A NONPARAMETRIC BAYESIAN FRAMEWORK FOR MOBILE DEVICE SECURITY AND LOCATION BASED SERVICES

A Dissertation
Presented to
the Faculty of the Electrical and Computer Engineering Department
University of Houston

in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in Electrical Engineering

by
Nam Tuan Nguyen

December 2013

A NONPARAMETRIC BAYESIAN FRAMEWORK FOR MOBILE DEVICE SECURITY AND LOCATION BASED SERVICES

Nam Tuan Nguyen

Approved:

Chair of the Committee
Dr. Zhu Han, Associate Professor
Electrical and Computer Engineering

Committee Members:

Dr. Rong Zheng, Associate Professor
Computer Science

Dr. Haluk Ogmen, Professor
Electrical and Computer Engineering

Dr. Saurabh Prasad, Assistant Professor
Electrical and Computer Engineering

Dr. Miao Pan, Assistant Professor
Computer Science
Texas Southern University, Houston, TX

Dr. Suresh K. Khator, Associate Dean,
Cullen College of Engineering

Dr. Badrinath Roysam, Professor and Chairman,
Electrical and Computer Engineering

Acknowledgements

First, I would like to express appreciation to my advisors, Dr. Zhu Han, and my co-advisor, Dr. Rong Zheng, for their guidance, valuable advice, encouragement, and constant support during my entire course of studies at University of Houston. Their profound academic knowledges and illuminating comments about my research were very valuable and helpful, not only to my doctoral study, but also to my future research and work.

Beside my advisors, I would like to thank the rest of my dissertation committee: Professor Haluk Ogmen, Professor Saurabh Prasad, and Professor Miao Pan for their precious time and support.

My appreciation also goes to my colleagues (Lanchao Liu, Yanrun Zhang, Zhou Yuan, Yi Huang, Mohammad Esmalifalak, and Najmeh Forouzandeh Mehr), who have provided me the perfect working environment.

Finally, I would like to thank my parents, my sister, my wife, Linh Nguyen, and my daughter, An Nguyen, for their encouragement, patience, and love. They are always beside me and support me in the most difficult times.

A NONPARAMETRIC BAYESIAN FRAMEWORK FOR MOBILE DEVICE SECURITY AND LOCATION BASED SERVICES

An Abstract
of a
Dissertation
Presented to
the Faculty of the Electrical and Computer Engineering Department
University of Houston

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in Electrical Engineering

by
Nam Tuan Nguyen

December 2013

Abstract

In June 2013, it was reported that, for the first time, more than half of American adults have smartphones [1]. Smartphones are carried by the users most of the time and used to access all types of personal sensitive information, from email, Facebook to banking, files server, etc. The fact that humans are more and more attached to their phones poses both good and bad aspects, namely, security challenges and new opportunities in improving users experiences. From the bad aspect, it is of interest to find out how to protect the users from cyber attack. Whereas from the good aspect, we can improve users' experiences by providing services related to their locations.

To address the first problem, in this dissertation, we propose a security framework to detect two kinds of attacks, the Masquerade attack and the Sybil attack. Most existing literature employs supervised learning and assumes the number of devices is known. We, on the other hand, propose a non-parametric Bayesian method to detect the number of devices as well as determine which observations belong to which devices in an unsupervised passive manner. An attack can be detected by comparing the number of registered users with the number of devices found, and the malicious nodes are found based on the labels of their observations.

For the second problem, we propose a location based service (LBS) enabler framework by providing a high accuracy indoor location identification and future location prediction algorithms. LBS are applications in which, locations of users are utilized to activate a set of services which significantly improve users experiences. Examples include a micro-climate control application, which can automatically adjust room temperature given that the room is occupied. It also can be a network scheduling users' access application, where users' future whereabouts can be predicted and used for arranging files transfer to better enhance users' experiences.

