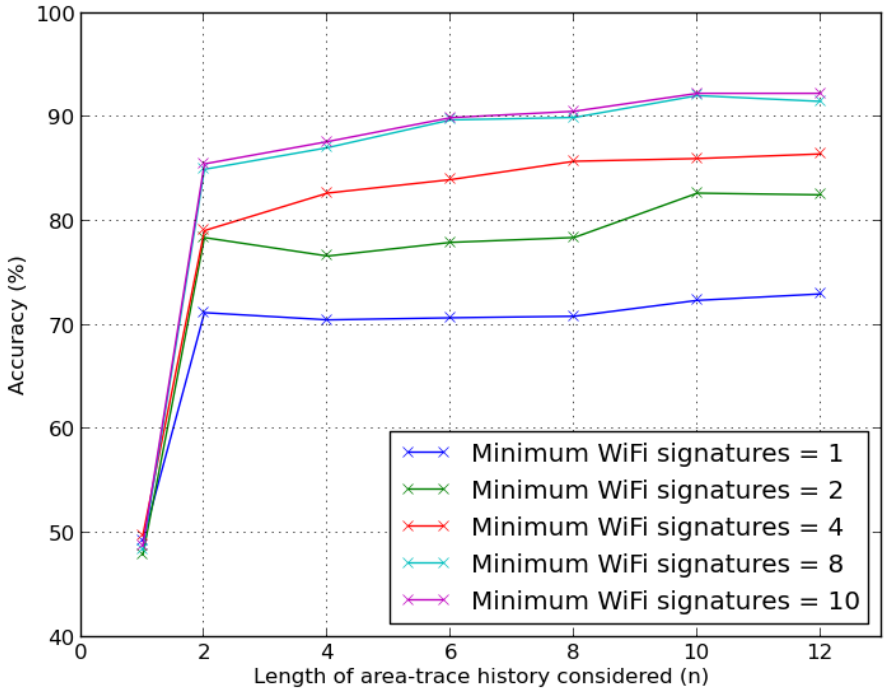


Figure 5.5: Variation of accuracies of indoor mobility model vs. length of trace history considered (i.e., n).

The results from this experiment (see Figure 5.5) show that as the history considered by the model is increased, the accuracy increases as expected but with diminishing gains, indicating the importance of a prior location on a user's next location diminishes as the prior location is further back in the region-trace. Even with $n = 20$ the indoor mobility model size remained under 20MB, and since retaining a model of this scale on a modern mobile device is not a strain on resources, model size does not impose any major restriction on history size. On the other hand, the training data requirement increase showed that a one-month long training dataset did not contain sufficient data to train a model with $n > 12$, causing the training process to use discounting to estimate the likelihoods of longer region-traces. As with the outdoor mobility model, the experimental results were used to conclude that $n = 10$ provides a reasonable level of accuracy in return for a plausible training data requirement.

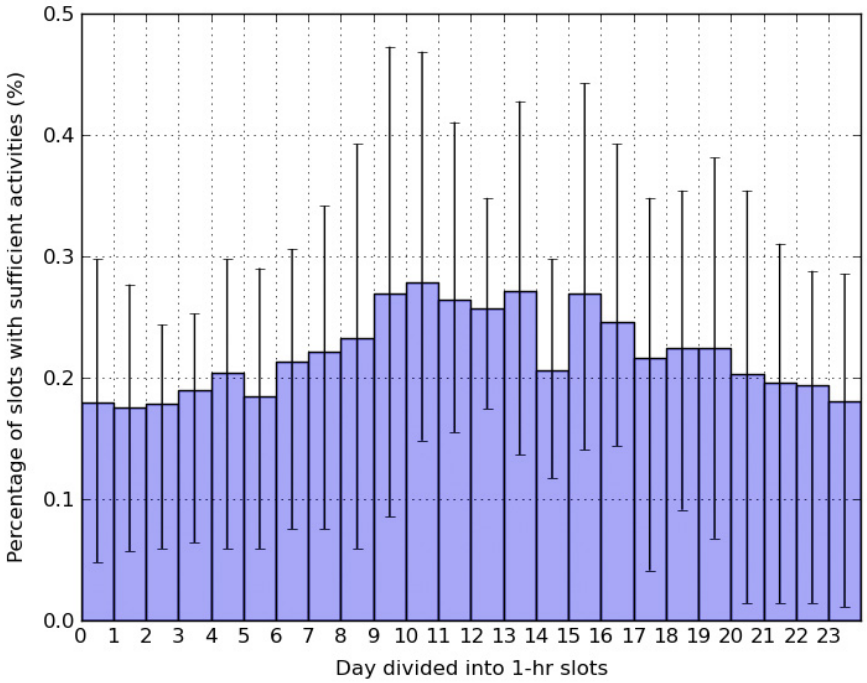


Figure 5.6: Coverage map of the indoor mobility model.

Model Coverage

See Subsection 2.4.2 for a detailed description of how the coverage map for a model is generated. The coverage likelihoods averaged across all users in the dataset for the indoor mobility model are shown in Figure 5.6. For the indoor mobility model a 1-hour window requires at least five labels to qualify the slot as having generated sufficient data for the a model to make an anomaly classification. The qualifying threshold for the model was selected with the aid of the ROC curve such that the model has at least 75% TPR at a 20% FPR.

The indoor mobility model has a lesser, but more uniform coverage throughout the day than the outdoor mobility model As with the outdoor mobility model, there is a large variation in coverage across users (evidenced by the large error bars in Figure 5.6). Once again, it would be ideal for this model to be combined with other models to give more complete coverage of the day.

The experimental results from this chapter and the previous chapter on using soft-sensor

information for modeling show that while each model provides reasonable accuracy (see Figures 4.2, 4.5, 5.1, and 5.4), the coverage of these models is quite low (see Figures 4.4, 4.7, 5.3, and 5.6). The next chapter presents the idea of combining multiple behavior aspect models to improve not only coverage but overall anomaly detection performance.

Chapter 6

Straightforward Combining of Models

The previous two chapters on using soft-sensor information and radio channel information for modeling aspects of behavior show that using these information sources provides reasonable accuracy (see Figures 4.2, 4.5, 5.1, and 5.4), but it was also seen that the coverage of these models is quite low (see Figures 4.4, 4.7, 5.3, and 5.6). This chapter explores the usefulness of combining different behavior aspect models. The combining approach presented in this chapter is straightforward (naive) voting approach with the aim of improving overall anomaly detection performance and coverage; a later chapter gives a more sophisticated method. This chapter also discusses experimental results of the contrasting the performance and coverage of a combination of models vs. the geo-trace model.

This straightforward approach of combining the behavior models was setup for evaluation as follows. Each model was configured to detect anomalies, using up to one hour of data. Each model presents its vote towards the final decision only if it has sufficient data to make a prediction. In the case of tied ballots, the decision of the outdoor movement model, the indoor movement model, the message response model are statically weighted to take precedence in descending order. This static order of precedence was decided by comparing the Receiver Operator Characteristic curves of the models (see Figures 4.2, 4.5, 5.1, and 5.4), *i.e.,* precedence ordered by the area each model has under its ROC curve.

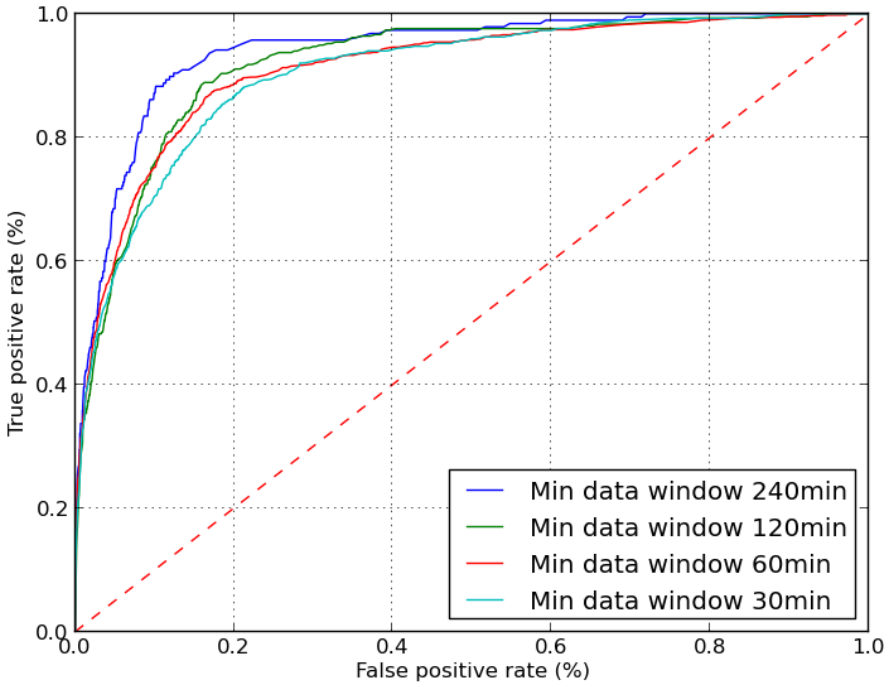


Figure 6.1: ROC curves for the baseline geo-trace model on Dataset II.

To evaluate this combination of models, again, Dataset II used for evaluating the individual models is used. The testing data from each stream (when available) are grouped into 1-hour chunks of each day and provided to the combination of models for classification.

6.1 Baseline : Geo-trace Model

To compare and contrast the performance and coverage of the combined behavior models, the geo-trace model (see Chapter 3) is used as a baseline. The geo-trace model was recreated with Dataset II (see Subsection 2.3.2) using the equal partitioning method with 10-gram language models [10]. The baseline results for this model are created using the same evaluation criterion used for the individual models described in this chapter. The resulting ROC curves for the GPS mobility model is shown in Figure 6.1. Since the combined behavior models are used to detect anomalies in one hour chunks of user data, the GPS mobility model's ROC curve with the one hour testing window is of particular interest for comparison.

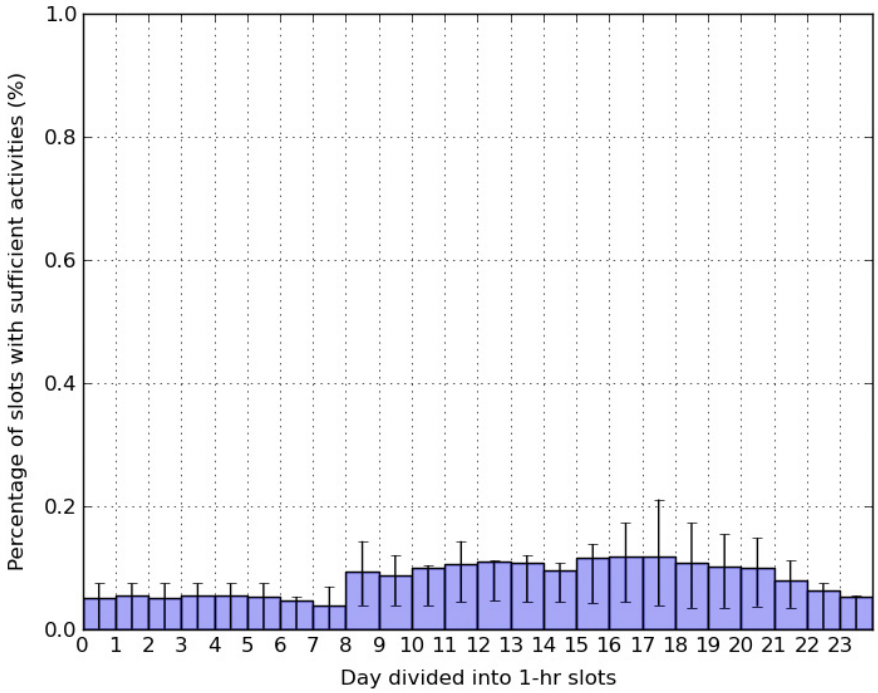


Figure 6.2: Coverage map for the geo-trace model on Dataset II.

A coverage map for the GPS mobility model was also generated, showing the likelihood for an average user from our dataset to have sufficient data to perform anomaly detection (see Figure 6.2).

6.2 Receiver Operator Characteristic (ROC) Curves

Since the combination of the four models presented in this chapter has four anomaly thresholds (one for each model in the combination) that can be varied, the ROC curve is a hyper-surface if all possible threshold values are iterated over. For comparison against the geo-trace model, all four threshold values were linearly varied through their ranges to generate a *slice* of the ROC hyper-surface (see Figure 6.3). When contrasting these ROCs against that of the geo-trace model with a 1-hour time window, the combination of models' ROC curve *outperforms* the geo-trace model. The ROC-curve for for the combination of models with the 30 minute test window shows *slightly* higher performance indicating that the combined models can perform at reasonable accuracy

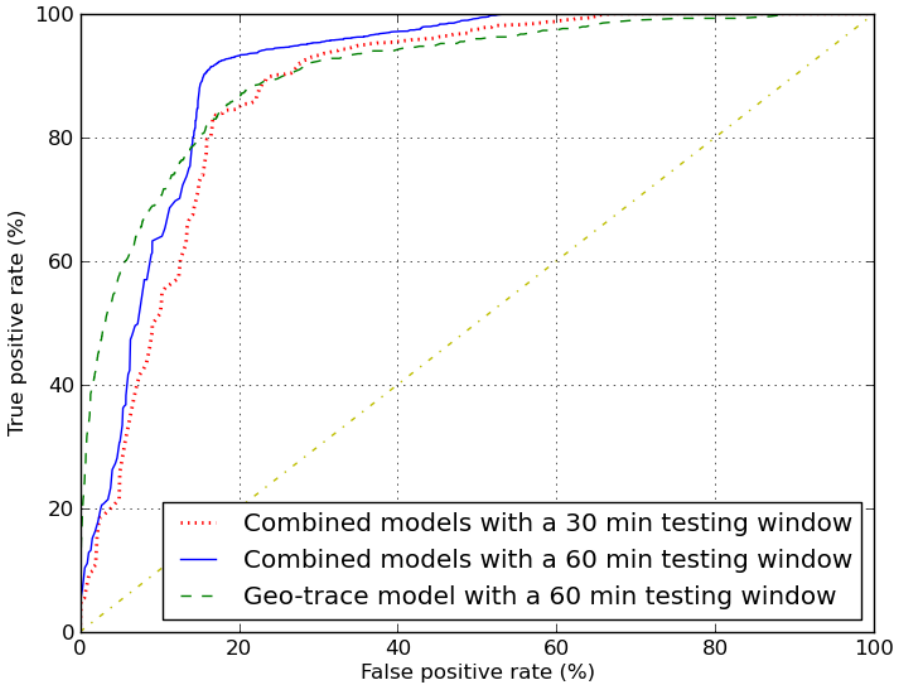


Figure 6.3: ROC curves for the combined models, contrasted with the baseline geo-trace model.

even at a fine time granularity. A summary of the results is presented in Table 6.1, which shows a 7.6% improvement in the total anomalies detected at a FPR of 20% over the GPS-based system.

6.3 Coverage Maps

A coverage map for the combined models was also generated, showing the likelihood for an average user from the dataset to have sufficient data to perform anomaly detection (see Figure 6.4). Contrasting this against the geo-trace model's coverage map, it is clearly visible that the combined models have *much higher* coverage. A summary of the coverage results is presented in Table 6.1 showing coverage increase throughout the day of more than 5 times over the GPS-based system.

These evaluations indicate that even by straight-forward combinations of models of user behavior aspects improves not only coverage, but also the accuracy over using a single model. A further few experimental results are presented next to show that these increase may even be

Table 6.1: Summary of results comparing the combined behavior models against the geo-trace mobility model.

	Combined Models (60 min. window)	**Geo-Trace Model**
TPR (at 20% FPR)	93.18%	86.58%
FPR (at 80% TPR)	14.20%	15.07%
Coverage (Likelihood)		
0000 - 0800	21.96%	5.02%
0800 - 1200	46.89%	9.66%
1200 - 1800	56.86%	11.07%
1800 - 0000	47.74%	8.37%
Overall	41.28%	8.14%

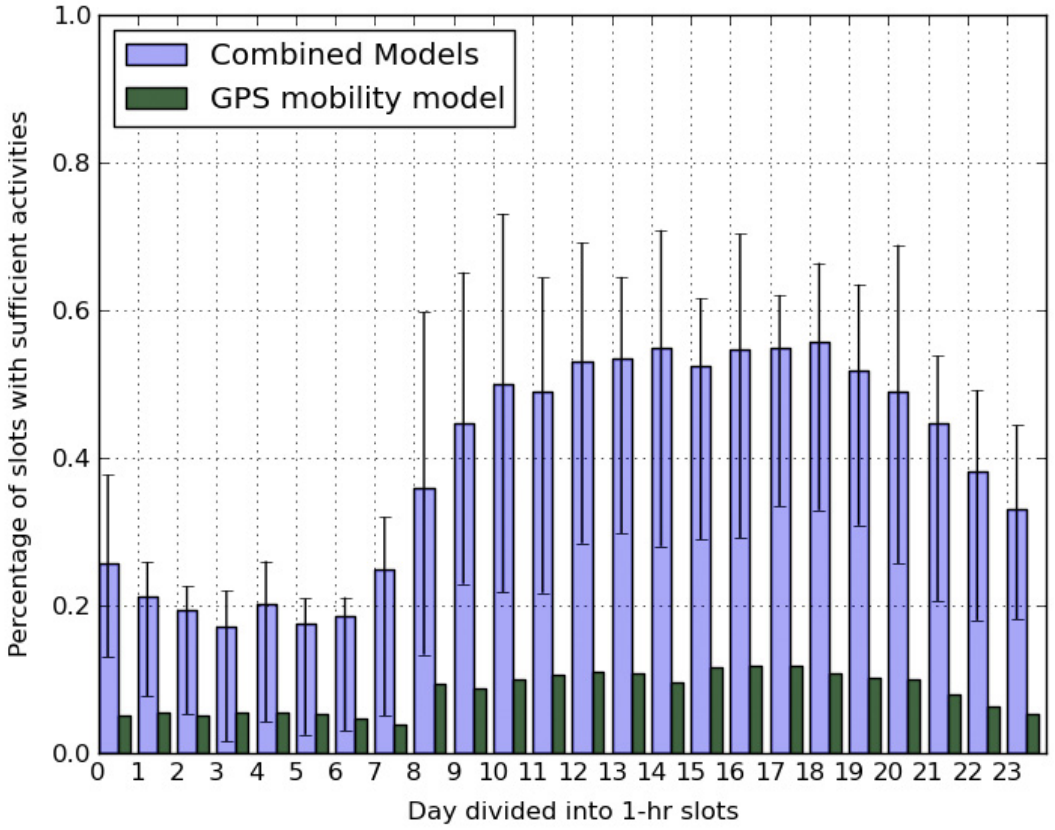


Figure 6.4: Coverage maps of the combined models and the geo-trace mobility model.

achieved at lower computational and power costs.

6.4 Computational Complexity of the Combination of Models

In this section, the computational complexities of evaluating the four models used in the combination of models is described. The computational complexities of training models as the training operations are performed off-line, *i.e.,* when the phone is docked and charged with the user's home computer.

The complexity approximations for the message response model and call pattern model are similar in nature, therefore a single generic calculation is presented. Assume a contact list of length n and m events (*i.e.,* either call events or messaging events) in the time-window used for anomaly detection. Parsing the m-event log, past-to-present, keeping track of the last missed call / message (if any) from each number present in the log, presents a complexity of $\mathcal{O}(m)$. Looking up the model parameters for each contact from a sorted list of length n presents a complexity of $\mathcal{O}(\log_2(n))$, while sampling a model of Gaussian mixture models is constant time. Therefore, the complexity of evaluating either model is $\mathcal{O}(m \log_2(n))$. In Dataset II, $m < 30$ and the longest contact list was $n = 1400$, *i.e.,* a very manageable computational load.

In the case of the outdoor mobility model, assume the n-gram model has a vocabulary of s words (formed by concatenating cell tower IDs and discretized signal power-levels). By representing the n-gram model as a tree with a branching factor of s, obtaining the likelihood of a particular n-gram (assuming the n-gram is present) is $\mathcal{O}(n \times s)$. If the n-gram is not found, back-off smoothing requires the likelihood of a $(n-1)$-gram subsequence. This process repeats until a subsequence is found in the model, and in the worst case it repeats until a unigram is reached. The querying complexity in the worst case is $\sum_{i=1}^{n}(i \times s) = (n+1).(n/2).s$. Since n is a constant value, the complexity is $\mathcal{O}(s)$ (where $s < 1000$ for these experiments).

The indoor mobility model's n-gram look-up complexity remains constant-time, unlike the outdoor mobility model, as the vocabulary size is fixed at training time by selecting the number of

Table 6.2: Power consumption (in milliwatts) comparison between the combined (non-GPS) behavior models and the geo-trace mobility model.

		Average (mW)	St. Dev. (mW)
Combination of Models		0.0419	0.0178
GPS Model	Sensing	175.0028	35.0250
	Processing	0.0356	0.0136
	Total	175.0384	35.0386

WiFi signature clusters (see Section 5.2). This model's complexity is dominated by the process of converting a list of observed networks to a "word" in the vocabulary. If the number of networks in the model are m and the last scan retrieved a list of p networks, creating a bit-vector used to look-up the word, in this implementation, has $\mathcal{O}(m.p)$ complexity. While there is clearly room for optimization, implementing these improvements were not considered a priority since in these experiments $m < 1000$ and $p < 25$.

6.5 Power Consumption

Presented next are the results of a power consumption experiment to contrast the power usage of the combination of models with the geo-trace model. The experiment was conducted with 5 students, each carrying an Android Nexus One device loaded with an application running the combination of models and a separate application running the geo-trace model. The applications were loaded with models trained off-line (*i.e.,* on a desktop computer).. The power consumption of each application was monitored using PowerTutor [76] and the overall results are presented in Table 6.2. As the table shows, the amount of power consumed by the combination of models is approximately 3000X less than the geo-trace model.

The final evaluation shows that even simple combinations of models of user behavior aspects result in significant performance improvements and largely increases coverage. The next chapter presents a more advanced approach to combine the output of individual classifiers which greatly

improves the overall accuracy.

Chapter 7

CobLE : Confidence-based Learning Ensembles

This chapter discusses Confidence-based Learning Ensembles (CobLE) [8], a novel approach for combining the output from multiple behavior aspect models. A formal description and theoretical analysis of the CobLE approach is presented. This chapter also presents results of extensive experiments comparing CobLE to other related approaches (including the straightforward voting scheme from Chapter 6), demonstrating significant improvement over existing ensemble learning and data fusion approaches, especially with asynchronous data sources.

7.1 Feature Fusion and Decision Fusion

Fusing data from a variety of different information sources observing a phenomenon has been shown to vastly improve the accuracy of classifying the phenomenon [71]. In most real-world applications the different information sources are heterogeneous and asynchronous and the amount of training data available tends to be limited in quantity (*e.g.,* activity recognition, user behavior prediction or anomaly detection using mobile phone sensors such as accelerometer, GPS, microphone, where the amount of training data per-user is limited). Fusing data from asynchronous

information sources creates sparse feature sets and coupled with limited amounts of labeled data the difficulty of training accurate classifiers increases.

Previous approaches to perform classification using information from a variety of sources can be broadly categorized as belonging to (or being hybrids of) two broad classes [34, 52], *i.e.,* 1) feature fusion - approaches combining features extracted from different information streams to form a single feature set before applying a classifier [65, 68], 2) decision fusion - approaches running separate classifiers on each individual information source and aggregating the classifications (*i.e.,* ensembles of classifiers) [57, 69, 74]. The feature fusion approaches of combining the information streams to form a single feature set before applying a classifier greatly increases the dimensionality of the feature space. Even with ample training data per-information source to train a classifier on each source independently, the combination of the datasets tends to be insufficient to train a classifier on the higher-dimensional fused feature set. Fusing a feature set from asynchronous information sources results in missing features from each source that is out of synchronization when a sample is logged.

Classifier ensemble approaches apply a separate classifier to each information source and their classifications are combined using weighted voting (*i.e.,* decision fusion). The weight of each classifier is either: 1) predetermined at training [12, 60, 70], or 2) dynamically based on the posterior probability output by each classifier [49, 56]. When weights are based on posterior probabilities output by the base classifiers, the ensemble is limited to using only base classifiers that can output posterior probabilities, *i.e.,* excluding classifiers such as decision trees. Statically weighting classifiers operating over different feature spaces at training time forces the assumption that each classifier operates with the same accuracy (relative to other classifiers in the ensemble) over its entire feature space (or uniformly over regions of its feature space [60]).

This chapter presents Confidence-based Learning Ensembles (CobLE), a learning approach where an ensemble of heterogeneous classifiers, each working on a subset of input features, uses a dynamically-weighted voting scheme for classification. A dynamic-weight for a vote corre-

sponds to a confidence measure of the vote's classifier in classifying a data point in that region of the input feature space. This ensemble technique is based on the observation that learning algorithms tend to perform well in certain regions of the input feature space, independent of the amount and quality of training data [58]. Based on this observation, a CobLE approximates the confidence of a classifier for regions of the feature space it operates on using a confidence function (*i.e.,* topographical map over the feature space of a classifier where the "height" at a point in the feature space maps to the classifier's confidence for classifying that point). The final classification of the ensemble is decided by the normalized-weighted voting of all of the participating classifiers (a classifier abstains from voting if part of its input feature space is missing in the current data point). By limiting each classifier to a feature set derived from a few number of information sources, each classifier's feature space has drastically smaller dimensionality compared to an aggregated feature set derived from all of the available information sources, thus reducing the training data required to train the classifiers. Additionally each classifier's confidence function is also approximated for cases where one or more features are missing, making the ensemble robust against the asynchronism of the information sources.

The next section describes the implementation of CobLE used in this dissertation and explains the training and classification procedure.

7.2 CobLE : Confidence-based Learning Ensembles

This section describes how a CobLE is trained and utilized for classification. A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are each accurate and diverse [35]. An accurate classifier is one that has an error rate that is better than random at guessing the classification. Two classifiers are diverse if they make different errors on the same data points. Since each base classifier in CobLE operates on feature sets derived from unique asynchronous information sources, the classifiers can be assumed to have the necessary diversity. Therefore the only requirement is that

each classifier is optimized to provide relatively high accuracy. For the purpose of this description it is assumed a single feature set is extracted from each information source and a single classifier is trained and used on each feature set. This assumption is for clarity of description; however is possible to use: 1) multiple feature sets extracted from an information source, 2) combined information sources to generate a feature set, 3) multiple classifiers on a single feature set, or any combination of these with CobLE.

In the training phase, the feature sets extracted from each information source are fused together with the classification of each sample. When a sample is logged, if an information source was out of synchronization (*i.e.,* not providing data) the features of the feature set derived from the source are set to a unique "missing" value. This fusion of feature sets is performed for the sole purpose of introducing missing values for each set, and next the fused set is split back into the original feature sets. Each of the labeled feature sets with missing values is used to train the classifier which will operate on it and to approximate a confidence function for it. The process of training only one classifier is described, and the other classifiers will be trained in a similar manner.

The first step of training a classifier is to approximate a confidence function for the classifier over the feature space where it operates. The approximation process is a F-fold training approach to estimate the classifier's confidence for regions of the feature space represented in the training dataset (this process is similar to an inverted F-fold cross-validation with the training and testing sets in each validation iteration reversed). That is, the training set is divided into F groups; using one group the classifier is trained and the remaining data points (in the $F - 1$ groups) are classified, and for each correct/incorrect classification +1/-1 respectively is added to a data point (see Figure 7.1 - the training data points are divided into three groups denoted by red-circles, blue-diamonds, and green-squares; Figure 7.2 - two classifiers trained on the Green and Blue data points respectively attempt to classify the Red data points and add +1 / -1 scores to the data points). This process is repeated F-times (by retraining the classifier) using each group as a

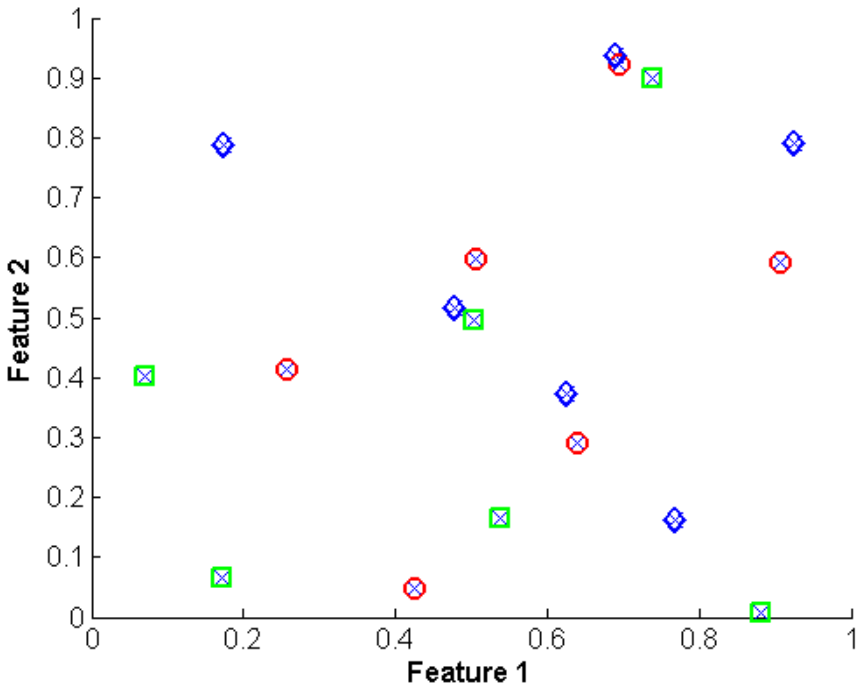


Figure 7.1: Steps in approximating a confidence function for a classifier - Step 1 : Training data points divided into 3 groups.

training set in turn. The final result is a training set with "heights", h ($-F < h < F, h \in \mathbb{I}$). To approximate a confidence function for any point in the feature space (assume the feature space is n-dimensional) handled by a classifier, we create a $(n+1)$-dimensional hyper-surface where the added dimension of the height of the surface represents the confidence for classifying that point. The implementation in this dissertation uses a Gaussian mixture model (GMM) to approximate the hyper-surface (where each Gaussian is n-dimensional). The heights from the F-fold training are used to fit the GMM for that underlying model (see Figure 7.3 - the confidence score of the data points are used to create a hyper-surface over the feature space, Figure 7.4 - an approximation of the hyper-surface is created using a 2-Gaussian mixture model). Section 7.6 discusses the effect of the number of training iterations (*i.e.,* F) and the number of Gaussian distributions in the hyper-surface GMM on classification accuracy. Next, the classifiers trained in the F-fold training are discarded, retaining only the confidence approximating hyper-surface;

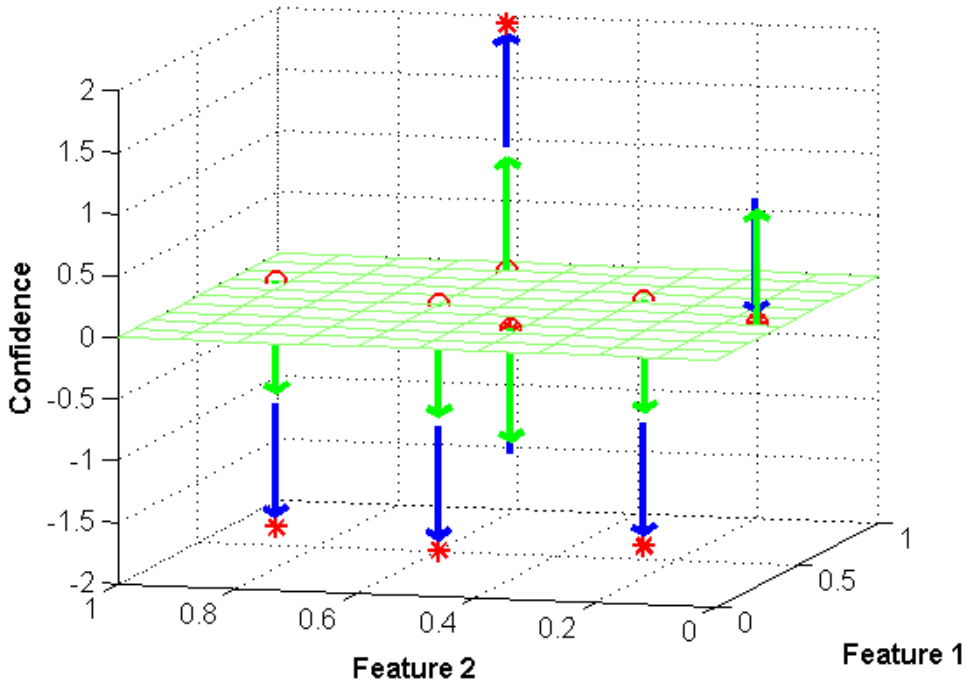


Figure 7.2: Steps in approximating a confidence function for a classifier - Step 2 : Data points in group Red rated for confidence by classifiers trained on the Blue and Green groups.

then, using the entire training dataset, the final classifier is trained.

To perform classification on a sample (d), each classifier in the ensemble is provided the subset of features it operates on. Any features extracted from a source out of synchronization in the current sample are filled with a value denoting a missing value. Each classifier (h_i) outputs a single classification (o_i) or a probabilistic classification ($P(k|d, h_i)$, the probability of d belonging to class k given the hypothesis i. The outputs from the classifiers are aggregated into a single classification (c) by,

$$c = \underset{k \in C}{\operatorname{argmax}} \frac{\sum_{i=0}^{N'} [w_{i,d} * \delta(o_{i,d}, k)]}{\sum_{i=0}^{N'} w_{i,d}}, \quad \text{where the Dirac delta function, } \delta(p, q) = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases} \tag{7.1}$$

$$c = \underset{k \in C}{\operatorname{argmax}} \frac{\sum_{i=0}^{N'} [w_{i,d} * P(k|d, h_i)]}{\sum_{i=0}^{N'} w_{i,d}}, \tag{7.2}$$

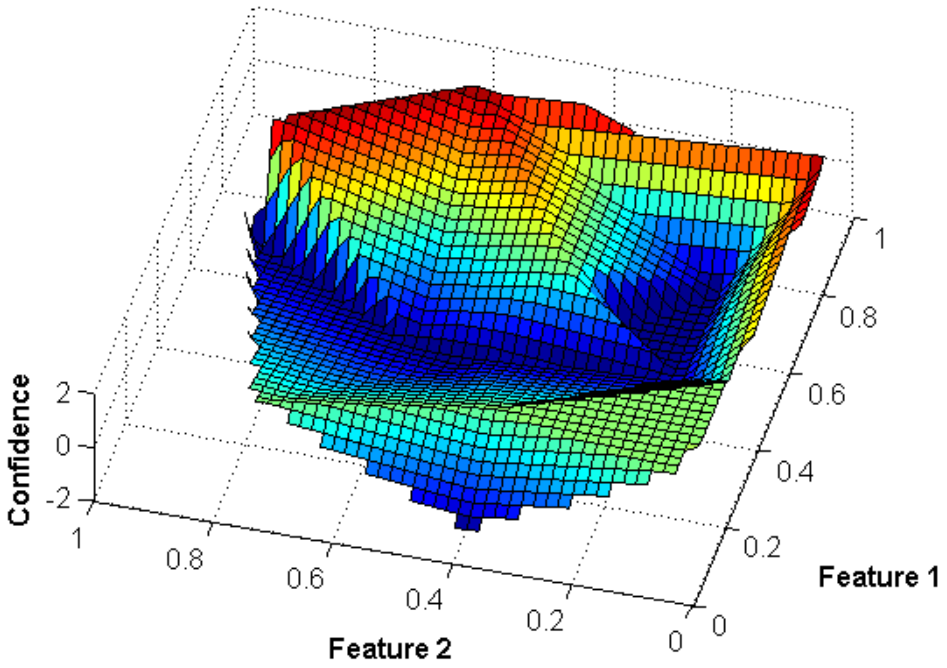


Figure 7.3: Steps in approximating a confidence function for a classifier - Step 3 : Confidence modeled with a 2D-hyper-surface (fitted through confidence ratings of all data points).

where Equation 7.1 is for classifiers that output a single class, and Equation 7.2 is for classifiers with probabilistic classification.

7.3 CobLE : Formal Description

This section describes CobLE in terms of a general framework for stacked generalizers presented by Wolpert in [72]. Assume the task of assigning a class c, from a set of classes C, to a vector x.

A learning set (Θ) of size M, inhabiting a R^{n+1} space and consisting of Θ_x input vectors (in R^n space) and Θ_y classifications (in one-dimensional space consisting of class labels from C),

$$\Theta = \left\{(x_a, y_a) : x_a \in R^n, y_a \in R^1, 1 \leq a \leq M\right\} \tag{7.3}$$

Classifiers operating in this R^{n+1} space are "level 0" classifiers. Let the set of classifiers operat-

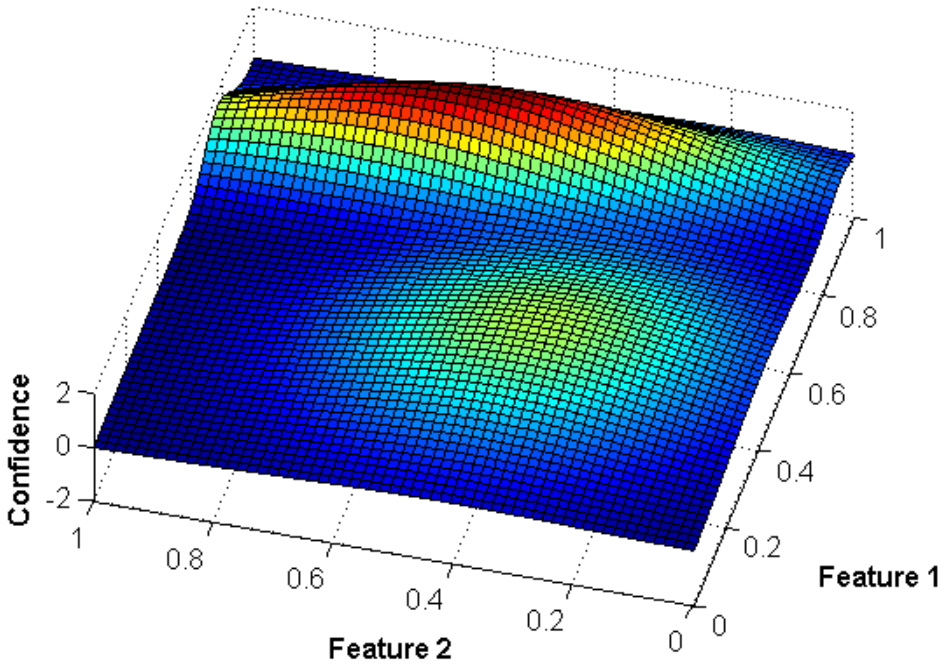


Figure 7.4: Steps in approximating a confidence function for a classifier - Step 4 : Confidence modeled with 2-Gaussian Mixture Model approximation of the hyper-surface.

ing in the level 0 space be $\left\{h^k\right\}$ (with $K = |\left\{h^k\right\}|$). Each h^k operates on $R^{n_{h^k}} \subseteq R^n$.

Θ is partitioned into r sets of $\{\Theta_{i1}, \Theta_{i2}\}$ where,

$$\Theta_{i1} \cap \Theta_{j1} = \emptyset \quad \forall\, i, j \in [1, r],\ i \neq j \quad \text{and} \tag{7.4}$$

$$\bigcup_{i=1}^{r} \Theta_{i1} = \Theta \tag{7.5}$$

For the i^{th} partition of Θ, each h^k generates a vector $r_{(k,i)}$ of length $|\Theta|$ (*i.e.,* M). The vector represents, if h^k when trained on Θ_{i1}, correctly classified each vector in Θ_{i2} Therefore the a^{th} element of $r_{(k,i)}$,

$$r_{(k,i),a} = \begin{cases} 0 & \text{if } \Theta_{x,a} \in \Theta_{i1,x} \\ \delta\left(h^k\left(\Theta_{i2,x} | \Theta_{F,i1}\right) - \Theta_{i2_y}\right) & \text{if } \Theta_{x,a} \notin \Theta_{i1,x} \end{cases} \tag{7.6}$$

where δ is the Dirac delta function. This can be concisely denoted as,

$$r_{(k,i),a} = [\![h^k\left(\Theta_{i2,x}|\Theta_{F,i1}\right) = \Theta_{i2_y}]\!] \tag{7.7}$$

where $[\![\pi]\!]$ for any predicate π is 1 iff π holds and 0 otherwise.

For each h^k, level 1 (*i.e.,* 2^{nd} stack) contains a "confidence" regressor S^k operating in a R^{n+1} space, $S^k : R^n \rightarrow R^1$. The learning set for a "level 1" regressor, S^k, is also in R^{n+1} space consisting of Θ_x input vectors (in R^n space) and r_k concatenated, where r_k is the normalized sum of the $r_{(k,i)}$ vectors,

$$r_k = \left(\sum_{i=0}^{F} r_{(k,i)}\right) / \left(F - 1\right) \tag{7.8}$$

Given a pair (x, y), each S^k estimates the likelihood, $S^k(x)$, of h^k correctly classifying x. *I.e.,*

$$S^k(x) = P\left[h^k(x|\Theta) \equiv y\right] \tag{7.9}$$

The final stack, *i.e.,* level 2, has a single aggregator, A, of level 0 outputs weighted by level 1 outputs ($A : R^{(2\times P)} \rightarrow R^1$).

$$A(x) \equiv \underset{c \in C}{\operatorname{argmax}} \left[\sum_{k=1}^{K} S^k(x|\Theta) \cdot [\![h^k(x|\Theta) = c]\!]\right] \tag{7.10}$$

To summarize, the three levels of the CobLE stacked generalizer are: level 1) base-classifiers, level 2) a confidence surface per classifier in level 1, and level 3) the weighted vote aggregation function.

7.4 Generalized Analysis of CobLE

As stated in the previous section, Θ with M training instances, is of the form $(x_a, y_a), a \in [1, M]$, where y is a scalar value and $y \in \{-1, 1\}$ can be assumed without any loss of generality.

Additionally x is (or can be thought of as) the combination of the K different input feature spaces $x^k (k \in \{1, K\})$. While $\bigcup_{k=1}^{K} x^k = x$, it is not a required constraint that $\bigcap_{k=1}^{K} x^k = \emptyset$, *i.e.,* independence is not required.

Let the hypotheses being combined in CobLE (and acting on each x^k) be $h^k (k \in [1, K])$ and the confidence surface for each h^k be S^k (where $\oint S^k(x) \mathrm{d}x = 1$).

Then CobLE's output,

$$H(x) = \text{sign}\left(\frac{\sum_{k=1}^{K} S^k(x) \cdot h^k(x)}{K}\right) \tag{7.11}$$

$$= \text{sign}(g(x)) \quad \text{where } g(x) = \frac{\sum_{k=1}^{K} S^k(x) \cdot h^k(x)}{K} \tag{7.12}$$

where the feature-subspace superscripts of x have been omitted for ease of reading and can be implied from the superscripts of h or S.

Then the training error of CobLE can be expressed as,

$$E_t = \frac{1}{M} \sum_{a=0}^{M} [\![y_a \neq H(x_a)]\!] \tag{7.13}$$

From Equation 7.11,

if $H(x_a) \neq y_a$	then,	$y_a \cdot g(x_a) \leq 0$
	implying that,	$\exp(-y_a \cdot g(x_a)) \geq 1$
	therefore,	$e^{-y_a \cdot g(x_a)} \geq [\![y_a \neq H(x_a)]\!]$

Applying this result to Equation 7.13,

$$\frac{1}{M}\sum_{a=0}^{M}[\![y_a \neq H(x_a)]\!] \leq \frac{1}{M}\sum_{a=0}^{M} e^{-y_a \cdot g(x_a)} \tag{7.14}$$

$$= \frac{1}{M}\sum_{a=0}^{M} e^{-y_a \cdot \sum_{k=1}^{K} \frac{S^k(x_a)\cdot h^k(x_a)}{K}} \tag{7.15}$$

$$= \frac{1}{M}\sum_{a=0}^{M} \left[\prod_{k=1}^{K} e^{-y_a \cdot \frac{S^k(x_a)\cdot h^k(x_a)}{K}}\right] \tag{7.16}$$

$$= \frac{1}{M}\sum_{a=0}^{M} \left[\prod_{k=1}^{K} e^{-y_a \cdot S^k(x_a)\cdot h^k(x_a)}\right]^{\frac{1}{K}} \tag{7.17}$$

Therefore the training error is bounded by how well $S^k(x)$ estimates $P(h^k(x_a) = y_a)$.