In this dissertation, we mainly focus our research on the above two fields. The nonparametric Bayesian framework was used as the generative model for both the observations extracted from the wireless signal in the wireless security problem, as well the observations extracted from the features that represent a location in LBS. Beside the framework, the major contributions of the dissertation

include a missing data handling algorithm, a light-weight indoor place identification algorithm, a stopping rule to terminate the algorithm in a quickest way while maintaining a acceptable false alarm rate, and a passive approach to defend against Masquerade and Sybil attacks in wireless networks. Moreover, several mechanisms to predict users' future whereabouts such as a Dynamic Hidden Markov Model that can evolve itself over time, or a prediction model based on Deep Learning were proposed. Most of the algorithms are evaluated using experimental data and proved to obtain considerably high performances compared with other state-of-the-art approaches.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivations

Due to the broadcast nature of wireless medium and programmability of wireless devices, identity of legitimate wireless devices can be easily spoofed. For instance, an ioctl system call in the Linux kernel can modify the MAC address of a network interface card. Modifying or replacing the EPROM in a phone would allow the configuration of any ESN (Electronic Serial Number) and MIN (Mobile Identification Number) via software for cellular devices. Once the device identification is compromised, multiple attackers can masquerade as a single legitimate user, or a single attacker can assume multiple legitimate identities. In both cases, many forms of attacks can be launched including Sybil attack, masquerade attack, resource depletion, traffic injection, denial of service, etc. These attacks may result in spurious outage of communication and leakage of critical information. Therefore, it is crucial to detect the presence of identity spoofing, and/or determine the number of attackers involved.

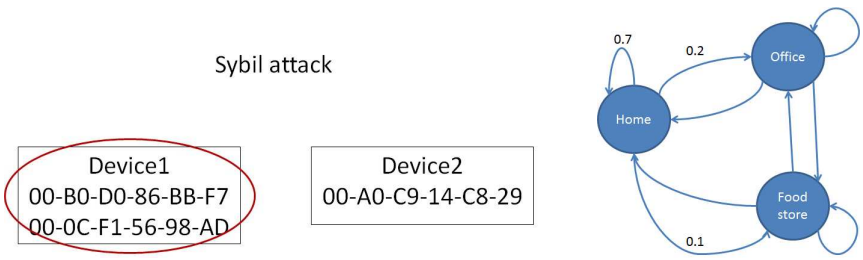
Beside wireless device security, another important perspective is users' experiences. One way to improve users' experiences is to provide convenient services based on their locations. Human come to a place with specific purposes that are normally highly correlated with the locations. For example, when they come to a shopping mall, they specifically want to make some purchases. For this reason, they would be interested in receiving coupons and promotions. Therefore, we can determine users' locations, we can develop many applications to improve the services. Fortunately, the locations can be determined with the aid from smartphones. Smartphones are readily equipped with multiple sensors such as GPS, WiFi, accelerometer, etc. These sensors provide valuable resources of the surrounding environments, which are crucial in determining the locations of the smartphones.

In this dissertation, we target both problems above. We follow the unsupervised approach, which does not require human labor and a training phase. For this reason, we utilize some unsupervised classification techniques as briefly described below.

Unsupervised classification techniques are always desired since it does not require a training phase and human labor. As can be inferred from the dissertation title, nonparametric classification techniques are the major techniques used in the dissertation. Nonparametric term has two different interpretations. First, it can be interpreted that no assumption is made about the distribution of the observations. Second, it also can be understood that there is no assumption about the number of classes. In this dissertation, the nonparametric term bears the second interpretation. With that meaning, it tries to resolve the problem or model selection, i.e., choose the model with a number of classes that best fits the observation set. Specifically, it is a classification techniques which does not require a prior knowledge about the number of classes. This important feature brings the unsupervised nature to the approach.

One of the nonparametric classification techniques is the Nonparametric Bayesian Classification (NBC). Classification based on Bayesian inference is an important topic of machine learning. Based on the observations, we can infer about the hidden information that we are interested. The hidden information can be the structure of the observations, the hidden processes created the observations or which class the observation belongs to. Classical methods normally assume the number of classes is known, leading to difficulties in model selection. The methods can be easily under fitting or over fitting. NBC, on the other hand, do not assume any prior knowledge on the number of classes or hidden processes. They, instead, assume the number of classes is infinite and only a limited number of classes are observed. As more data is observed, the number of classes can vary. This advanced property overcomes the difficulties in the model selection task in classical approaches. In this dissertation, one noticeable approach, the Infinite Gaussian Mixture Model (IGMM), is studied. Besides, a more relaxed nonparametric classification approach, the mean shift classification, will also be investigated. MS is useful when the classes are well separable while IGMM performs exceptionally well compared to MS when the classes are overlapping.

In a wireless security setting, nonparametric classification helps to detect an attack since it can determine the number of devices as well as the device that transmitted the observation. The framework is briefly described in Section 1.2. In a wireless device localization problem, it helps to



(a) Sybil attack (b) Markov chain

Figure 1.1 Two possible applications.

identify places based on the settings around the devices, as introduced in Section 1.3. All of these tasks can be finished without any users' interactions or war driving. For this important role, we will first give a brief introduction about some nonparametric classification approaches below. It is important to note that we improve many aspects of the approaches, as will be detailed out later.