When the training set is $\{(x_a, y_a)\}_{a=1}^{M}$, the confidence surface of a hypothesis is estimated using the derived dataset $\left\{(x_a, y_a = h^k(x_a|\{x_b\})\right\}$ where $a, b \in [1, M]$. There are two cases based on the selection of a and b,

1. when $a = b$ is allowed (as with AdaBoost and RealBoost [23, 60]) it is equivalent to using the training dataset to perform testing, where the estimated error is:

$$\widehat{E}\left[(y - h^k(x))^2\right] = \frac{1}{M}\sum_{a=1}^{M}(y_a - h^k(x_a))^2 \tag{7.18}$$

2. when $a \neq b$ is enforced (as with CobLE) it is equivalent to using an isolated dataset for estimating the performance of a learner. In this case the error (estimated over an isolated sample set of the same size M) is:

$$E\left[(y - h^k(x))^2\right] = \frac{1}{M}\sum_{a=1}^{M}((y_a + \delta_{y,a}) - h^k(x_a + \delta_{x,a}))^2 \tag{7.19}$$

where $\delta_{x,a}$ and $\delta_{y,a}$ are random offset terms indicating the difference between the training set's a^{th} element and testing set's a^{th} element.

Since $\delta_{x,a}$ and $\delta_{y,a}$ have positive variances,

$$(y_i - h^k(x_i))^2 \leq ((y_i + \delta_{y,i}) - h^k(x_i + \delta_{x,i}))^2 \tag{7.20}$$

$$\frac{1}{M}\sum_{i=1}^{M}(y_i - h^k(x_i))^2 \leq \frac{1}{M}\sum_{i=1}^{M}((y_i + \delta_{y,i}) - h^k(x_i + \delta_{x,i}))^2 \tag{7.21}$$

$$\tag{7.22}$$

Therefore the above two cases (in Equation 7.18 and Equation 7.19) can be compared as follows,

$$\widehat{E}\left[(y - h^k(x))^2\right] \leq E\left[(y - h^k(x))^2\right] \tag{7.23}$$

By using the training dataset to estimate the performance of $h^k(x)$ the variability is underestimated (previous work also shows (1) is a worse estimate of actual performance than (2) [36, 67]). **Therefore CobLE's approach of using multiple folds to estimate confidence surfaces provides a more accurate estimate of $P(y = h^k(x))$.**

The accuracy of an estimated probability distribution increases as the number of mutually independent observations drawn from the same underlying distribution increase [38]. ***I.e.,***

$$\begin{aligned} E\left[P(h^k(x) = y) - S^k\left(x|\left\{x_a, y_a - h^k(x_a)\right\}_{a=1}^{M_1}\right)\right] > \\ E\left[P(h^k(x) = y) - S^k\left(x|\left\{x_a, y_a - h^k(x_a)\right\}_{a=1}^{M_2}\right)\right] \\ \text{where } M_1 < M_2, \text{ and } x_a \sim \text{PDF}(x) \end{aligned} \tag{7.24}$$

CobLE's approach of using F folds to estimate the performance of h^k at each training data point results in $F - 1$ mutually independent [1] observations of the distribution $P(h^k(x) = y)$ which is, on average, $F - 2$ more observations than with traditional approaches [23, 60]. **Therefore CobLE better estimates $P(y = h^k(x))$ using S^k than previous approaches.**

[1] For each x_i, $F-1$ independent observations of $h^k(x_i|X_{T,f}) = y$ are generated by ensuring $X_{T,f_1} \cap X_{T,f_2} = \emptyset$ for all $f_1 \neq f_2$ where $f_1, f_2 \in [1, F-1]$.

The next section presents experimental results on performance comparisons between CobLE- and other ensemble learning and sensor fusion approaches.

7.5 Performance Comparison Experiments and Results

The objective of the experimental setup is to empirically demonstrate and validate CobLE's performance in comparison to:

1. Monolithic learners operating on a single feature set (*i.e.,* a fused feature-set) created by feature-level fusion.
2. Classifier-fusion (*i.e.,* decision-fusion) approaches for combining classifiers trained on individual feature-sets.
3. Ensemble-approaches for combining learners operating on a single fused feature-set (combined using straightforward/weighted voting approaches).

See section 7.5.2 for a full list of learners used to compare against CobLE.

Using the comparative experimental results this section shows:

1. CobLE is a viable data fusion algorithm for systems with synchronized data-sources - it performs on-par or better than other approaches in situations where all data-sources are providing data (*i.e.,* all feature-sets carry valid values).
2. CobLE is an ideal data fusion algorithm for real-world systems with asynchronous data-sources - it outperforms other approaches in situations where only a few of the data-sources are providing data (*i.e.,* only some of the feature-sets carry valid values).
3. CobLE retains a high-level of performance even when the asynchronicity of data-sources increases - its degradation in performance is slight, whereas the degradation in performance of other approaches is much larger, as the number of data-sources providing valid data decreases.

7.5.1 Experimental Procedure

Four databases from diverse application areas, whose details are provided in Subsection 7.5.3, were used. For each of the databases, the following experimental procedure was followed.

Step 1 : One database naturally contained different feature sets, *i.e.,* real-world data from a data-fusion problem. The other three databases were randomly partitioned into disjoint subsets of features. Each partition was used to simulate features extracted from data generated by a different data-source (*i.e.,* a data fusion setting).

Step 2 : Neural networks (NNs) with a single hidden layer were chosen as the base-learners for the experiments due to their ability to simulate multiple hypotheses based on different initialization values. Using NNs as base classifiers, extensive n-fold cross-validation tests were conducted to determine optimum hidden layer sizes for each feature-set and for the fused feature-sets. Hidden layer sizes were varied from 1 perceptron to 10 perceptrons in increments of 1 and from 10 perceptrons to 100 perceptrons in increments of 5. Optimum hidden layer sizes were selected, based on averaged classification accuracy and lowest variance over 100 training and testing iterations. The optimal parameters are presented in Table 7.1. NNs with optimal layouts for each feature-set were used as the base-classifiers for CobLE and the other classifier-fusion approaches. NNs with the optimal layout for the fused feature-sets were used as the monolithic classifier (*i.e.,* a single model operating on the fused feature set) and the base-classifier for ensemble approaches. All of the neural networks trained in this step were discarded once the optimal values were found.

Step 3 : To simulate conditions where the data sources are asynchronous, new datasets are generated. The i^{th} generated dataset would have all possible combinations of i feature sets with valid values (and the other feature sets with missing values). In each successive dataset a sample is replaced with multiple samples where each has a feature-set with values set to missing (*e.g.,* for a dataset with 3 feature-sets, in the first generated dataset each sample would be replaced with 3 samples with the $1^{st}, 2^{nd},$ and 3^{rd} feature-sets set to miss-

ing respectively). After generating a dataset, duplicate samples are deleted. Therefore, if the original dataset has m samples and n feature-sets, the first new dataset has $m \times {}^{n}C_{1}$ samples, the second new dataset has $m \times {}^{n}C_{2}$, etc.

Step 4 : For each dataset, the samples are randomly divided up into 5 folds so that the class distribution in each fold remains approximately the same as the dataset's. In 5 iterations, each iteration using a different fold as the training set, all of the classifiers (CobLE, Monolithic AdaBoost, etc.) are trained and evaluated on the other 4 folds of data. The performance results for each classifier from the 5 iterations are averaged.

Step 5 : The procedure of Step 4 is repeated ten times to remove the bias of data selected into each of the folds.

7.5.2 Learners used for comparison

The following list describes the learners used to provide a comparison for CobLE's performance.

1. **Classifier fusion with simple voting scheme :** For each feature set a separate classifier (NNs in these experiments) is trained independently. At testing-time, each classifier operates only on the feature set it is trained on to perform classification. The classifications (from all of the classifiers) are aggregated (each vote carries equal weight), and the final classification is the class with the highest vote count [70]. This is the approach used for combining the behavior aspect models in Chapter 6.
2. **Classifier fusion with weighted voting scheme :** A set of classifiers are trained in the same manner as above. The fusion of votes are weighted by the posterior probabilities output by the classifiers [49, 56].
3. **Monolithic classifier :** The feature sets are first fused to create a single dataset (*i.e.,* feature-level data-fusion) and then a single classifier (in these experiments a NN) is trained using the entire fused dataset [65, 68]. Classification is the class output by the single clas-

sifier when provided the fused instance data (*i.e.,* input features).

4. **AdaBoosted monolithic classifiers :** AdaBoost [23] is used to train an ensemble of monolithic classifiers on the fused feature-set. AdaBoost begins by uniformly weighting all samples in the training dataset, where the weights indicate the importance of a sample. Then AdaBoost generates a new classifier based on the weighted samples and uses it to classify the samples in the training dataset. The weights of each incorrectly classified sample are increased, and the weights of each correctly classified example are decreased. AdaBoost repeats the generation of a new classifier and the readjusting of sample weights procedure until it has created a predetermined maximum number of classifiers or the training error is below a certain threshold.

5. **Single feature set classifiers :** This is n different classifiers, each operating on just a single feature set and completely ignoring the other datasets during training and testing. For brevity the results of the classifiers are averaged and presented as a baseline result.

7.5.3 Databases

The databases for these experiments were obtained from the UCI repository [3]. The details of each dataset are provided in Table 7.1. The second row gives a brief description of the dataset and the classification task. The next row describes the dataset size and the class distributions. The final row provides information on the available features and the compositions of the individual feature sets.

Table 7.2 describes the optimal hidden layer size for the neural networks used in the experiments for each feature set and for the fused datasets.

7.5.4 Results and Discussion

The results from these experiments are shown in Figures 7.5 - 7.8.

Table 7.1: Details of datasets used in experiments

Dataset	**Optical Recognition of Handwritten Digits (OCR) Dataset**	**Wine Dataset**	**Vehicle Silhouettes Dataset**	**Libras Dataset**
Description	Feature sets are extracted by applying different feature extraction methods (*e.g.,* Fourier analysis) on digit image.	Classify the type of wine based on features such as alcohol percentage, ash content, magnesium content, *etc.*	Identify vehicles using scale independent features extracted using moments, circularity, compactness, *etc.* [66]	Data on hand movements in LIBRAS. Features per sample are based on hand locations on 45 frames of a video.
Dataset Details				
Classes	10	3	4	15
Instances	2000	176	846	360
Instances per class	200	$1:59$ $2:71$ $3:48$	$1-3:190$ $4:176$	24
Feature Sets				
Total Features	649	13	18	90
Number of Sets	6	4	4	5
Set Sizes	$1:216$ $2:\ 76$ $3:\ 64$ $4:240$ $5:\ 47$ $6:\ 6$	$1-3:3$ $4:4$	$1-2:5$ $3-4:4$	1 - 5 : 5

Table 7.2: Details of optimal hidden layer size for the neural networks used in the experiments for each feature set and for the fused datasets.

Dataset	**Optical Recognition of Handwritten Digits Dataset**	**Wine Dataset**	**Vehicle Silhouettes Dataset**	**Libras Dataset**
Optimal Hidden Layer Size				
Feature Set 1	45	5	40	90
Feature Set 2	35	5	35	95
Feature Set 3	45	5	30	90
Feature Set 4	45	5	10	90
Feature Set 5	15	-	-	85
Feature Set 6	45	-	-	-
Fused Feature Set	95	30	70	85

Each figure shows the variation of the average classification accuracy[2]of each approach, varying as the percentage of features providing data increases to 100%. Results on all 4 datasets show that the CobLE approach and the approach of applying AdaBoost on the fused feature set perform approximately equally well but clearly better than the other approaches when all data sources are active (*i.e.,* all feature sets have valid values).

Another observation that is clear across the results on all four datasets is, as the percentage of active data sources decreases, the performance of the AdaBoost approach reduces drastically. The performance of the weighed voting approach (and to a lesser extent the performance of the simple voting approach) remains somewhat stable reducing less dramatically than the performance of the AdaBoost approach. In contrast, the performance of the CobLE approach is the most stable as the number of feature sets with missing values increases.

By performing on-par with AdaBoost when all data sources are active, and having more stable

[2] Accuracy $= \frac{\text{(number of true-negatives and true-positives)}}{\text{(total number of samples)}}$

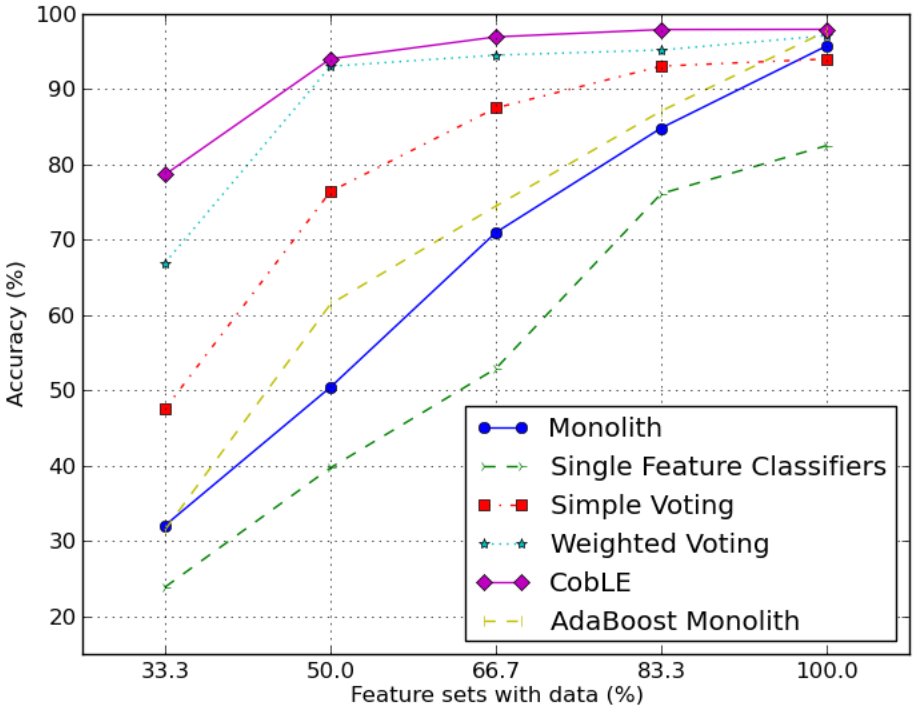


Figure 7.5: Comparative results on the OCR dataset.

performance than any other approach as the number of active data sources reduce, CobLE is a no-loss classifier approach for situations where 1) the data sources are asynchronous, 2) the data sources may fail, and even 3) when the data sources are synchronous and reasonably stable.

7.6 CobLE Model Tuning

The Confidence-based Learning Ensembles framework in principle has only one parameter to change its behavior, *i.e.,* the parameter defining the number of folds the training data is split into, to estimate the confidence surface heights at each training data point. The implementation of CobLE discussed in this dissertation using Gaussian mixture models to approximate confidence surfaces introduces a second parameter to the model, *i.e.,* the number of Gaussian distributions per mixture model. The objective of the experiments presented in this section was to observe the effect that the values of these parameters had on the performance of CobLE.

The OCR dataset was again used keeping the asynchronous dataset creation methods and

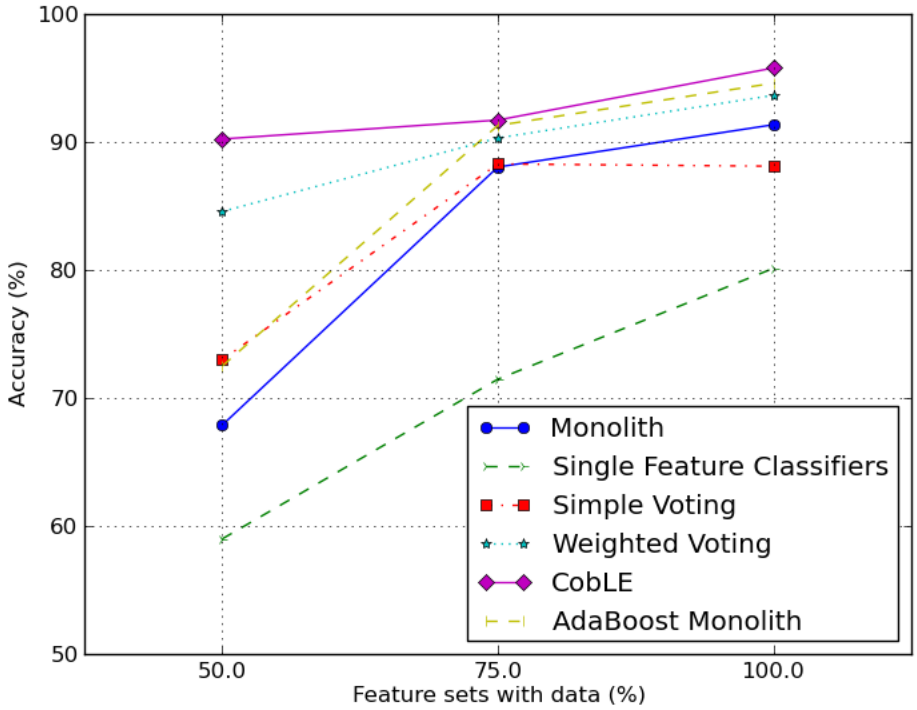


Figure 7.6: Comparative results on the Wine dataset.

training-testing procedures unchanged (as outlined in experiments in Section 7.5). In the first part of the experiment, the number of Gaussian distributions are varied from 10 to 100 in steps of 10, with 3-fold sampling. In the second part of the experiment, the number of folds used to estimate the confidence surface is varied from 3 to 20, with the samples used to train a 10 Gaussian GMM.

CobLE's accuracy as the number of Gaussian distributions used to estimate the confidence surface are increased, initially increases, albeit minutely, but then reduces as the number is increased further (see Figure 7.9). The pattern holds even though the number of active data sources are reduced. The initial increase in performance as the number of Gaussians used increase is due the confidence surface being better modeled. As the number of Gaussians are increased further, the confidence surface over-fits towards having a Gaussian mean at each training data point, thus giving inaccurate confidence estimates for any points which are not immediately adjacent to the training data points, leading to a reduced performance.

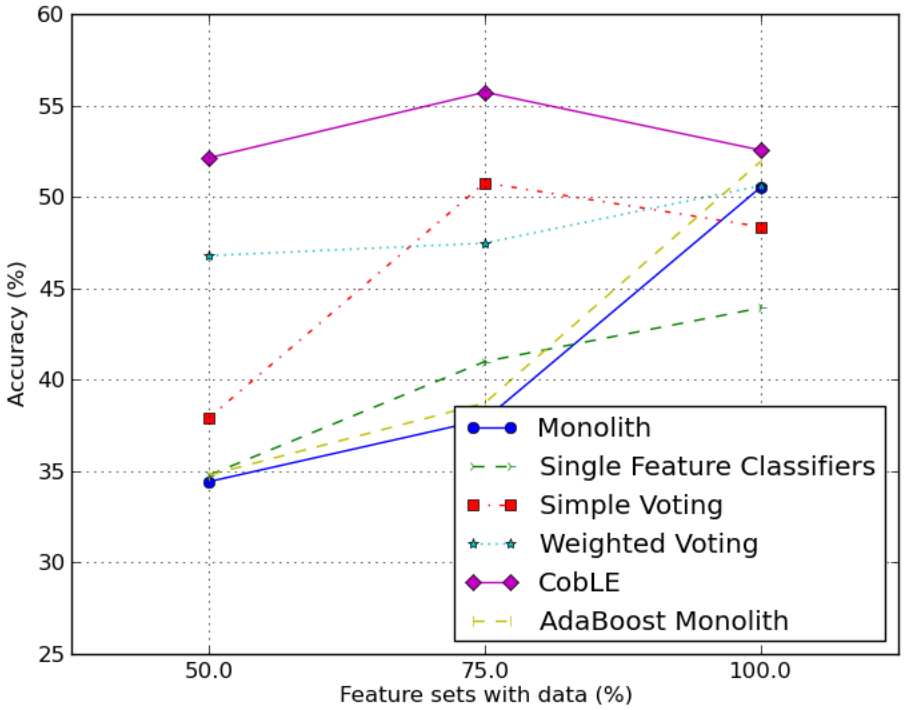


Figure 7.7: Comparative results on Vehicle dataset.

As the number of training iterations performed to estimate the confidence surface are increased, there is an initial increase in performance (see Figure 7.10). This initial increase can be attributed to the confidence surface estimation getting more fine-grained estimates on how the base-classifier performs at each training data-point (since, if k iterations are performed, each data point receives $k - 1$ correct/incorrect points towards the surface estimation). When the number of iterations are increased further, CobLE's classification accuracy begins to decline (see Figure 7.10). As the number of iterations are increased further, the amount of training data used to train a ("throw-away") classifier for each iteration of the surface estimation is reduced, creating less reliable approximations of how the base classifier will perform when it is trained with all of the training data. This results in a less accurate confidence surface estimate.

Selecting the number of Gaussians to represent the confidence surface and the number of training iterations used to estimate confidence surface requires some empirical testing, but the size of the training dataset can be used as a rough pointer for picking appropriate values. A further

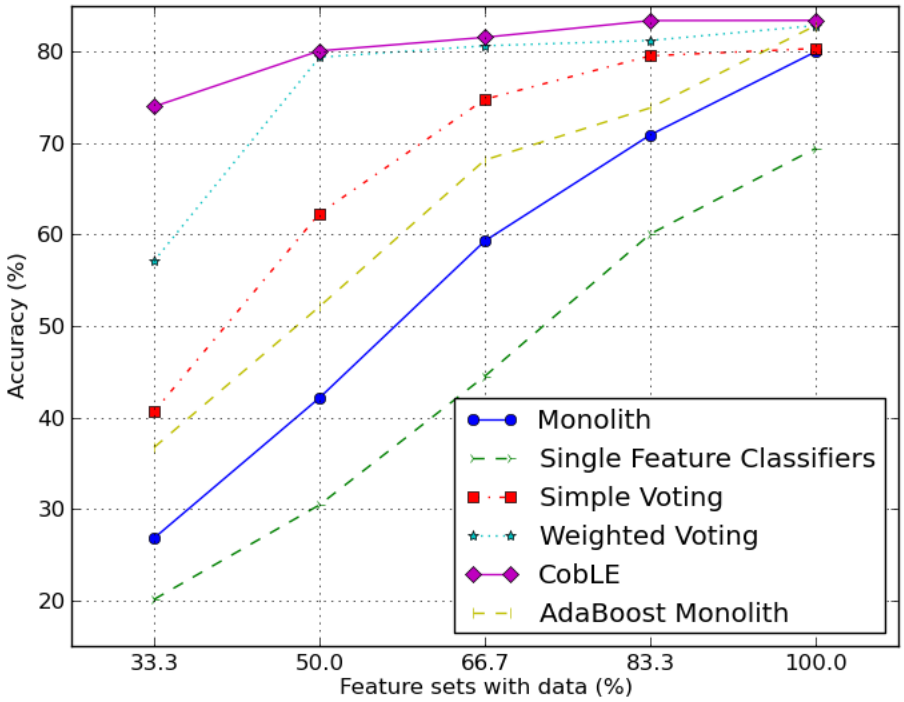


Figure 7.8: Comparative results on Libras dataset.

consideration to keep in mind when picking these values are the additional training compute and time costs they incur. While the additional training costs of the CobLE approach has not been experimentally evaluated, it has been observed that there is approximately linear increase in training time as, either, the number of Gaussians or training iterations are increased.

This chapter presented CobLE, an approach for creating an ensemble of classifiers based on the hypothesis that different classifiers in an ensemble will perform with varying accuracy in different regions of the input feature space. A formal analysis of the CobLE approach was presented. Extensive experimental evaluations show that CobLE is able to effectively create an ensemble of learners using different and asynchronous data sources. Experimental results show that CobLE is also effective as a traditional ensemble learning approach and a data-fusion approach for synchronous data sources.

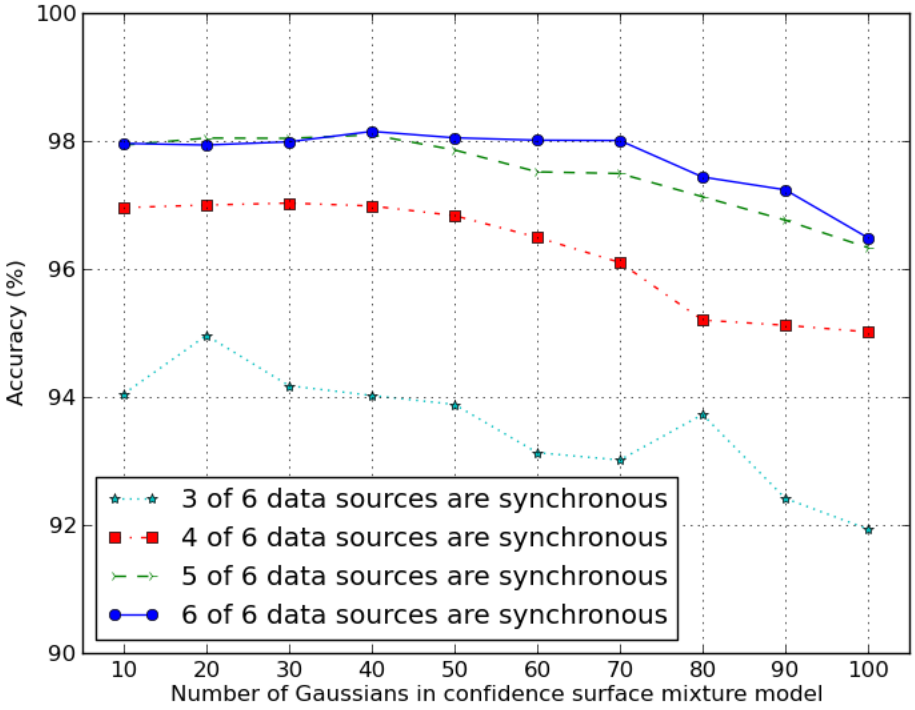


Figure 7.9: Accuracy variation vs. number of Guassians used in mixture models.

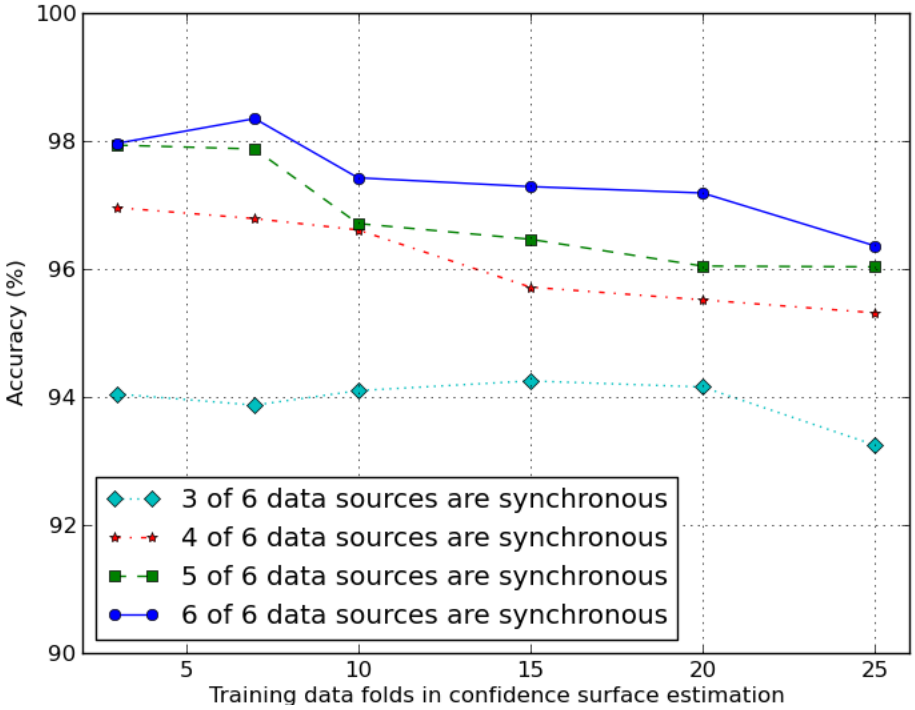


Figure 7.10: Accuracy variation vs. number of folds used in confidence estimation.

Chapter 8

McFAD : Mobile Computing Framework for Anomaly Detection in User Behavior

In this chapter, the Mobile Computing Framework for Anomaly Detection in User Behavior (McFAD) is described. McFAD is a framework designed to fuse models capturing different aspects of a user's behavior. The framework uses the Confidence-based Learning Ensembles (CobLE) learning approach to fuse the output behavior aspect models. The goal of McFAD is to perform anomaly detection more effectively and with better coverage throughout the user's day. This chapter describes the framework and how it was implemented for the Android platform. McFAD is a general framework providing anomaly detection services to the mobile phone's operating system as well as to applications running on the device.

8.1 McFAD Architecture

The McFAD framework sits atop the OS on a mobile device (see Figure 8.1) and obtains sensor and usage data from the operating system. The output from the framework is an anomaly signal indicating whether or not an anomaly has been detected, and a confidence score. This feedback is also provided to the authentication service on the mobile device, signaling to the authentication

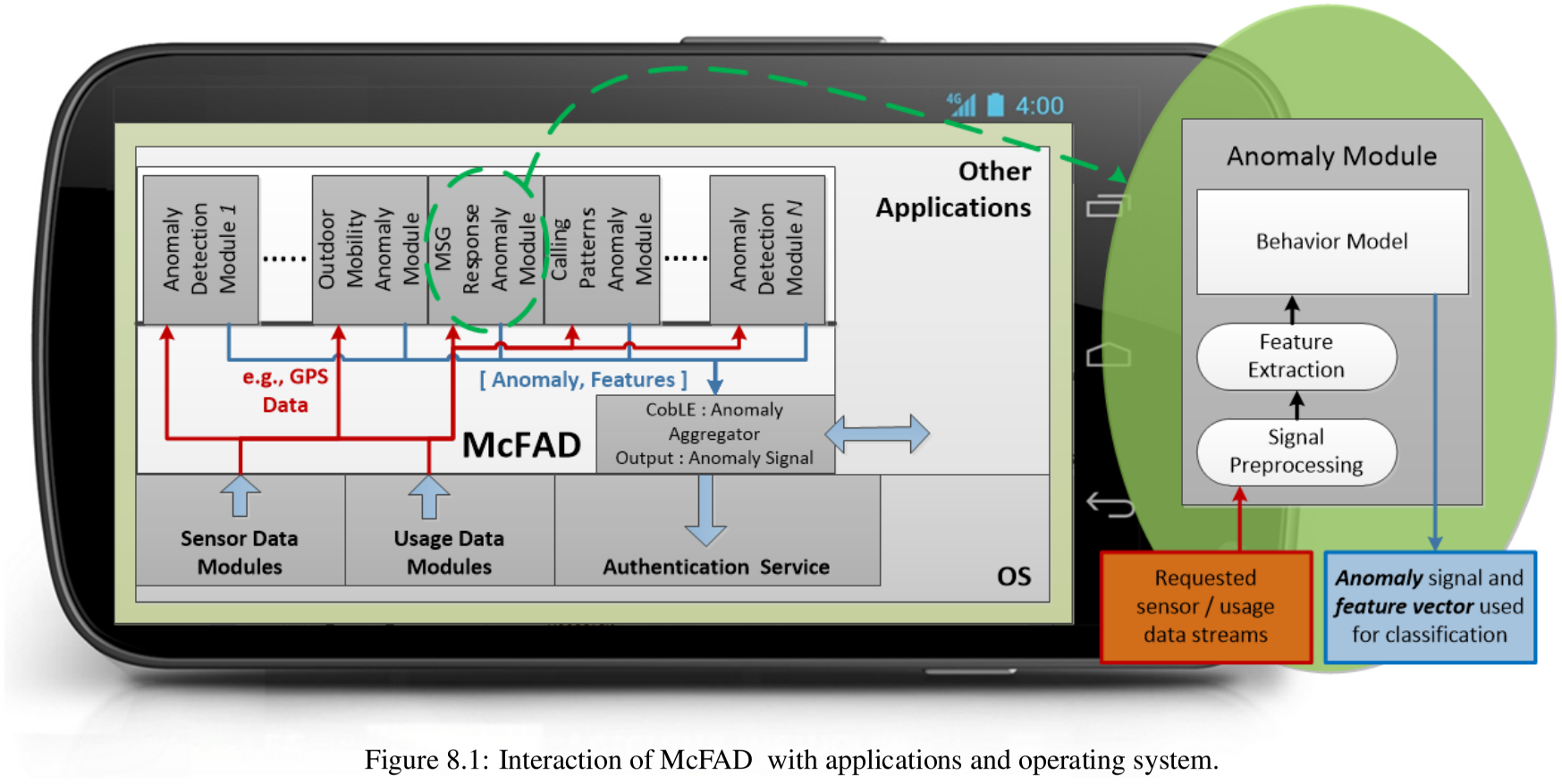


Figure 8.1: Interaction of McFAD with applications and operating system.

service to revert to traditional authentication approaches (e.g., password-based) when anomalous behavior with at least an adjustable level of confidence is detected. McFAD can also be configured by applications to monitor particular sensor stream combinations and to notify the applications when anomalies are detected.

The framework in itself does not provide any anomaly detection capabilities, rather it provides an interface for plugging-in different anomaly detection modules. Each module models an aspect of the user's behavior (see Section 2.1). A module can request the soft-sensor, hard-sensor, and/or radio channel information streams it requires from the framework. As output each module signals the framework whether or not the current behavior is anomalous along with the feature vector used to arrive at that conclusion. A module may also abstain from presenting a classification (*e.g.,* if the model deems it does not have sufficient data). The framework uses the feature vector provided by a module with the module's confidence surface, estimated off-line during training using Confidence-based Learning Ensembles (CobLE), to weight the anomaly signal provided by the module. Finally, the framework combines the weighted anomaly signals.

An anomaly detection module is trained and primed for the use with the McFAD framework off-line, where priming involves the estimation of the module's confidence surface to use with CobLE (see Section 7.2).

In the following two sections, the interfaces McFAD opens up for interactions are presented. The first interface is for creating anomaly detection modules that integrate with McFAD. The second interface is for applications which utilize McFAD's anomaly detection capabilities.

8.2 McFAD Anomaly Detection Framework

The anomaly detection framework is implemented as an Android Service which runs in the background and does not provide a user interface. Once it is installed, it is setup to automatically start when the device is booted up and to stay operational until the device is shutdown. The framework comes with a companion controller application which the user can use to force the shutdown or

restarting of the framework.

Though there could be modules loaded into the framework which use network connectivity information as information sources, McFAD is designed to operate completely locally on the mobile device. This design decision was made to keep the anomaly detection functionality available even if the device is out of network range (or forced to loose connectivity, say, if the device were stolen). The only instance where data is taken off the mobile device is to retrain modules, which happens periodically when the device is docked with the user's desktop computer.

As the user's behavior evolves over time, the anomaly detection modules also need to be updated (retrained) with new user data. To support retraining of modules, the framework also logs any data samples from the information sources requested by the anomaly detection modules. These logged data can be offloaded to the user's desktop computer when the mobile device is docked with the computer. Again, this is the only instance where user data is taken off the mobile device (*i.e.,* to retrain the anomaly detection modules). Note that the retraining strategy for each module is dependent on the models used by the module, *i.e.,* the module could be capable of updating or may require retraining from scratch.

8.3 Anomaly Detection Module Interface

The McFAD framework provides an interface for creating new anomaly detection modules which can be integrated into the framework. The interface is comprised of three components:

1. ***IAnomalyDetectionModule* interface** - a Java class interface that the anomaly detection module implements so that McFAD can interact with the module.
2. **Manifest file format** - a file format for a manifest used by the McFAD framework to load the anomaly detection module. The manifest also specifies the information streams the module requires.
3. **Confidence surface file format** - a file format to describe the confidence surface of this

anomaly detection module. This file is generated off-line at training time by a tool provided with the framework.

An anomaly detection module is provided to the McFAD framework as a archive file containing the class file implementing the *IAnomalyDetectionModule* interface, the module's manifest file, and the module's confidence surface description file. The archive can also hold extra classes used by the module or additional data files used by the module. Any additional data files and/or classes are loaded by the McFAD framework and made available to the module at run-time.

8.3.1 *IAnomalyDetectionModule* Interface

The *IAnomalyDetectionModule* interface has three method definitions which an anomaly detection module must implement so it can be used by McFAD. The method definitions are shown in Listing 8.1.

```
public void init( String dataPath );

public void receiveNewData(
                ArrayList< DataSample > newData );

public ArrayList< Double > detectAnomaly(
                Long timestamp );
```

Listing 8.1: *IAnomalyDetectionModule* interface definition.

The *init (...)* method is called by the framework when it loads the modules. The passed parameter is the path to where the data files (if any) that were provided in the module's archive have been extracted to.

McFAD periodically calls the *receiveNewData (...)* method to provide new data from the information sources to the module. The new data is timestamped samples from the information sources the module has requested data (specified in the manifest file). This method will only be called when new data samples are available, and the parameter carries only the the samples that

have been generated by the information sources since the last invocation of this method. The frequency of updating a module is defined by the McFAD framework.

McFAD periodically invokes the *detectAnomaly (...)* method to get the module's opinion of whether the user's present behavior is normal or anomalous. The method is called with a single parameter indicating the time in milliseconds since epoch for UTC. McFAD expects the method to return an array of floating point numbers. The first value in this array is an indication of whether the module believes the current behavior is routine or anomalous (it may also indicate that the module has abstained from making a decision). The other values in the array represent the feature vector used by the module to arrive at its decision. This feature vector is used by McFAD along with the module's confidence surface to gage the module decision's reliability (*i.e.,* in combining the output of multiple modules using CobLE).

8.3.2 Manifest File

The manifest is used by the framework to load the anomaly detection module. The manifest defines the main class of the anomaly detection module, *i.e.,* the class that implements the *IAnomalyDetectionModule* interface. The manifest also contains a list of classes and files used by the anomaly detection module, so that the framework can extract and allow the module access to those files. Finally, the manifest also holds a list of information sources from which the module requires data.

McFAD maintains and manages the information sources required by anomaly detection modules. The framework periodically updates each anomaly detection module with any new data from the information source specified in the module's manifest, if new data is available.

8.3.3 Confidence Surface File

McFAD uses the Confidence-based Learning Ensembles (CobLE) approach to combine the output of multiple anomaly detection modules. CobLE operates using a confidence surface

function estimated at training time (*i.e.,* training time of the module) over the feature space over which the module operates (see Section 7.2 for a full description).

The McFAD framework provides an off-line tool to be used during the training phase of a module for estimating a confidence surface for the module. The tool models the confidence surface as a Gaussian mixture model over the module's feature space. The estimated confidence surface function is stored in a file and archived along with the module so it can be loaded by McFAD at run-time.

8.4 Client Application Interface

The McFAD framework has an open interface that applications on the mobile device can use to leverage its anomaly detection capabilities. The life-cycle of an application's interactions with the framework are quite straightforward:

1. Register with McFAD framework requesting anomaly detection notification.
2. The framework broadcasts notifications to all registered applications if an anomaly in the user's behavior is detected.
3. When the application is no longer interested in receiving anomaly notifications (or when the application is terminating), unregister from the framework.

The interface provided to applications to connect with the framework is a very straightforward two item interface.

1. ***McFADServiceConnection*** **class** - an Android class which handles all communications with McFAD.
2. ***IMcFADResponseListener*** **interface** - a Java interface which a class in the application can implement if the class wants to fully interact with the *McFADServiceConnection*.

8.4.1 *McFADServiceConnection* Class

This is an Android class handling all of the communications with the McFAD framework. It hides all of the details of the communication protocol used by the framework. An application can instantiate an instance of this class to establish communication with the framework. The class exposes three methods shown in Listing 8.2.

```
public void bind();
public void unbind();

public void addResponseListener(
                IMcFADResponseListener listener );
```

Listing 8.2: Methods exposed by the *McFADServiceConnection* class.

By invoking *bind ()* the application causes the *McFADServiceConnection* object to register the application for anomaly notifications with the McFAD framework. As the name implies, *unbind ()* unregisters the application from the McFAD framework, thus preventing the application from receiving any further anomaly notifications. Using the *addResponseListener (...)* method of the *McFADServiceConnection* object, the application can register a class (implementing the *IMcFADResponseListener* interface) to receive callbacks when the McFAD framework sends out notifications or when there are changes in the connection/registration status of the application with the framework.

8.4.2 *IMcFADResponseListener* Interface

Classes implementing this interface can be registered with a *McFADServiceConnection* object to receive callbacks when the McFAD framework sends out notifications or there are changes in the connection/registration status of the application with the framework (see interface Listing 8.3).

```
public void onRegistered();
public void onUnregistered();
public void onAnomalyRaised(    Date timestamp,
                                float severity,
                                float confidence );
```

Listing 8.3: Callbacks to be implemented by a class registering a *McFADServiceConnection* object. Defined in the *IMcFADResponseListener* interface.

The *onRegistered ()* callback is invoked when a registration request (for receiving anomaly notifications) with the McFAD framework has been completed. The *onUnregistered ()* callback is invoked when a unregister request from the framework has been completed. The *onAnomalyRaised(...)* callback is invoked when the *McFADServiceConnection* object has received an anomaly notification from the McFAD framework. Parameters of the callback are: the time at which the anomaly was detected, the severity of the anomaly, and the confidence with which the McFAD framework detected the anomaly.

Since the framework runs as a Service on top of the Android OS, the coupling mechanism as been intentionally left light. All data exchanged between the McFAD service and applications using the framework are via an Android Messenger, allowing message-based communication across processes.

This chapter presented the McFAD framework and the implementation of it on the Android platform. In the next chapter the McFAD framework's anomaly detection performance and coverage evaluations are discussed.

Chapter 9

Comparison of Approaches

The chapter presents a performance evaluation of Mobile Computing Framework for Anomaly Detection in User Behavior (McFAD) using the Confidence-based Learning Ensembles (CobLE) approach to combine the individual models, each capturing a single aspect of a user behavior. This anomaly detection performance evaluation is presented to contrast this modular approach with a monolithic learning approach. The performance evaluation also contrasts utilizing the same behavior aspect models with a straight-forward voting scheme to combine their output (as opposed to using CobLE).

First a description of the monolithic learner used for the comparison is presented.

9.1 Monolithic Learner

The most extreme version of a monolithic learner would be to present a learner with near-infinite modeling flexibility with the raw information sources, thus creating a completely unstructured foundation upon which the learner can begin to model the user's behavior. While this approach would lead to a very versatile learner capable of learning almost any routine, the training data requirements would be enormous. Therefore, for comparison against the McFAD a more structured monolithic learner is utilized.

The monolithic learner presented in this section is designed to capture:

- The mobility patterns of the user both indoors and outdoors.
- The messaging patterns of the user.
- The calling patterns of the user.

By modeling only the parts of the user's routine that McFAD captures in this evaluation, the monolithic learner provides a fair monolithic equivalent to the McFAD framework for performance and training data requirement comparison.

For the monolithic model a dynamic Bayesian network (DBN) [28, 39, 50] is used. The DBN was chosen for its versatility as a Bayesian network, and also for its ability to capture temporal relationships. A notable previous work using a (modified) DBN for modeling user behavior is the Opportunity Knocks project [54, 55].

DBNs build upon Bayesian networks, i.e., directed acyclic graphs where the nodes represent random variables and the edges represent the conditional dependencies between the random variables. As with a random variable in a Bayesian network, a random variable in a DBN may represent 1) an observable quantity, 2) a latent variable, 3) an unknown parameter, or 4) a hypothesis. The dynamism in DBNs as compared with Bayesian networks arise from their ability to model sequential data. A hidden Markov model (HMM) is a good example of a simple DBN. In general, the concept of DBNs provides a rich framework in which dynamic probabilistic models can be treated in a uniform way. A DBN can be thought of as being comprised of a series of identical Bayesian networks. The sequential dependencies, *i.e.,* dependencies between random variables in adjacent Bayesian networks in the series, are represented by edges between the consecutive Bayesian networks. Hence, a DBN is defined by two components: 1) A set of nodes that represent all random variables for a given point the in the sequence, and the edges between those nodes. 2) A set of edges that connect nodes in two consecutive Bayesian networks in the sequence.