1.2 Mobile Device Security

In this dissertation, a typical Sybil attack, namely the Primary user emulation (PUE) attack in cognitive radio will be investigated. An illustration of a Sybil attack is given on Figure 1.1(a). As we can see, in Sybil attack, two different devices with different MAC addresses claim the same identity. In this case, a Sybil attack is detected. PUE attacks, where attackers mimic the signals of primary users (PUs), can cause significant performance degradation in Cognitive Radio (CR) systems. Detection of the presence of PUE attackers is thus an important problem. In this chapter, using device-specific features, we propose a passive, nonparametric classification method to determine the number of transmitting devices in the PU spectrum. Our method, called DECLOAK, is passive since the sensing device listens and captures signals without injecting any signal to the wireless environment. It is nonparametric because the number of active devices needs not to be known as priori. Channel independent features are selected forming fingerprints for devices, which cannot be altered postproduction. The infinite Gaussian mixture model (IGMM) is adopted and a modified collapsed Gibbs sampling method is proposed to classify the extracted fingerprints. Due to its unsupervised nature, there is no need to collect legitimate PU fingerprints. In combination with received power and device MAC address, we show through simulation studies that the proposed method can

efficiently detect the PUE attack. The performance of DECLOAK is also shown to be superior than that of the classical non-parametric Mean Shift (MS) based classification method.

1.2.1 Contributions to Wireless Device Security

The main contributions of the security framework is as follow:

- *An unsupervised approach:* We are the first to propose an unsupervised approach. Previously, any security scheme would require a training phase. In other words, devices are required to pre-registered and fingerprints are recorded. This is not feasible in practice. DECLOAK overcomes this shortcoming, using no training phase.
- *Unique and hard-to-clone device fingerprint:* We propose some hardware dependent and channel independent features. Thus, the fingerprint built upon the features are hard-to-clone. Except that the attacker can produce exactly the same hardware, the fingerprint cannot be duplicated.
- *Upper bound and lower bound of classification performance:* We derived and calculated the performance bounds, demonstrating that DECLOAK performance is approaching the upper bound.

1.3 Location Based Services

There are two main factors to enable the LBS. First, we need to determine the locations of the users. Based on the locations, we can provide services that can enhance users' experiences. Examples include sending promotions to users in a shopping mall, reminding users about some tasks they need to do before leaving/coming from/to a place, micro-climate control, etc. Second, if we can predict users' future whereabouts, we can schedule the service accordingly. For example, users' locations are highly correlated to network signal profiles. Thus, knowing future locations is equivalent to awareness of future network conditions. Therefore, for delay tolerant tasks, we can schedule network access when the network is in its best condition, avoiding delay and network overloading. In this dissertation, we propose a complete framework that is capable of handling both aspects.

The first component is an indoor location identification algorithm, namely LOIRE. LOIRE is a nonparametric, profiling-free, yet lightweight and energy-efficient inference framework, to identify recurring and new places mobile users visit indoor. Combining WiFi scans and accelerometer readings, LOIRE can decide a new place and an old place with 100% and 92.2% accuracy, respectively, using sub-second to seconds of radio signal strength (RSS) samples (assuming a sampling rate of 10Hz). LOIRE is shown to be robust to signal variations, missing data and device heterogeneity through experimental evaluations on mobile phones of different brands/models.

The second component include several prediction approaches and applications based on the predicted locations. The first prediction approach utilizes the idea of a Markov chain, as illustrated on Figure 1.1(b). We propose a Dynamic Hidden Markov Model (DHMM) to model the time dependent and location dependent network conditions observed by individual users. The model is dynamic since transition matrix and states are updated when new observations are available. On the other hand, it has all the properties of a Hidden Markov Model. DHMM can predict precisely the next state given the current state, hence can provide a good prediction of network condition. DHMM has two layers, the top layer is Received Signal Strength (RSS) and the bottom layer consists of states, defined as a mixture of location, time and the signal strength itself. Since the state is defined as a mixture, it is hidden and the number of states is also not known a priori. Thus, the Nonparametric Bayesian Classification is applied to determine the hidden states. We show through simulations that when combined with a Markov decision process, the opportunistic scheduling can reduce transmission costs up to 50.34% compared with a naive approach.