A DBN's structure can be completely described by specifying all the nodes and edges that

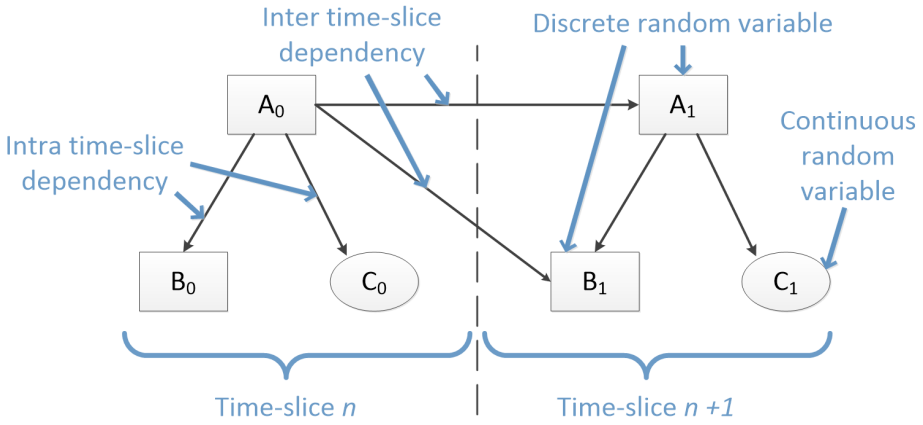


Figure 9.1: An example dynamic Bayesian network illustrating the components of a DBN model.

belong to two consecutive Bayesian networks (*i.e.,* time-slices) in the series[1](See Figure 9.1). That is, the nodes and edges of a one of the Bayesian networks describes both Bayesian networks as they are identical, but the second Bayesian network is required to describe the relationships between random variables as the series progresses. This set of nodes and edges can be thought of as repeating for as long a sequence of steps are required to be modeled. Nodes representing continuous random variables are represented as elliptical nodes and nodes representing discrete random variables are denoted using rectangular nodes.

Creating a DBN model to represent a phenomenon based on observed data (*i.e.,* training data) can be thought of as a two-stage problem, *i.e.,* 1) defining the structure of the DBN, and 2) determining parameter values of the model. The structure of a DBN can be learned from the training data or manually defined. Approaches for learning the structure of a DBN from training data [15, 25] greatly increase the training data requirements and also have high computational complexities. Therefore, for the monolithic model used in this evaluation, the structure was manually defined. Once the structure of the DBN is determined, estimating the parameter values of the model is done by finding the maximum likelihood values for the parameters based on the training dataset. Apart from a few simple DBN architectures, it is nearly impossible to explicitly

[1] Complex alternatives to DBNs with conditional probabilities dependent on random variables lying in the Bayesian network before the previous time-slice's Bayesian network are possible. Strictly speaking, such models are not considered DBNs.

calculate the maximum likelihood values of the DBN parameters. Therefore, DBNs are typically trained using an expectation maximization (EM) algorithm. Using multiple iterations of the EM algorithm, the maximum likelihood values for the DBN's parameters are estimated.

The most intuitive method to detect anomalies using DBN models would be to introduce an "anomaly" variable dependent on the "state" variable from the current time-slice and the "anomaly" variable from the previous time slice. The work in this dissertation assumes that capturing examples of anomalous behavior for an individual is a very expensive process and therefore assume that the training dataset contains only positive examples of normal behavior. With zero examples of anomalies in the training dataset, adding an "anomaly" variable and properly estimating its likelihoods becomes a convoluted task. This problem is addressed by estimating the likelihood of the entire DBN over a series of observations and comparing this estimate to a predetermined anomaly threshold, *i.e.,* the likelihood of observations being from routine behavior. This approach is similar to the approaches used with the behavior aspect models described in Chapters 3, 4, and 5.

The DBN based monolithic model used here was implemented using the Mocapy++ toolkit [51]. The Mocapy toolkit provides functionality for parameter learning and inference in DBNs using Markov-chain Monte Carlo methods [30]. By using Markov-chain Monte Carlo methods Mocapy is capable of handling complex DBNs, with large training datasets containing long observational sequences.

Before describing the full monolithic model, sections of the model that capture the user's mobility, messaging patterns and calling patterns are presented separately for the purpose of descriptive clarity.

9.1.1 Modeling Mobility in the Monolithic Learner

The user's location is modeled using the same sensor streams used for capturing the user's location in McFAD, namely the GPS, WiFi signatures, and cell-tower information. The obser-

vations from the GPS sensor are modeled as two separate continuous random variable nodes, each representing longitude and latitude respectively (see Figure 9.2, nodes long_i and lat_i). The two nodes model their random variable values as a Gaussian mixture model. The WiFi signatures are created from the WiFi scan data of the device. The procedure described in Section 5.2 is used to convert the scan data into a discrete label, modeled as a discrete random variable (wifi_sig_i nodes in Figure 9.2). The cellular tower information (the tower to which the phone is currently connected and the visible neighboring cell-towers and their signal strengths) is fed to a pre-processing step similar to the one described in 5.1 to obtain a discrete label for the user's location. These labels are modeled as a discrete random variable within the monolithic model (tower_sig_i nodes in Figure 9.2).

The four nodes, *i.e.,* longitude, latitude, WiFi signature, and cell-tower signature, representing observed values are considered to be dependent only on the user's state. The user's state, modeled as a discrete random variable node in the monolithic model, is an unobservable variable. The user's state is considered to be influenced by only on the user's previous state and the current time of day. The time of day is also modeled as a discrete random variable node in the monolithic model. Its value is influenced only by the time of day in the previous iteration of the model (*i.e.*, in the previous time-slice). The user's state and current time are depicted in Figure 9.2 as s_i and t_i nodes respectively.

9.1.2 Modeling Messaging Patterns in the Monolithic Learner

The messaging patterns of the user are modeled using the information obtained from the messaging logs on the device, yielding the times at which and number to/from which messages are sent/received are obtained. The sequences of sent and received messages from a single contact are preprocessed to form a sequence of four observations (at each point in the sequence). The four observations, per-contact, in a sequence are,

1. **recv** - whether a message has been received from that contact.

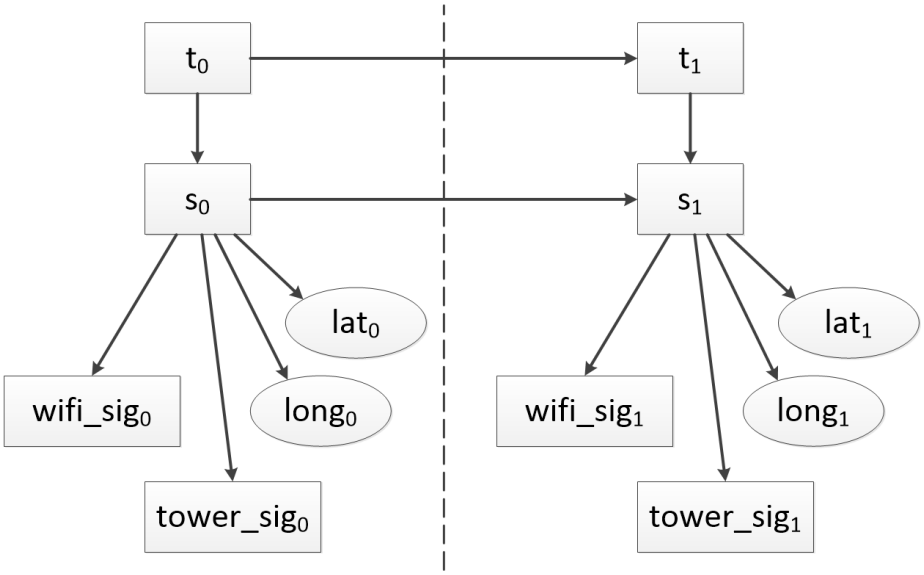


Figure 9.2: Part of monolithic model to capture the mobility patterns of the user.

2. **sent** - whether a message has been sent to that contact. A sent message also resets the *recv* observation, as the received message(s) has now been responded to.
3. **delay** - the value for this observation is set to *undefined* (represented as -1) unless trigged by *sent* while *recv* is also set. In which case, the delay is calculated as the time difference between when the user responded to this contact's message and when the contact last messaged the user.
4. **un_asw** - whether a message from this contact has gone unanswered. That is, a *recv* variable has been set for a extended period without a message from the user. The period before marking a message as unanswered was experimentally decided upon to be 1 hour. In the preprocessing, setting the *un_asw* observation clears the *recv* observation. The *un_ans* observation is cleared after a single time-step in the sequence.

Per-contact, *recv*, *sent*, *un_asw* are modeled as discrete random variables, while *Delay* is modeled as a continuous random variable (see Figure 9.3a). Mocapy places some restrictions on continuous random variable nodes, and in this case the restriction of interest is that each node representing a continuous random variable may be influenced and influence a total of one node.

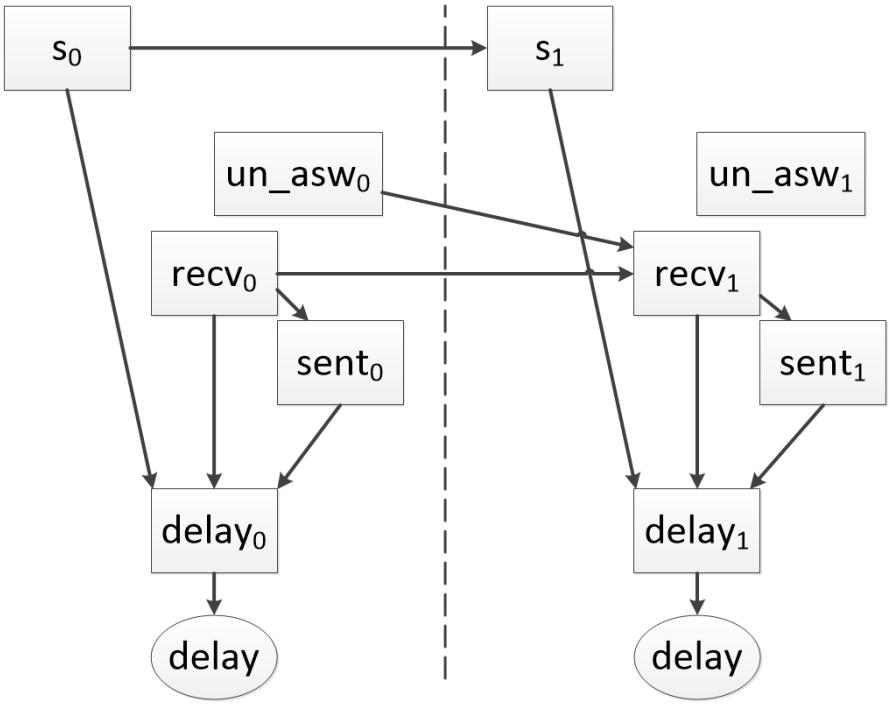
In the monolithic model, the delay is assumed to depend on the user's state, and whether or not a message has been received. To work around Mocapy's restriction in this case, a "dummy" discrete random variable node was introduced (also termed *delay*) which represents the presence or absence of a delay, and this node becomes the only influencer node on the continuous random variable node *delay* (see Figure 9.3a). Using these observation variables and internal state variables, the part section of the monolithic model used to capture the user's message response behavior towards a single contact is depicted in Figure 9.3a).

The partial single-contact model (see Figure 9.3a) is extended to multiple contacts by replicating all of the nodes, except for the common user's state discrete random variable (node S_i). The user's most important (in terms of number of interactions) n contacts are each assigned a partial per-contact model, and all remaining contacts are clumped together to form a single contact (represented by the final partial per-contact model). The number of contacts for whom individual partial per-contact models are assigned, *i.e.,* n, was experimentally evaluated and it was seen that $n = 15$ provided optimum results for this dataset. The section of the monolithic model capturing the user's full message response behavior is depicted in Figure 9.3b).

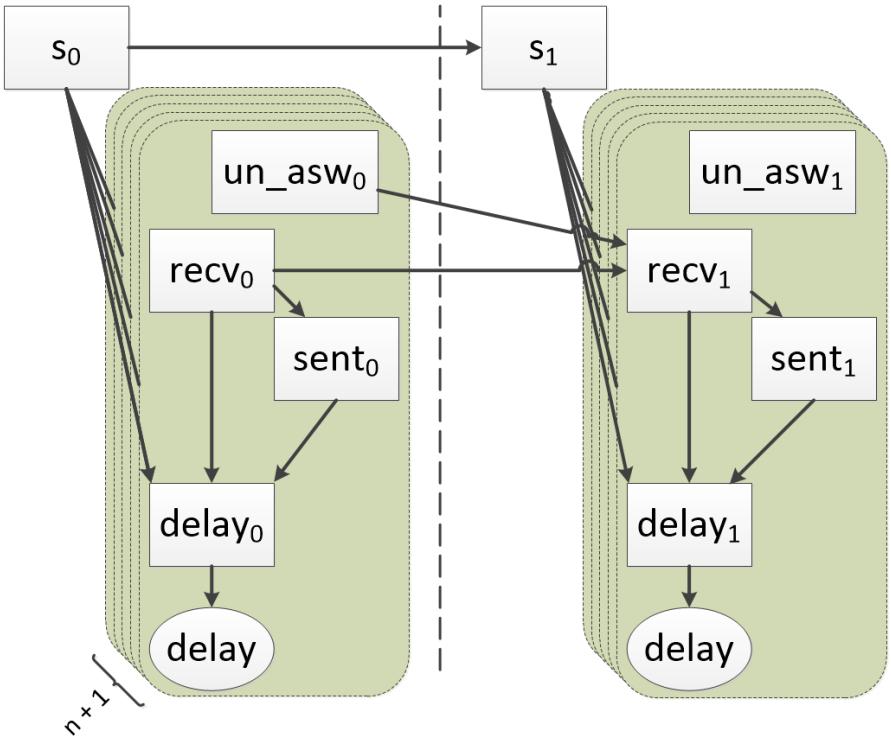
9.1.3 Modeling Calling Patterns in the Monolithic Learner

The calling patterns of the user are modeled using the information obtained from the call logs on the device. A list of tuples are extracted from the call log for each entry in the log. A tuple consists of the contact number involved in the event, the time of the event, and the type of event (*i.e.,* 1) outgoing call, 2) incoming answered, and 3) unanswered incoming call). The series events from a single contact are preprocessed to form a sequence of 5-tuple observations. The five observations in each tuple, per-contact, in a sequence are,

1. **in** - whether a call was received from the contact and if was answered by the user. The observation state is cleared on time-step in the sequence after the call has ended.
2. **miss** - whether a call from this contact has been missed. The observation is cleared if a



(a) Part of the monolithic model capturing the user's message response patterns with a single contact.



(b) Extending the single contact model capture the user's message response patterns with most top n contacts and all other contacts clumped in to a single contact.

Figure 9.3: Part of monolithic model to capturing the user's message response patterns with all contacts.

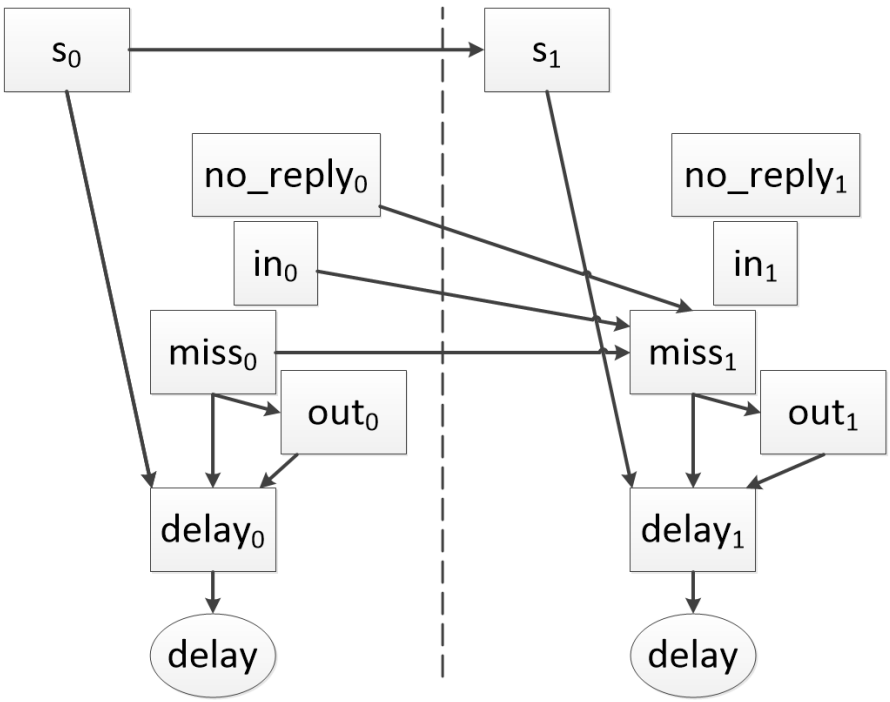
call from the same contact is answered or if an outgoing call is made to that user. This observation is also cleared after a time-out (see list entry on *no_reply* below).

3. **out** - whether an outgoing call was made to this contact by the user. Setting this observation also clears the *miss* observation of a missed call.
4. **delay** - the value for this observation is set to *undefined* (represented as -1) unless trigged by *out* while *miss* is also set. In which case, the delay is calculated as the time difference between last missed call from that contact and the user making an outgoing call to that user. The value of this observation remains set only for a single time-step of the observations sequence.
5. **no_reply** - observation whether a missed call from this contact has been completely ignored by the user. That is, a *miss* variable has been set for a extended period with another call from the same contact being answered or an outgoing call being made to that contact. The period before marking a missed call as *no_reply* was experimentally decided upon to be 1 hour. When creating the sequence of observations, setting the *no_reply* observation clears the *miss* observation (*i.e.,* the time-out mentioned in the description of *miss*). The *no_reply* observation is cleared after a single time-step in the sequence.

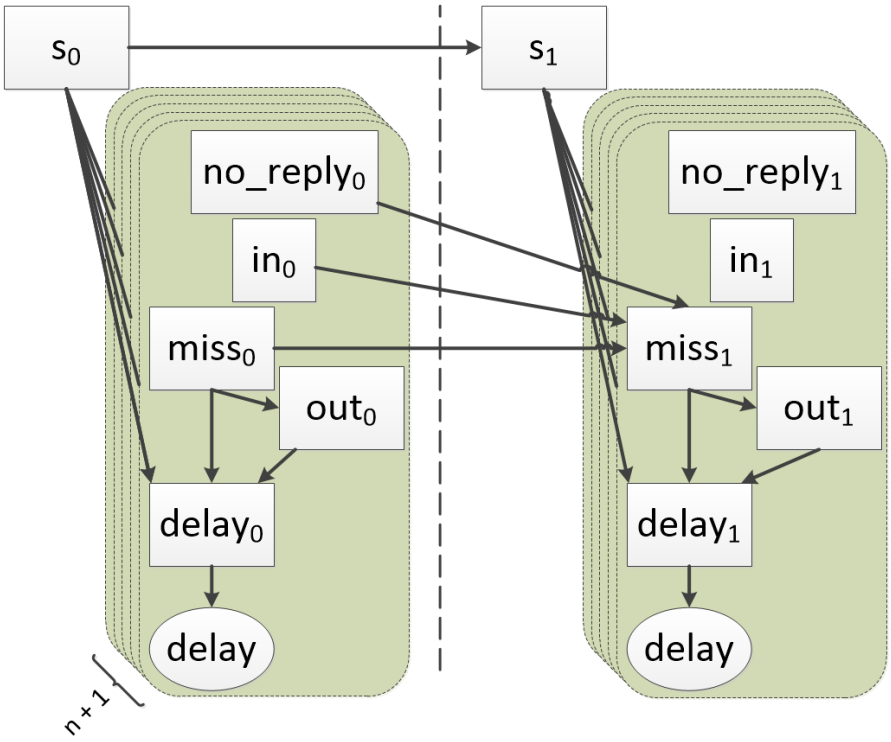
For each contact, these five observations are modeled as four discrete random variables (each for *in*, *out*, *miss*, and *no_reply*) and a continous random variable (for *delay*) in the monolithic model. As with the message response section of the monolithic model additional "dummy" discrete random variable nodes are introduced, labeled as *delay*. These observation variables, internal state variables are depicted in Figure 9.4a along with their relationships, in a similar way to how messaging pattern are modeled (see Figure 9.3a).

The partial single-contact model is extended to multiple contacts. The section of the monolithic model capturing the user's full calling behavior is depicted in Figure 9.4b).

Finally, the three sections of the model (*i.e.,* 1) mobility, 2) message response, and 3) calling) were brought together with a single shared state across the three models. The shared discrete



(a) Part of the monolithic model capturing the user's calling patterns with a single contact.



(b) Extending the single contact model capture the user's calling patterns with most top n contacts and all other contacts clumped in to a single contact.

Figure 9.4: Part of monolithic model to capturing the user's calling patterns with all contacts.

random variable across all three partial models is the user's state variable. The full monolithic model is depicted in Figure 9.5.

To detect anomalies using this model, a sequence of observations are extracted from the logged data. The sequence of observations are provided to the monolithic model and using Mocapy's Monte Carlo sampling approach, the likelihood of the observations being generated by the model is estimated. The likelihood is then compared to an "anomaly threshold" and if the likelihood is lower than the threshold an anomaly is flagged. The anomaly threshold can be decided upon by studying the model's receiver operator characteristic (ROC) curve and selecting a threshold which provides the right balance between true-positive rate and false positive rate for the application in which model is to be used.

9.2 Straightforward (Naive) Combination of Behavior Aspect Models

Another model used for performance comparison against McFAD is presented in this Section. This model was comprised of five behavior aspect models. These were the same models used in McFAD during this evaluation:

1. Geo-trace mobility model (see Section 3.3).
2. Outdoor mobility model (see Section 5.1).
3. Indoor mobility model (see Section 5.2).
4. Message response patterns model (see Section 4.1).
5. Call patterns model (see Section 4.2).

This combination of models decides on a final classification by a straightforward voting. This approach is identical to the voting approach used to combine the models using soft sensor information sources and radio channel information sources to compare how they stack up against

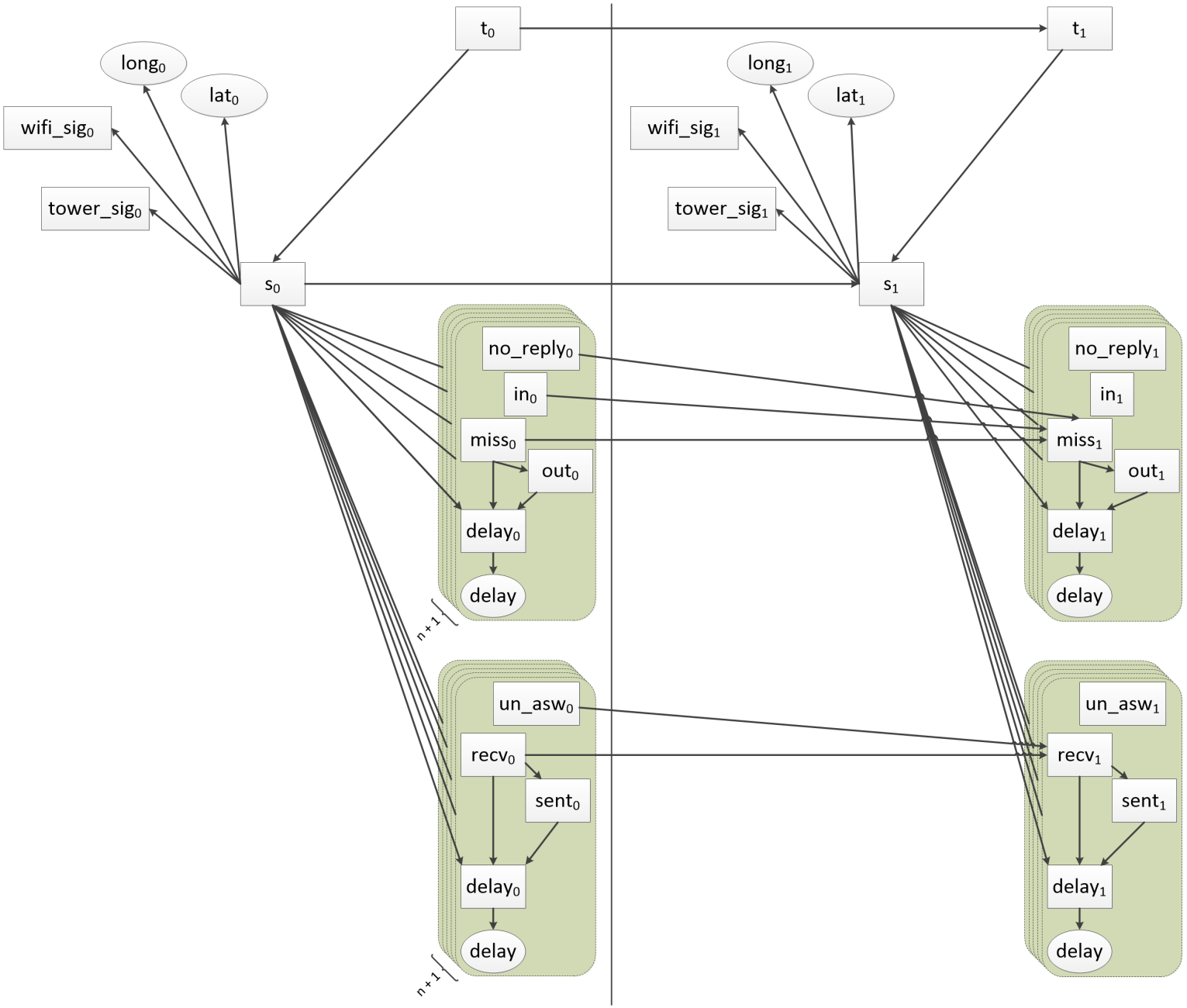


Figure 9.5: Complete monolithic model used for evaluation.

a single model using a hard sensor information source (see Chapter 6). In voting, any model that does not have sufficient data, abstains from casting a vote. In the case of a tied decision, the classifications of 1) the geo-trace mobility model, 2) the outdoor mobility model, 3) the indoor mobility model, and 4) the message response patterns model (in descending order) are given precedence (see Chapter 6 for details on how model precedence is decided).

9.3 Performance Evaluations

The evaluations presented in this section are aimed primarily at comparing the overall performance the three approaches discussed thus far, namely 1) the McFAD approach, 2) the monolithic model approach, and 3) the straightforward combination of behavior aspect models. The evaluation results are presented as ROC curves[2]which can be visually inspected to gauge the comparative performance of three approaches. This section also provides performance results, in the form of ROC curves, for the three models as they are trained with a range of training dataset sizes to show how the models training data requirements vary to achieve an acceptable level of performance.

The evaluations on these models were performed using the Dataset II (see subsection 2.3.2 for a detailed description of the dataset). All 3 months (12 weeks) of data from each of the 30 users in the dataset were used for the evaluations. Since the experiment also evaluate how the performance of each approach varies with the amount training data, the experiment was conducted five times, each with a different training dataset size. The training dataset sizes were:

- **1 week** - a 12-fold validation approach was used for each user, where one fold (*i.e.*, one week of data) is used to train a model to be tested. The other $\frac{11}{12}$ths of the user's data are used as testing data of *normal behavior*. To simulate *anomalous behavior* the complete data sets of the other 29 users in the dataset are used. This experimental procedure assumes that 1) during the data collection period each user did not have any substantial anomalies

[2] See Subsection 2.4.1 for a detailed description on how to interpret ROC curves.

in their own behavior, and 2) the anomalies that are of interest are significant to the extent of mirroring the behavior of a different user.

- **2 weeks** - the dataset was divided into six 2-week chunks. Similar to the 1 week training dataset size approach, a 6-fold validation was performed for each of the models.
- **4 weeks** - four weeks of training data were used to train each model, and the models were evaluated on the other 8 weeks of data. The experiment was repeated two more times, taking the each of the other (non-overlapping) four week chunks of data as the training dataset in turn (*i.e., a three-fold validation*).
- **6 weeks** - the dataset was divided into two six-week chunks. Two iterations of the experiment were performed for each user, using one of the chunks as training data and the other as testing data for *normal behavior*, and in the second iteration the roles of the chunks of data were reversed.
- **8 weeks** - once again a two-fold evaluation was performed. To train a model in the first iteration of the experiment, the first eight weeks of the user's data was used while the next four weeks of data was used as *normal behavior* data for testing. To train the models in the second iteration of the experiment, the first four weeks of data are kept aside to simulate *normal behavior* while the remaining eight weeks of data are used to train the models.

Results presented for each experiment are the average results across all users in the dataset and across all folds per-user.

To use the data streams of users to simulate anomalous behavior on the same device a few preprocessing steps were applied to the dataset. These changes are similar to the changes applied in the experiments described in Chapters 4 and 5. The changes made to each information source are as follows,

- **Messaging logs** - the numbers in each user's logs were replaced with numbers from a shared list. The process for this number substitution was to first sort the numbers in a user's logs according to the number of interactions the user has with each number in descending

order. Then each number was replaced with the number in the corresponding position in the shared list. The shared list was a list of randomly generated non-repeating numbers. In the case where a device is stolen and used to message numbers outside of the owner's contact list and messages from known contacts are ignored, the model will flag anomalies quickly. These experiments were designed to simulate harder to detect situations where the user's behavior has changed significantly or the device is stolen and being used to either phish for information about the device owner through contacts or impersonate the user with malicious intent.

- **Call logs** - as with the messaging logs the numbers in each user's logs were replaced with numbers from a shared list. Again, the process for this number substitution was to first sort the numbers in a user's logs according to the number of calling interactions the user has with each number in descending order. Then each number was replaced with the number in the corresponding position in the randomly generated list of non-repeating numbers. In the case where a device is stolen and used to call numbers outside of the owner's contact list and calls from known contacts are ignored, the model will flag anomalies quickly. These experiments simulate harder to detect situations where the user's behavior has changed significantly or impersonate the user with malicious intent.
- **WiFi scan results** - to test the WiFi signatures across users, the WiFi signature clustering and cluster labeling technique described in Section 5.2 is performed for each user separately. The labeling of clusters is done from a shared list of labels. Using the created clusters and cluster names WiFi scan data are used to create the sequences of cluster labels per-user, and these sequences are then shared across users. The case of major variations from routine where the user leaves known areas is straight-forward to detect, but these experiments simulate harder to detect cases where the user has broken from routine but remains within known areas (*e.g.,* within the user's college-campus).

9.3.1 Receiver Operator Characteristic (ROC) Curves

To compare the three approaches, the main tool used is the ROC curve. The ROC curves of each approach and training data set size can be compared with each other and also to an ideal anomaly detector's ROC curve. See Subsection 2.4.1 for a full description of ROC curves and refer to Figure 2.1 on how to estimate the performance of an approach independently of other curves. Eight different ROC curves are presented in this subsection, five of which are for comparison of the three approaches under varying training dataset size (*i.e.,* 1, 2, 4, 6, and 8 week(s) of training data). Each of the other three ROC curves shows, for a single anomaly detection approach, the performance gain as the training dataset size is increased.

Since the straightforward (naive) combination of models approach and McFAD both have five models, each approach has five anomaly thresholds (one for each model in the combination) that can be varied, thus the ROC curves created by these two approaches are hyper-surfaces if all possible threshold values are iterated over. For comparison, all five threshold values, of each approach, were linearly varied through their ranges to generate a *slice* of the ROC hyper-surface.

Figure 9.6 shows the ROC curves for the three approaches when trained on just a single week of user data. As all three curves are very close to the performance curve for random blind guessing (*i.e.,* assuming a 50% likelihood of an anomaly - coin toss) line (depicted in the figure as a dashed red curve). This indicates all three approaches performed very poorly with just one week of training data.

The ROC curves depicted in Figure 9.7 are for the performance of the three approaches when trained on two weeks of user data. The straightforward (naive) combination of models and McFAD show substantial performance improvements, but they are both far from an ideal anomaly detector when trained on just two weeks of user data. While these two approaches combined identical models, the somewhat large performance gap between the two approaches was seen to be caused by the difference in how the output of the classifiers is combined. Especially in the case of just two classifiers offering a result, the straightforward combination approaches static

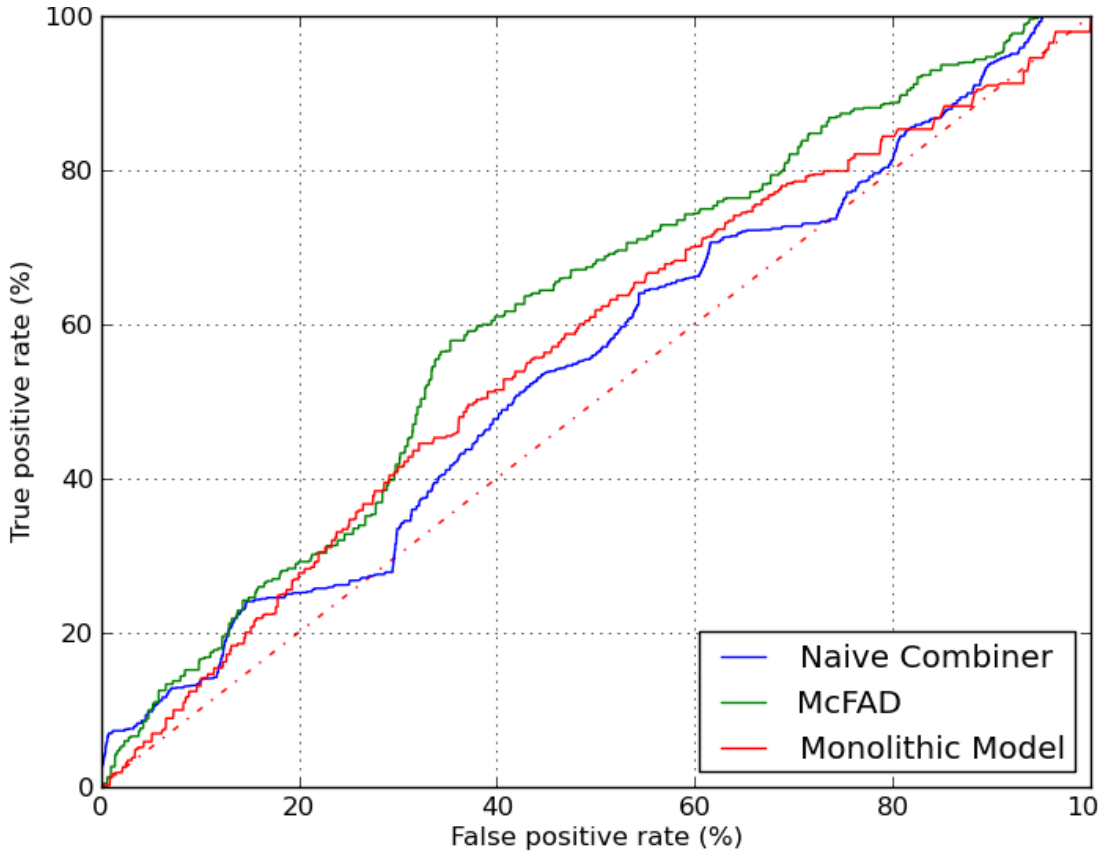


Figure 9.6: ROC curves of the three anomaly detection approaches, each approach trained with 1 week of user data.

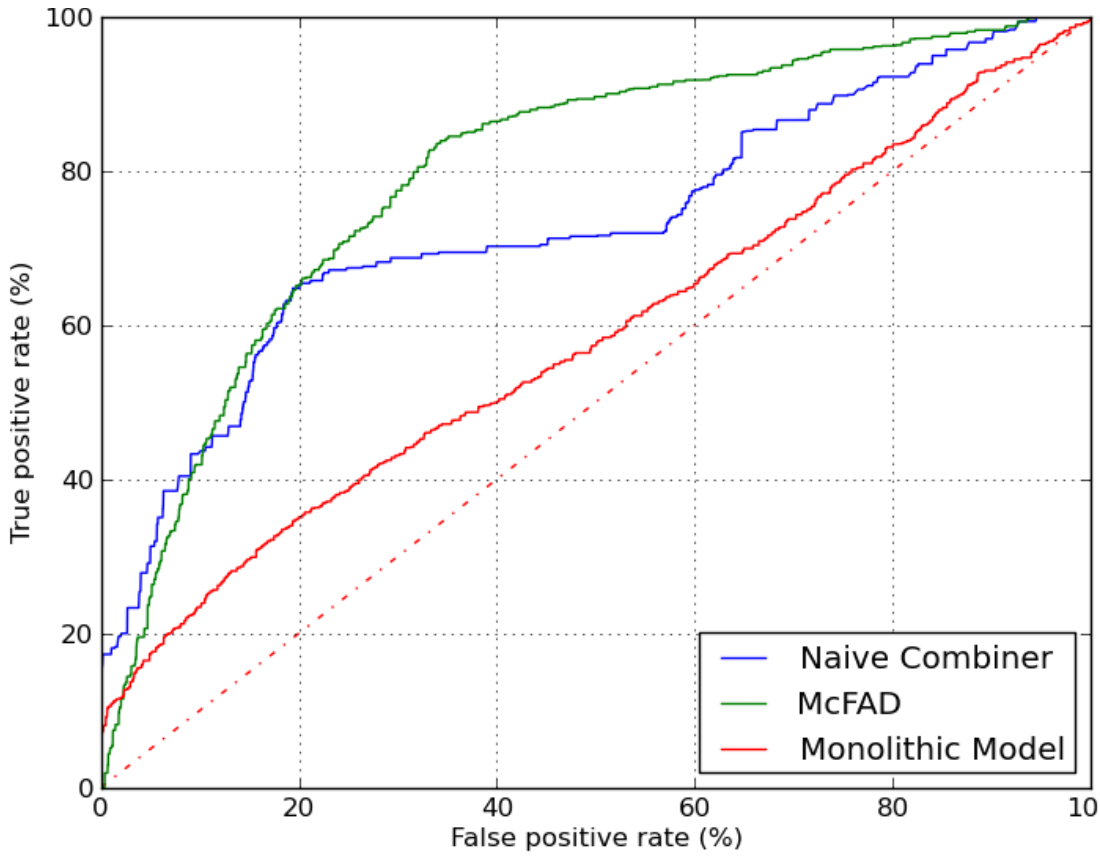


Figure 9.7: ROC curves of the three anomaly detection approaches, each approach trained with 2 weeks of user data.

weighting of models seemed to hinder its performance. The monolithic model on the other hand still shows quite low performance, still close to the performance curve for random blind guessing curve.

Figure 9.8 depicts the performance of the three anomaly detection approaches when trained with 4 weeks user data. With four weeks of training data, the monolithic models performance shows significant improvement, but still lags behind the other two approaches. Even with four weeks of training data the monolithic model is still slightly behind the McFAD approach trained with two weeks of user data (comparing McFAD's ROC curve from Figure 9.7). The straightforward combiner and McFAD both showed very good performance. The ROC curves for these two approaches are somewhat deceptively close to each other, and therefore it is worth comparing the performance of the two approaches with a few data points from the curves. For easier

Table 9.1: Performance comparison of the straightforward combiner and McFAD when trained on 4 weeks of data.

Point of comparison	Straightforward combiner	McFAD
80% **TPR**	12.46% FPR	8.49% FPR
20% **FPR**	94.64% TPR	97.96% TPR

comparison Table 9.1 lists the performance of these two approaches at the 80% true positive rate (TPR) mark and at the 20% false positive rate (FPR) mark.

Figures 9.9 and 9.10 show the ROC curves for the three approaches when each is trained with 6 and 8 weeks of user data, respectively. The ROC curves for McFAD showed a slight larger increase in performance over the performance improvement shown by the straightforward combiner approach from the 4 week training data results. This larger delta in performance gain shown by the McFAD approach is possibly due to increased training data providing better estimates for the confidence surfaces used to combine the decisions of the classifiers in the McFAD approach (see Section 7.2). While the monolithic model showed the largest gains in performance gains from the 4 week training data results, it still lagged behind the McFAD and straightforward (naive) combiner approaches (albeit by a smaller margin).

Figures 9.11, 9.12, and 9.13 show the ROC curves of the straightforward (naive) model combining approach, the monolithic model, and the McFAD approach, respectively, each for a single type of approach generated with training dataset sizes of 1, 2, 4, 6, and 8 weeks. The Figures 9.11 and 9.13 show that as the training dataset size was increased towards 4 weeks both approaches show significant performance improvements. Increasing the training dataset size further (to 6 and 8 weeks) seems to have improved the performance further, but the improvements seem to plateau. The monolithic model on the other hand showed more gradual improvements in performance as the training dataset size is increased, but the ROC curves for 6 and 8 weeks (in Figure 9.12) still show significant performance improvements.

In summary, the experimental results show that the McFAD approach provides best anomaly detection performance when trained with 8 or less weeks of user data (barring a few corner case

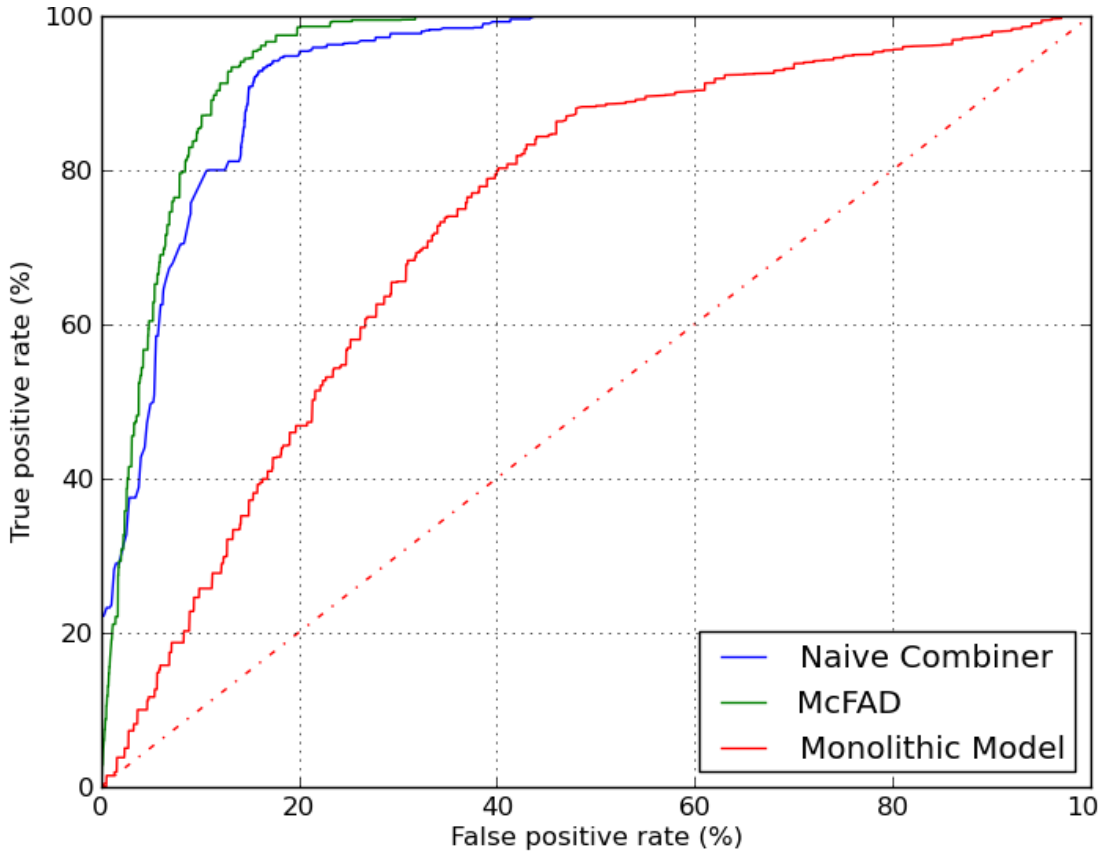


Figure 9.8: ROC curves of the three anomaly detection approaches, each approach trained with 4 weeks of user data.