The Markov model is powerful, but it poses some problems such as expressing different patterns over different times or exploiting embedded frequencies in human behavioral patterns [2]. In the second prediction approach, we utilize the idea of Deep Learning in finding typical users' movement patterns. Similar to Principle Component Analysis (PCA), Deep Learning can extract principle components, but with a much more precise results. We treat the users trajectories as a 2-D image, and extract the typical moving patterns of the users. The Deep Learning network learns a multiple layers of representations of the users movements with the lowest layer represents the

movements and the highest layer represents the patterns. It also learns the weights connecting the layers. Based on the network, one can infer about the future locations of the users.

1.3.1 Contributions to the LBS

The main contributions to the LBS are as follow:

- *Incomplete data handling:* For indoor localization, we utilize WiFi Received Signal Streng (RSS) as the signatures for places. Our first contribution is to systematically handle incomplete data, which allows grouping places with similar RSS readings despite Missing at Random (MAR) and Missing not at Random (MNAR).
- *Online classification technique:* In an effort to reduce the power consumption of WiFi scans, our second contribution is to devise an online classification mechanism that accumulates *just enough* measurements and stops the WiFi interface from scanning until the user moves again.
- *Profiling free:* It is a non-parametric approach and assumes no knowledge of the number of places a user visits. It requires no manual on-site survey.
- *Using experimental data:* Multi-week long traces collected from multiple smart phones from two buildings on campus were used. Using this data set, our indoor place identification algorithm can accurately identify a revisited place and a new place 92.2% and 100% of the time.
- *Dynamic Hidden Markov Model:* Propose a new hidden Markov model, which evolves itself over time and automatically profile new states.
- *Using Deep Learning to predict future locations:* We are the first to propose the Deep Learning in learning users' moving patterns, significantly improve the performance.

1.4 Dissertation Organization

The rest of the dissertation is organized as follow. We first study about the modern classification techniques in Chapter 2. The mobile device security is covered in Chapter 3. For the rest of the dissertation, we discuss about the complete framework for LBS. We first describe an indoor place identification technique in Chapter 4. And then, to complete the LBS framework, we introduce two mechanisms in predicting future users' whereabout in Chapter 5 and Chapter 6. Finally, a

conclusion and possible future works are described in Chapter 7.

Chapter 2

Nonparametric Classification Approaches

The term "nonparametric" means the number of classes is not given as prior. Classic approaches like K-means, Support Vector Machine, finite mixture model, etc., always assume a prior knowledge about the number of classes. However, in an unsupervised learning, the number is not available in any situation. For this reason, nonparametric classification approaches are desirable with unsupervised learning. In this chapter, we will introduce two major nonparametric approaches. The first approach involves Bayesian graphical model, hence, called nonparametric Bayesian classification (NBC). NBC has two major parts, the generative model is described in Section 2.1 and the inference model is described in Section 2.2. Performance bounds and the classification evaluation criterion will be investigated in Section 2.3.

2.1 Nonparametric Bayesian Generative Model

Two different approaches will be discussed in the next two sections. The first approach is related to data generation and will be discussed below. In other words, given a model, and parameters of the model, we want to generate random samples, which are very similar to the observations. If we can generate a data set which is similar to the observations set in the 4-D feature space, then we can reasonably assume that the data set follows the specific generative model. Under the assumption, we will next describe an inference algorithm in the next section. In the inference algorithm, given the observations in the 4-D feature space, the goal is to estimate the associated parameters.

Depending on our prior knowledge about the data, two classes of generative models can be adopted. The Finite Mixture Model (FMM) is used in the case where the number of classes is known a priori, while the Infinite Mixture Model (IMM) is used in the situation where the number of classes is unknown or may vary over time. Note that although the IMM is adopted in our classification approach, the IMM is built upon the FMM, and thus the understanding of the later is essential. When each class is Gaussian distributed, FMM becomes the Finite Gaussian Mixture Model (FGMM) and

IMM becomes Infinite Gaussian Mixture Model (IGMM). Before delving into the details of the models, we first define the Dirichlet Distribution [3], a core distribution in both models.

2.1.1 Dirichlet Distribution

The Dirichlet distribution is the multivariate generalization of the beta distribution, and the conjugate prior of the categorical distribution and multinomial distribution in Bayesian statistics [4]. In other words, its probability density function represents the belief that the probabilities of K rival events $(w_1, \dots w_K)$, given that event w_i has been observed $\alpha_i - 1$ times, $i = 1, \dots, K$.