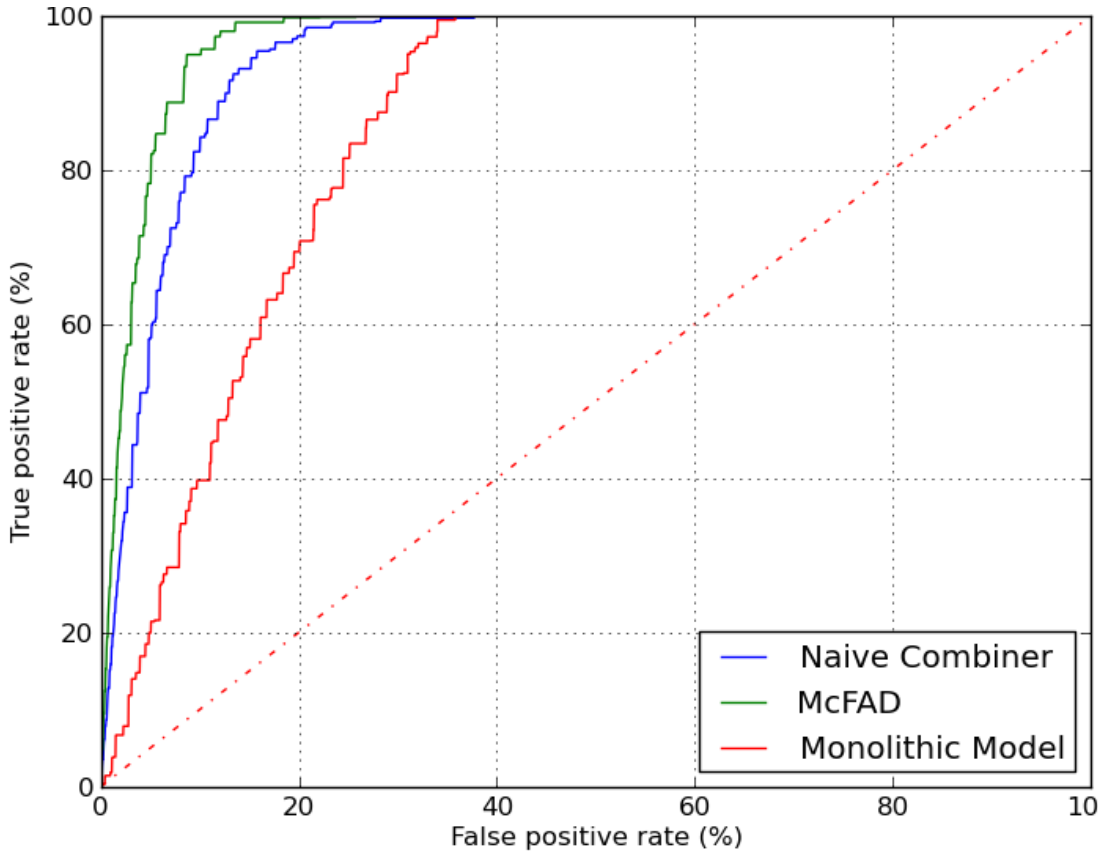


Figure 9.9: ROC curves of the three anomaly detection approaches, each approach trained with 6 weeks of user data.

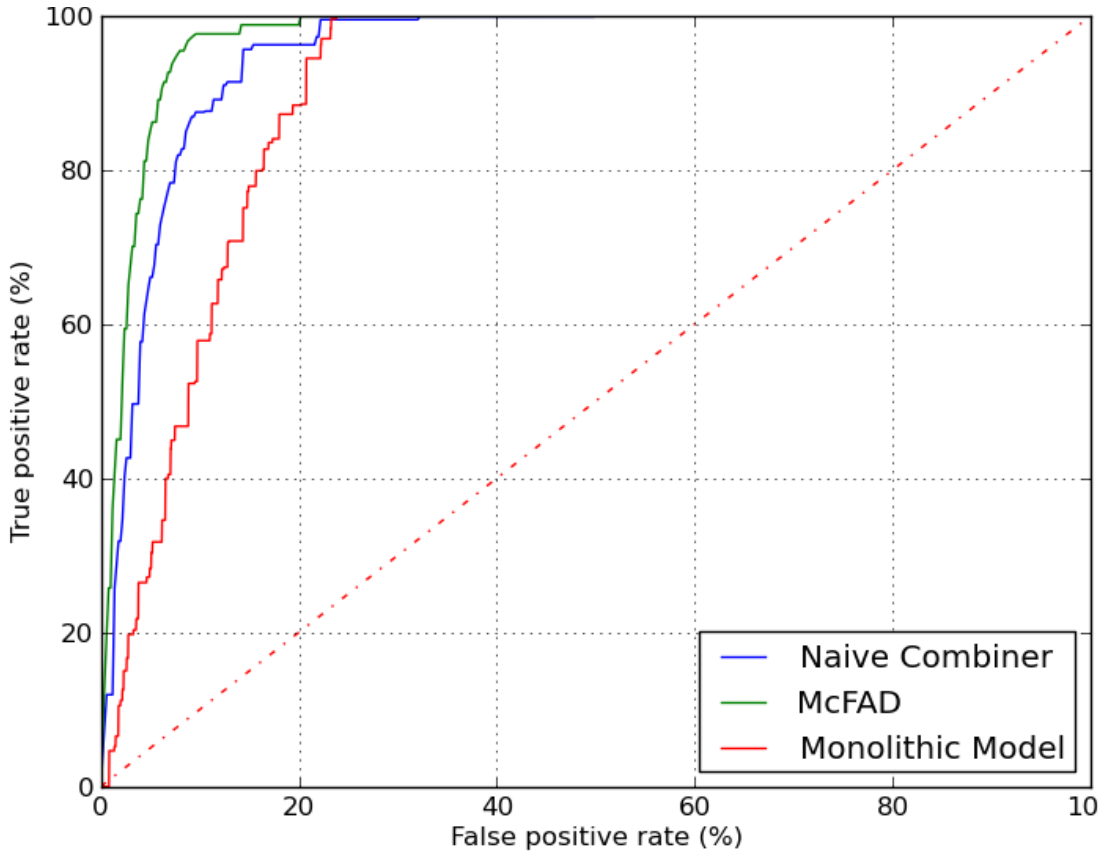


Figure 9.10: ROC curves of the three anomaly detection approaches, each approach trained with 8 weeks of user data.

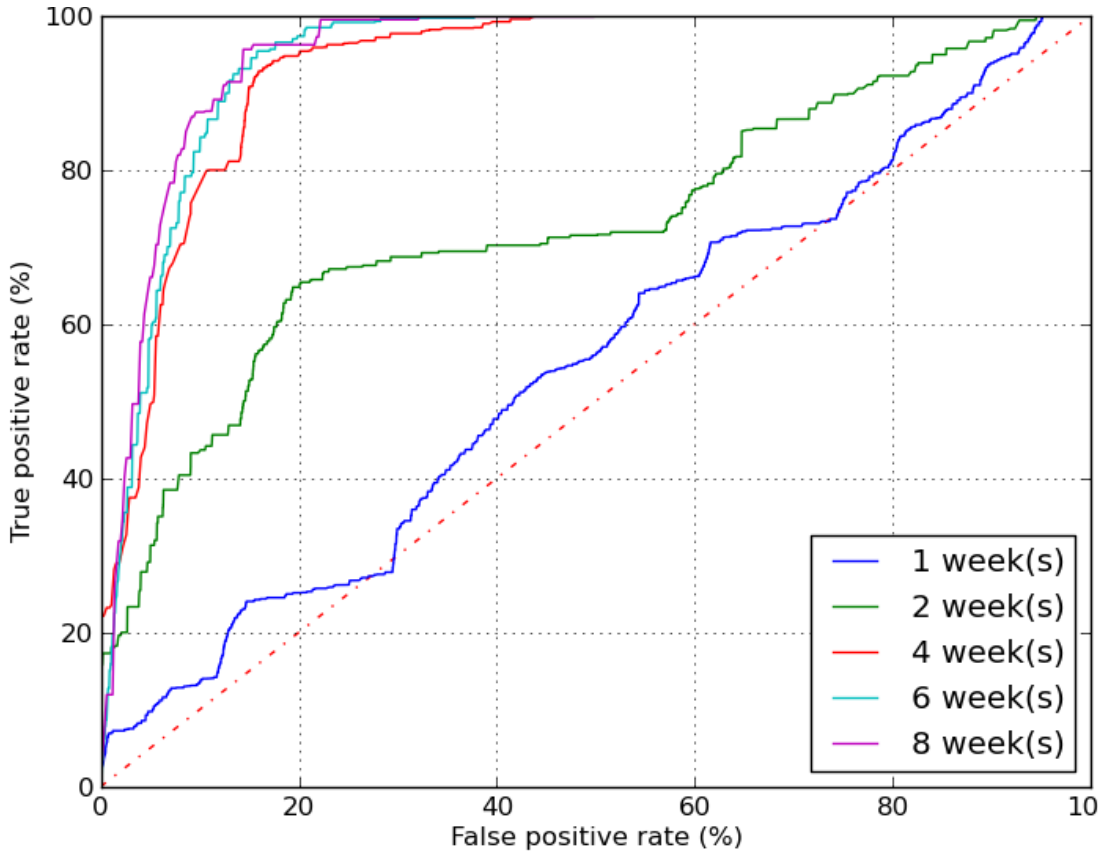


Figure 9.11: ROC curves of the straightforward (naive) combination of behavior aspect models approach as the training dataset size is varied from 1, 2, 4, 6, to 8 weeks.

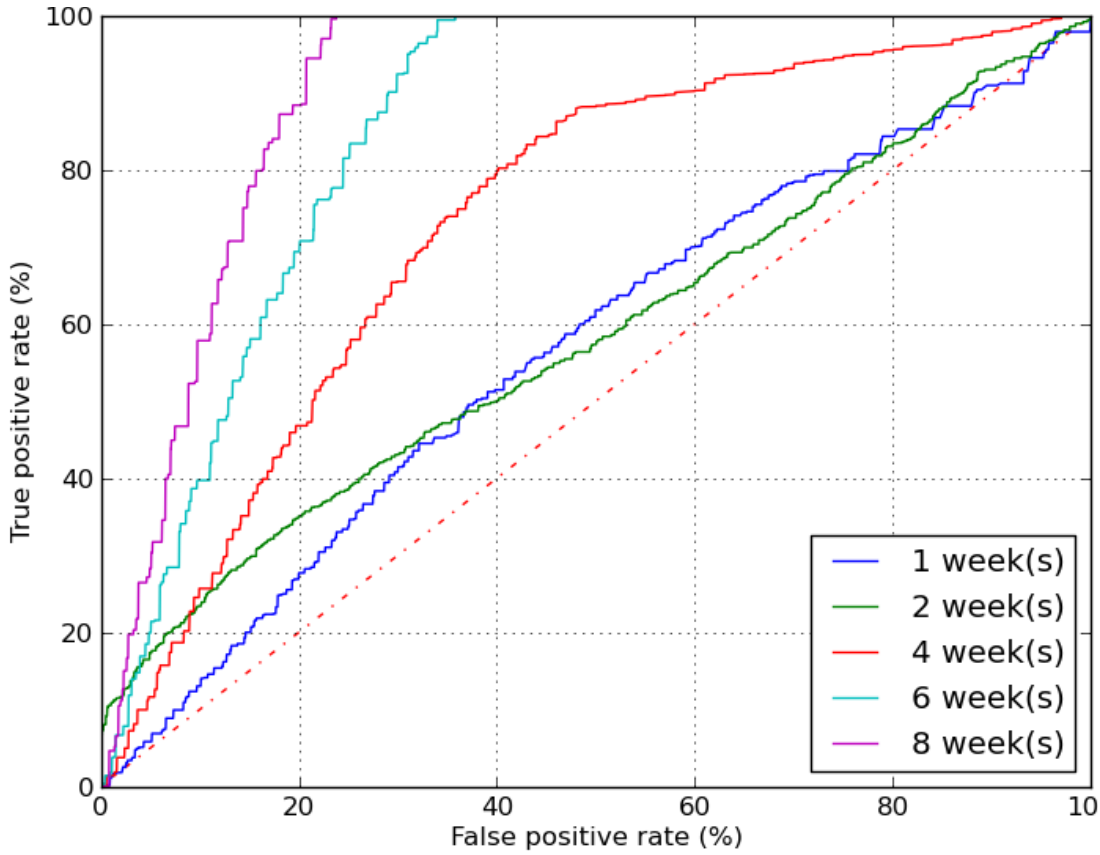


Figure 9.12: ROC) curves of the monolithic model approach as the training dataset size is varied from 1, 2, 4, 6, to 8 weeks.

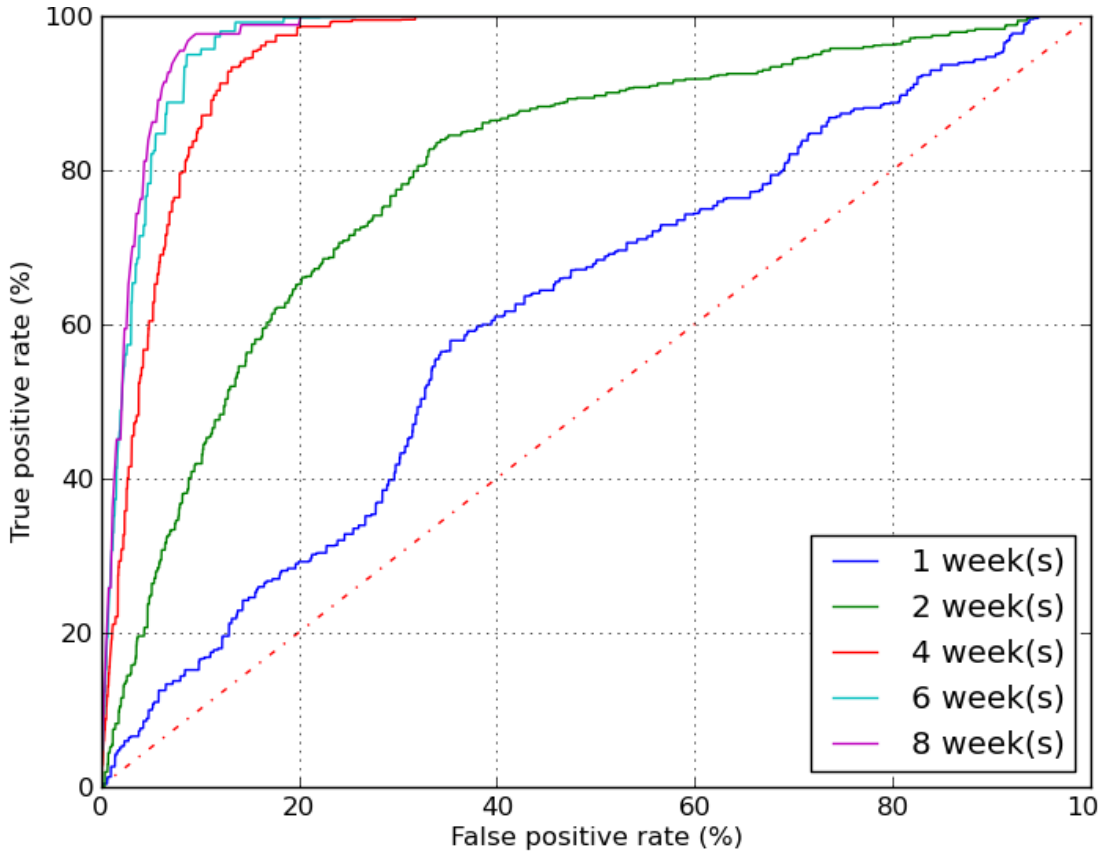


Figure 9.13: ROC curves of the Mobile Computing Framework for Anomaly Detection in User Behavior (McFAD) approach as the training dataset size is varied from 1, 2, 4, 6, to 8 weeks.

regions, *e.g.,* Figures 9.7, 9.8). The results also indicate that the training data requirements of the monolithic model to perform on par with the McFAD approach is larger than 8 weeks of user data. Considering the assumption of user routines evolving over time (see Section 1.2) and the anomaly detection approach having to retrain to the evolving behavior, McFAD shows the most adaptability while maintaining a higher level of anomaly detection performance.

9.3.2 Coverage Map

The preprocessing applied on the data streams (*i.e.,* the activity thresholds for the messaging response patterns model and the call patterns model and the label collapsing approaches in the three mobility models) sets minimum data requirements on each information stream for the individual classifiers in the straightforward combiner and the McFAD approach. These limits were kept identical for the models used in the straightforward combiner and the McFAD approach. To obtain a fair estimate of performance, the monolithic model was also tested only on the sequences of user behavior qualifying for anomaly detection by the other two approaches. Therefore the coverage map shown in Figure 9.14 represents the coverage for either of the three approaches.

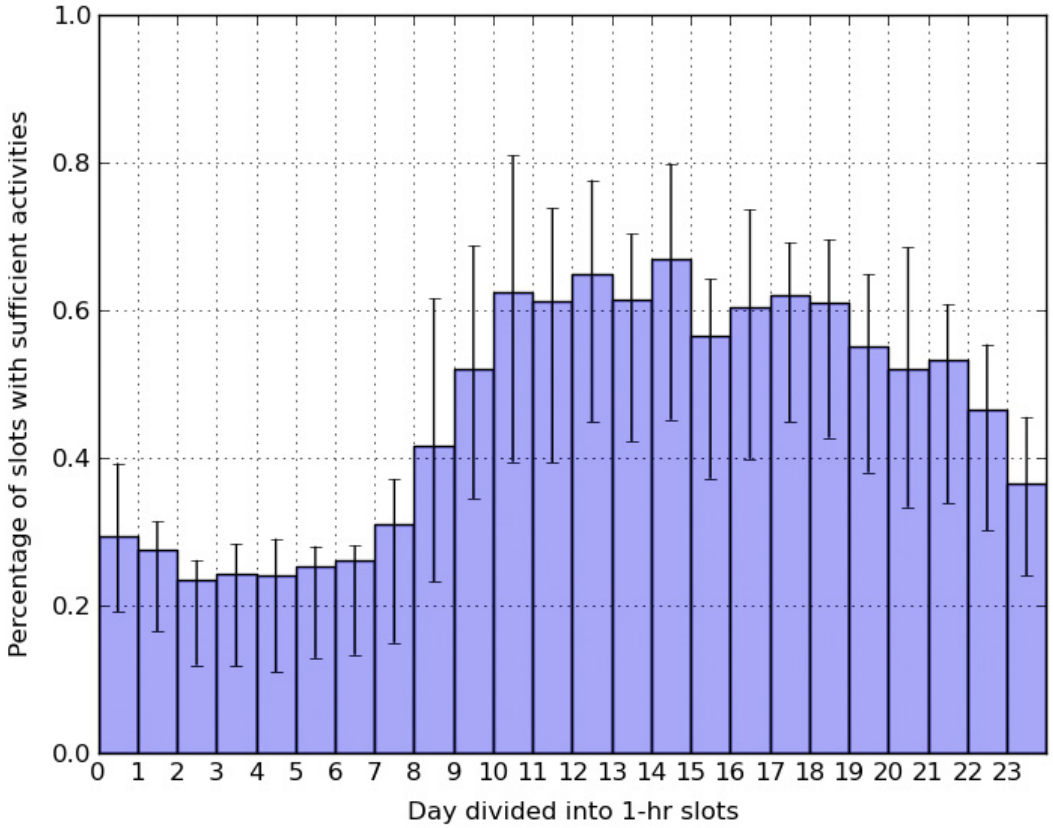


Figure 9.14: Coverage map for the straightforward (naive) combination of behavior aspect models, the monolithic model, and McFAD.

Chapter 10

Conclusion

In this final chapter, a summary of the work and the key contributions are presented. Finally, some areas that merit future research are discussed.

10.1 Research Summary

This research has developed a novel approach to modeling the behavior of a mobile device user with the specific aim of detecting anomalous behavior exhibited by the user. The developed approach is based on breaking down the user's behavior into smaller *aspects of behavior*, creating separate models for each aspect, and finally combining the models to gain a holistic picture of the user's routine. This is in contrast to the traditional idea of modeling user behavior with a single monolithic model. By following the presented modeling approach, each model can be trained using only positive examples of the user's routine behavior, making the real-world bootstrapping of models to new users extremely straightforward.

This research also devised a new ensemble learning approach Confidence-based Learning Ensembles (CobLE) for combining the behavior aspect models each operating on a asynchronous set of information sources, by learning the areas of a feature space where each model excels or fails it dynamically weights the output of the models for consensus. This learning approach is

able to combine any type of model, and account for missing feature values (or sets) presented to some models. Experimental results also showed CobLE can be used to combine learners for other applications, especially when there is asynchronousness or unreliability in the features being provided to the classifiers.

An anomaly detection framework, Mobile Computing Framework for Anomaly Detection in User Behavior (McFAD), was implemented embodying the new modeling approach and the combining approach to gauge their applicability on real-world mobile devices. Because of CobLE, McFAD can be easily extended to incorporate new behavior aspect models, without any changes being made to the existing models. Therefore McFAD allows easy integration of 1) new aspects of behavior, and 2) new information sources for anomaly detection simply by adding in models that use the new information sources. The implementation also identified and addressed the engineering issues in implementing an extensible anomaly detection system on a mobile device.

The McFAD framework was throughly evaluated and contrasted against the concept of using a monolithic learner. The evaluation addressed the performance of the different approaches, the bootstrapping data requirements (*i.e.,* training data requirements), and the evolution of performance over time. From evaluation results it was shown that the McFAD system bootstraps faster (*i.e.,* requiring less training data) than previous approaches, has better anomaly detection performance, and maintains this performance by adapting to evolving user behavior.

10.1.1 Recap of Chapters

In Chapter 1 of this dissertation the idea of detecting anomalous behavior in a mobile device user's routine was introduced. A key idea is creation of model(s) of user behavior to automate the task of detecting anomalous behavior. Also presented were, 1) the specific goals in creating models which could automate this task, 2) the constraints under which such models would have to operate, and 3) the spefic assumptions made creating models for anomaly detection.

In Chapter 2, the notion of capturing "aspects of behavior" as a model approach to build an overall model of mobile user behavior was introduced. As a motivating example, a model capturing the aspect of "who the use met during specific events" over the course of the user's day was discussed. Underlying all the research are the many data sources presently available on mobile devices, which are broadly categorized as,

- hard-sensor information sources,
- soft-sensor information sources, and
- radio channel information sources

For each type of information source, several aspects of user behavior can be modeled. The evaluation metrics,

1. Performance of anomaly detection
2. Coverage throughout the day

, used for the rest of this dissertation was presented along with a description of the datasets used in evaluations in this dissertation.

Chapters 3, 4, and 5 each reviewed specific models of aspects of user behavior using each of the information classes listed.For each model, the optimal parameters were explored as well the performance and coverage was assessed.From these assessments it was observed that individual models capturing a single aspect of user behavior could not provide satisfactory coverage throughout the day.

In Chapter 6, the idea of combining multiple models of behavioral aspects was explored. By using a straightforward voting approach, it was shown that a small collection of models could perform on-par or even better than a single more computationally complex and power consuming model. It was also observed that a combination of models could provide far greater anomaly detection coverage throughout a user's day than a single model. The results from this chapter motivated the exploration for a more sophisticated approach for combining models of behavioral aspects to achieve better anomaly detection performance whilst 1) maintaining good coverage

and 2) be easily extend-able by adding new behavioral aspect models.

Chapter 7 presented a novel ensemble learning approach, Confidence-based Learning Ensembles (CobLE), for combining classifiers in a variety of circumstances. The circumstances under which the ensembles are created is a combination of one or more of the following scenarios,

1. where the classifiers operate in independent feature spaces
2. where the classifiers operate in a single unified feature space
3. where the classifiers receive data asynchronously forcing the classifiers to operate asynchronously
4. where the input feature vectors to classifiers are missing feature values
5. where the classifiers are of different types.

A formal proof and bounds for the CobLE approach were discussed in the chapter. This chapter also presented a comparison of performance between CobLE and other ensemble learning (and single model) approaches on public dataset. The results from these experiments showed that, in the scenarios presented previously, CobLE outperformed other learning approaches and therefore would be an ideal approach for combining behavioral aspect models.

Mobile Computing Framework for Anomaly Detection in User Behavior (McFAD) is a framework for bringing together behavioral aspect models using models using the CobLE approach, was presented in Chapter 8. This chapter presented the framework as an implementation on the Android platform and showed how it could 1) be extended by adding new behavioral aspect models, 2) used by applications on the mobile device to leverage the framework's anomaly detection capabilities.

Chapter 9 presents a thorough evaluation of the McFADframework against a single monolithic model and a straightforward voting approach of combining behavioral aspect models for the task of anomaly detection in the user's routine. The results from these evaluations show that,

1. McFAD framework provides the best performance for anomaly detection in mobile user behavior (over other approaches)

2. McFAD framework is the best suited at updating the behavior model(s) as the user's routine organically evolves over time.

10.2 Open Questions and Future Work

To detect anomalous behavior, the approach developed in this research estimates the likelihood of the current and a finite number of past states of the user. Where as predicting the behavior of a user is to pick the most like next state of the user. Considering the effectiveness of modeling aspects of behavior for anomaly detection, an interesting extension would be to adapt it for behavior prediction. Novel methods would need to be developed so each aspect of behavior model can improve the prediction capabilities of the other models.

Another avenue of research warranting exploration is identifying the cause of the detected anomalies, *e.g.,* medical emergencies, accidents, theft of device. Since collecting data for anomalies of each cause is probably not feasible. An initial rule-based system which identifies the cause based on the anomaly detection modules detecting anomalous behavior is a possible starting point (*e.g.,* the geo-trace model and the calling patterns model flagging an anomaly could be taken to mean a device theft). Improving such a rule-based system "in the wild" (once the system is deployed) by sharing feedback between users could be a interesting extension to this work.

10.3 Contributions

The following research contributions were made by this research:

1. **Concept of modeling behavior aspects :** The idea of modeling smaller aspects of a user's routine in separate models, and then finally combining these models to gain a holistic model of the user's behavior routines was developed. Modeling simpler aspects of a user's routine allows each model to be trained with limited data. In contrast to traditional monolithic modeling approaches, by combining these aspect models a more holistic pic-

ture of a user's routine created using less training data. Since the number of information sources used by a aspect-based model is generally far smaller than that of a monolithic model, the likelihood that at least one aspect model will have reliable information on all (or most) of its sources is far higher than that for a monolithic model.

2. **CobLE ensemble learning approach :** CobLE is able to combine any type of classifier, and account for missing feature values (or sets) presented to some classifiers through dynamically weighting the output of classifiers for aggregation by learning the areas of a feature space where each classifier excels or fails. CobLE can also be used to combine learners for other applications, especially when there is asynchronousness or unreliability in the features being provided to the classifiers.

3. **Methodology for creating behavior models trained on purely positive examples of routine behavior :** The models developed in this research can be trained by providing only positive examples of the user's routine behavior. This removes the need for synthetic anomalies to be created and presented to the model at training, making the data collection process to bootstrap a model for a new user extremely straightforward.

4. **McFAD :** The McFAD framework can be easily extended to incorporate new behavior aspect models without any changes being made to the existing models (nor requiring the existing models to be retrained). Since, new (or already modeled) aspects of behavior can be modeled and added for anomaly detection without additional training being performed, it allows new information sources to be integrated for anomaly detection simply by adding in models that use the new information sources. The implementation also identifies and addresses the engineering issues in implementing an extensible anomaly detection system on a mobile device.

5. **A comparison of performance :** The anomaly detection approach presented was throughly evaluated and contrasted against the traditional approaches (such as using a monolithic learner) looking not only the performance of the different approaches, but looks also at the

bootstrapping data requirements (*i.e.,* training data requirements) and evolution of performance over time.

6. **Behavior aspect models :** This dissertation presents multiple behavior aspect models using a variety of information sources available on a mobile device. The models focus on each using a single information source, which fall into the categories of hard, soft, or radio channel information sources. Each model is thoroughly evaluated using an evaluation methodology developed in this dissertation, in order to compare user anomaly detection approaches.

10.4 Conclusions

When combining the output of an ensemble of learners, taking into consideration the how well each learner is estimated to perform in the current region of its feature space 1) increases the overall accuracy, 2) makes the ensemble more robust. This hypothesis holds true even when the the learners operate on asynchronous and/or separate feature spaces.

For detecting anomalous behavior, modeling aspects of user behavior separately and then combining the output from the models taking into consideration how each model performs in the current region of its feature space, 1) reduces the training data requirement, 2) increases coverage, and 3) increases performance.

Bibliography

[1] AIPPERSPACH, R., COHEN, E., AND CANNY, J. Modeling human behavior from simple sensors in the home. In *Pervasive 2006: Proceedings of the Fourth international Conference on Pervasive Computing*, K. Fishkin, B. Schiele, P. Nixon, and A. Quigley, Eds., vol. 3968 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 337–348. 2

[2] ASHBROOK, D., AND STARNER, T. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing 7*, 5 (2003), 275–286. 1, 3.1, 3.2, 3.4.1, 3.6

[3] BACHE, K., AND LICHMAN, M. UCI machine learning repository, 2013. 7.5.3

[4] BELLMAN, R. E., AND DREYFUS, S. E. Applied dynamic programming. 2.1

[5] BHATTACHARYA, A., AND DAS, S. K. Lezi-update: an information-theoretic framework for personal mobility tracking in pcs networks. *Wireless Networks 8*, 2/3 (2002), 121–135. 3.1, 1

[6] BRUSH, A. J., KRUMM, J., AND SCOTT, J. Exploring end user preferences for location obfuscation, location-based services, and the value of location. In *UbiComp '10: Proceedings of the Twelfth ACM International Conference on Ubiquitous Computing* (2010), ACM, pp. 95–104. 3.2.1

[7] BUI, H. H. A general model for online probabilistic plan recognition. In *IJCAI: Proceedings of the International Joint Conference on Artificial Intelligence* (2003), vol. 3, Citeseer,

pp. 1309–1315. 2.1

[8] BUTHPITIYA, S., DEY, A. K., AND GRISS, M. Coble : Confidence based learning ensembles. In *CSCI '14: Proceedings of the 2014 International Conference on Computational Science and Computational Intelligence* (2014). 7

[9] BUTHPITIYA, S., DEY, A. K., AND GRISS, M. Soft authentication with low-cost signatures. In *PerCom '14: Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communications* (March 2014), pp. xx–xx. 4, 5

[10] BUTHPITIYA, S., ZHANG, Y., DEY, A., AND GRISS, M. n-gram geo-trace modeling. In *Pervasive 2011: Proceedings of the Ninth International Conference on Pervasive Computing*, K. Lyons, J. Hightower, and E. Huang, Eds., vol. 6696 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, pp. 97–114. 3.1, 5.1, 6.1

[11] CHANDOLA, V., BANERJEE, A., AND KUMAR, V. Anomaly detection: A survey. *ACM Comput. Surv. 41*, 3 (2009), 1–58. 1, 3.6

[12] CHEN, D., AND CHENG, X. An asymptotic analysis of some expert fusion methods. *Pattern Recognition Letters 22*, 8 (2001), 901–904. 7.1

[13] CHEN, G., AND KOTZ, D. A survey of context-aware mobile computing research. Tech. rep., Hanover, NH, USA, 2000. 3.6

[14] CHENG, P.-W., CHENNURU, S., BUTHPITIYA, S., AND ZHANG, Y. A language-based approach to indexing heterogeneous multimedia lifelog. In *ICMI-MLMI '10: Proceedings of the Twelfth International Conference on Multimodal Interfaces and the Seventh Workshop on Machine Learning for Multimodal Interaction* (2010), pp. 26:1–26:8. 3.1

[15] CHICKERING, D. M., HECKERMAN, D., AND MEEK, C. A Bayesian approach to learning Bayesian networks with local structure. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence* (1997), Morgan Kaufmann Publishers Inc., pp. 80–89. 9.1

[16] DARRAH, C., ENGLISH-LUECK, J., AND FREEMAN, J. Families at work: An ethnography of dual career families. *Report for the Sloane Foundation* (2001), 98–6. 4.1, 4.2

[17] DAVIDOFF, S. *Routine as Resource for the Design of Learning Systems*. PhD thesis, Carnegie Mellon University, 2011. 4.1, 4.2

[18] DAVIDOFF, S., ZIMMERMAN, J., AND DEY, A. K. How routine learners can support family coordination. In *CHI '10: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2010), pp. 2461–2470. 2

[19] DERAWI, M., NICKEL, C., BOURS, P., AND BUSCH, C. Unobtrusive user-authentication on mobile phones using biometric gait recognition. In *IIH-MSP 2010: Proceedings of the Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (2010), pp. 306–311. 2.1

[20] DUONG, T. V., BUI, H. H., PHUNG, D. Q., AND VENKATESH, S. Activity recognition and abnormality detection with the switching hidden semi-Markov model. *Computer Vision and Pattern Recognition 1* (2005), 838–845. 3.6

[21] DUTTON, G. Encoding and handling geospatial data with hierarchical triangular meshes. In *Proceeding of Seventh International Symposium on Spatial Data Handling* (1996), vol. 43, Netherlands: Talor & Francis. 3.2.1

[22] ESKIN, E., ARNOLD, A., PRERAU, M., PORTNOY, L., AND STOLFO, S. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Applications of Data Mining in Computer Security* (2002). 3.6

[23] FREUND, Y., SCHAPIRE, R. E., ET AL. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning* (1996), vol. 96, pp. 148–156. 1, 7.4, 4

[24] FRIEDMAN, J. H. On bias, variance, 0/1loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery 1*, 1 (1997), 55–77. 2.1

[25] FRIEDMAN, N., NACHMAN, I., AND PEÉR, D. Learning Bayesian network structure from massive datasets: the "sparse candidate" algorithm. In *UAI '99: Proceedings of the Fifteenth conference on Uncertainty in Artificial Intelligence* (1999), Morgan Kaufmann Publishers Inc., pp. 206–215. 9.1

[26] GAFUROV, D., SNEKKENES, E., AND BOURS, P. Gait authentication and identification using wearable accelerometer sensor. In *2007 IEEE Workshop on Automatic Identification Advanced Technologies* (2007), pp. 220–225. 2.1

[27] GEHRKE, J., AND WOJTUSIAK, J. Traffic prediction for agent route planning. In *Computational Science at ICCS 2008*, M. Bubak, G. van Albada, J. Dongarra, and P. Sloot, Eds. 2008, pp. 692–701. 3

[28] GHAHRAMANI, Z. Learning dynamic Bayesian networks. In *Adaptive Processing of Sequences and Data Structures*. Springer, 1998, pp. 168–197. 9.1

[29] GIANNOTTI, F., NANNI, M., PINELLI, F., AND PEDRESCHI, D. Trajectory pattern mining. In *KDD '07: Proceedings of the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2007), pp. 330–339. 2.1, 3.1, 3.3.3, 3.6

[30] GILKS, W. R., RICHARDSON, S., AND SPIEGELHALTER, D. J. Introducing Markov chain Monte Carlo. In *Markov chain Monte Carlo in Practice*. Springer, 1996, pp. 1–19. 9.1

[31] GONZALEZ, M. C., HIDALGO, C. A., AND BARABASI, A.-L. Understanding individual human mobility patterns. *Nature 453* (2008), 779–782. 3.1

[32] GRUTESER, M., AND GRUNWALD, D. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03: Proceedings of the First International Conference on Mobile Systems, Applications and Services* (2003), pp. 31–42. 3.6

[33] GUPTA, M., INTILLE, S. S., AND LARSON, K. Adding GPS-control to traditional thermostats: An exploration of potential energy savings and design challenges. In *Pervasive '09: Proceedings of the Seventh International Conference on Pervasive Computing* (2009),

Springer-Verlag, pp. 95–114. 3.1, 1

[34] HALL, D. L., AND LLINAS, J. An introduction to multisensor data fusion. *Proceedings of the IEEE 85*, 1 (1997), 6–23. 7.1

[35] HANSEN, L., AND SALAMON, P. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 10 (1990), 993–1001. 7.2

[36] HANSEN, P. R. In-sample fit and out-of-sample fit: Their joint distribution and its implications for model selection. Working paper at Department of Economics, Stanford University, 2009. 7.4

[37] HODGE, V., AND AUSTIN, J. A survey of outlier detection methodologies. *Artificial Intelligence Review 22* (2004), 85–126. 1, 3.6

[38] HUGHES, G. On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory 14*, 1 (1968), 55–63. 7.4

[39] JORDAN, M. I., GHAHRAMANI, Z., JAAKKOLA, T. S., AND SAUL, L. K. *An introduction to variational methods for graphical models*. Springer, 1998. 9.1

[40] KLASNJA, P., CONSOLVO, S., CHOUDHURY, T., BECKWITH, R., AND HIGHTOWER, J. Exploring privacy concerns about personal sensing. In *Pervasive 2009: Proceedings of the Seventh International Conference on Pervasive Computing* (2009). 3.2.1

[41] KRUMM, J. Inference attacks on location tracks. In *Pervasive Computing*, A. LaMarca, M. Langheinrich, and K. Truong, Eds. 2007, pp. 127–143. 3.1, 3.2.1

[42] KRUMM, J. A markov model for driver turn prediction. *SAE SP 2193*, 1 (2008). 3.6

[43] KRUMM, J., AND HORVITZ, E. Predestination: Inferring destinations from partial trajectories. In *UbiComp 2006: Proceedings of the ACM International Conference on Ubiquitous Computing* (2006), vol. 4206/2006, pp. 243–260. 1, 3.3.1, 3, 3.6, 5.1

[44] LIU, G., AND MAGUIRE, JR., G. A class of mobile motion prediction algorithms for wireless mobile computing and communication. *Mobile Networks and Applications 1*, 2

(1996), 113–121. 3.1

[45] LIU, H., DARABI, H., BANERJEE, P., AND LIU, J. Survey of wireless indoor positioning techniques and systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 37*, 6 (2007), 1067–1080. 5.2

[46] LIU, T., BAHL, P., AND CHLAMTAC, I. Mobility modeling, location tracking, and trajectory prediction in wireless atm networks. *IEEE Journal on Selected Areas in Communications SAC 16*, 6 (1998), 922–936. 3.1, 1

[47] MAGNUSSON, M. S. Discovering hidden time patterns in behavior: T-patterns and their detection. *Behaviour Research Methods Instruments and Computers 32*, 1 (2000), 93–110. 2.1, 3.1, 3.3.3, 3.3

[48] MOORE, L. K. *Emergency Alert System (EAS) and All-Hazard Warnings*. DIANE Publishing, 2011. 4

[49] MUHLBAIER, M., TOPALIS, A., AND POLIKAR, R. Ensemble confidence estimates posterior probability. In *Multiple Classifier Systems*. Springer, 2005, pp. 326–335. 7.1, 2

[50] MURPHY, K. P. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, 2002. 9.1

[51] PALUSZEWSKI, M., AND HAMELRYCK, T. Mocapy++ - a toolkit for inference and learning in dynamic Bayesian networks. *BMC Bioinformatics 11*, 1 (2010), 126. 9.1

[52] PARIKH, D., AND POLIKAR, R. An ensemble-based incremental learning approach to data fusion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37*, 2 (2007), 437–450. 7.1

[53] PATCHA, A., AND PARK, J.-M. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks 51*, 12 (2007), 3448–3470. 3.6

[54] PATTERSON, D. J., LIAO, L., FOX, D., AND KAUTZ, H. Inferring high-level behavior

from low-level sensors. In *UbiComp 2003: Proceedings of the Fifth International Conference on Ubiquitous Computing* (2003), vol. 2864/2003, pp. 73–89. 2.1, 9.1

[55] PATTERSON, D. J., LIAO, L., GAJOS, K., COLLIER, M., LIVIC, N., OLSON, K., WANG, S., FOX, D., AND KAUTZ, H. Opportunity knocks: A system to provide cognitive assistance with transportation services. In *UbiComp 2004: Ubiquitous Computing*. Springer, 2004, pp. 433–450. 1.1, 1.4, 2.1, 9.1

[56] POLIKAR, R. Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE 6*, 3 (2006), 21–45. 7.1, 2

[57] RAO, N. S. V. On fusers that perform better than best sensor. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23*, 8 (2001), 904–909. 7.1

[58] RAUDYS, Š. Trainable fusion rules. i. large sample size case. *Neural Networks*, 10 (2006), 1506–1516. 7.1

[59] SALVADOR, S., PHILIP CHAN, P., AND BRODIE, J. Learning states and rules for time series anomaly detection. Tech. rep., Department of Computer Sciences, Florida Institute of Technology, Melbourne, FL, 2004. 3.3.4, 3.6

[60] SCHAPIRE, R. E., AND SINGER, Y. Improved boosting algorithms using confidence-rated predictions. *Machine learning 37*, 3 (1999), 297–336. 7.1, 1, 7.4

[61] SCOTT, J., KRUMM, J., MEYERS, B., BRUSH, A. J., AND KAPOOR, A. Home heating using gps-based arrival prediction. Tech. rep., Microsoft Research, USA, 2010. 2.1, 1

[62] SHANNON, C. E. A mathematical theory of communication. Tech. rep., Bell Systems Technical Journal, 1948. 3.3.1

[63] SHANNON, C. E. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review 5*, 1 (2001), 3–55. 3.1

[64] SHI, E., NIU, Y., JAKOBSSON, M., AND CHOW, R. Implicit authentication through learning user behavior. In *Information Security*, M. Burmester, G. Tsudik, S. Magliveras, and

I. Ilic, Eds., vol. 6531. 2011, pp. 99–113. 1.1, 2.1, 3.6

[65] SHI, X., AND MANDUCHI, R. A study on Bayes feature fusion for image classification. In *Computer Vision and Pattern Recognition Workshop, 2003.* (2003), vol. 8, IEEE, pp. 95–95. 7.1, 3

[66] SIEBERT, J. P. Vehicle recognition using rule based methods. Tech. rep., March 1987. 7.1

[67] STOCK, J. H., AND WATSON, M. W. Forecasting output and inflation: the role of asset prices. Tech. rep., National Bureau of Economic Research, 2001. 7.4

[68] SUN, Q., ZENG, S.-G., LIU, Y., HENG, P.-A., AND XIA, S. A new method of feature fusion and its application in image recognition. *Pattern Recognition 38*, 12 (2005), 2437–2448. 7.1, 3

[69] TAX, D. M., VAN BREUKELEN, M., DUIN, R. P., AND KITTLER, J. Combining multiple classifiers by averaging or by multiplying? *Pattern Recognition 33*, 9 (2000), 1475–1485. 7.1

[70] VERIKAS, A., LIPNICKAS, A., MALMQVIST, K., BACAUSKIENE, M., AND GELZINIS, A. Soft combination of neural classifiers: A comparative study. *Pattern Recognition Letters 20*, 4 (1999), 429–444. 7.1, 1

[71] WALTZ, E., LLINAS, J., ET AL. *Multisensor data fusion*, vol. 685. Artech House Norwood, 1990. 7.1

[72] WOLPERT, D. H. Stacked generalization. *Neural Networks 5*, 2 (1992), 241–259. 7.3

[73] XIANG, T., AND GONG, S. Video behaviour profiling and abnormality detection without manual labeling. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision* (2005), pp. 1238–1245. 3.6

[74] XU, L., KRZYZAK, A., AND SUEN, C. Y. Methods of combining multiple classifiers and their applications to handwriting recognition. *Systems, Man and Cybernetics, IEEE Transactions on 22*, 3 (1992), 418–435. 7.1

[75] ZHAI, C., AND LAFFERTY, J. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information and System Security 22*, 2 (2004), 179–214. 3.3.1, 5.1

[76] ZHANG, L., TIWANA, B., QIAN, Z., WANG, Z., DICK, R. P., MAO, Z. M., AND YANG, L. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the Eighth International Conference on Hardware/Software Codesign and System Synthesis* (2010), ACM, pp. 105–114. 6.5

[77] ZIEBART, B. D., MAAS, A. L., DEY, A. K., AND BAGNELL, J. A. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *UbiComp '08: Proceedings of the Tenth International Conference on Ubiquitous Computing* (2008), ACM, pp. 322–331. 1, 3.1, 3.6