Definition 2.1. *The Dirichlet distribution of order $K \geq 2$ with parameters $\alpha_1, \dots, \alpha_K > 0$ has a probability density function with respect to the Lebesgue measure on the Euclidean space $\mathbb{R}^{K-1}$ given by*

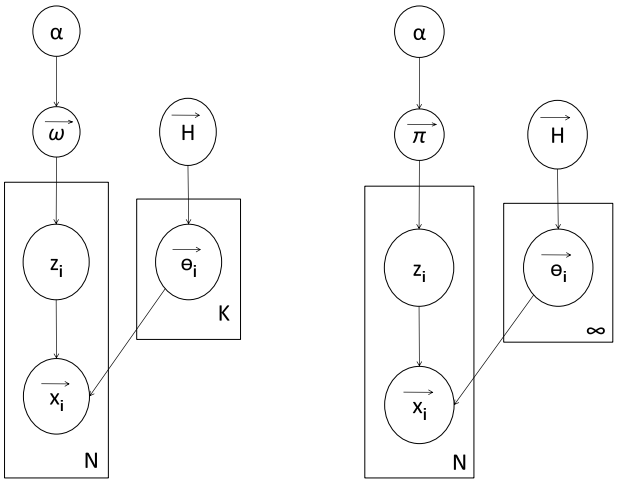
$$Dir(\alpha_1, \dots, \alpha_K) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} w_i^{\alpha_1 - 1} \tag{2.1}$$

for all $w_1, \dots, w_K > 0$ satisfying $w_1 + \cdots + w_K = 1$. The density is zero outside this open $(K-1)$-dimensional simplex. The normalizing constant is written in terms of the multinomial beta function, which can be expressed in terms of the gamma function:

$$B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)}. \tag{2.2}$$

2.1.2 Finite Gaussian Mixture Model

We will describe FGMM first and IGMM can be obtained by extending FGMM. Recall matrix $\vec{X} = [\vec{x}_1, \vec{x}_2, ..., \vec{x}_N]$ is the data set of N data points, in which each $\vec{x}_i$ is a vector of D dimensions. In Figure 2.1(a), $\vec{w}$ is the mixing weight vector, $[w_1, w_2, ..., w_K]$, and represents the probability of assigning one data point to one of the classes. K is the number of classes. Each class has its own distribution and $\vec{\theta}_k$ is the vector of parameters for that distribution. Because we assume the distribution of the observations is Gaussian, each $\vec{\theta}_k$ consists of mean, $\vec{\mu}_k$ and covariance matrix, Σ_k. $\vec{H}$ are the hyper-parameters, which are parameters of the distribution of $\vec{\theta}_k$'s. The hyperparameters represent our knowledge of the observations. In Figure 2.1(a), the rectangular shape denotes the



(a) Finite (b) Infinite

Figure 2.1 Gaussian Mixture Models.

repeated sub-structures. Each indicator, z_i, is associated with a data point $\vec{x}_i$ indicating which class it belongs to. Specifically, $z_i = k$ indicates that $\vec{x}_i$ belongs to class k with probability $p(z_i = k) = w_k$. The FGMM can be defined as below [4].

Definition 2.2. *The Finite Gaussian Mixture Model is defined as*

$$w|\alpha \sim Dir(\vec{M});\ z_i|w \sim \textit{Multinomial}(\cdot|\vec{w});\ \vec{\theta}_k \sim \vec{H}; \vec{x}_i|\vec{\theta}_k \sim \textit{Gaussian}(\cdot|\vec{\theta}_k), \tag{2.3}$$

where $\vec{M} = [\alpha/K, \alpha/K, ..., \alpha/K]$ *and K is finite.*

FGMM can be easily interpreted as follow. We start with a $Dir(\vec{M})$ distribution to create a finite number of weights $\vec{w}$. Given the weights, a Multinomial distribution is used to generate the labels for each observations. If the labels are known, we know which class each observation belongs to, hence we can determine the distribution that each observation follow, i.e., $\vec{x}_i|\vec{\theta}_k \sim \mathrm{G}(\cdot|\vec{\theta}_k)$. Now, we want a very flexible model, thus, we do not want a fixed prior for the parameters $\vec{\theta}$. Hence, $\vec{\theta}$ is further assumed to follow a based distribution $\vec{H}$, which gives a wider range of flexibility for the model. The same analysis can be also applied for IGMM mentioned in the next section.

The most difficult task in the FGMM is model selection, i.e., how to determine the number of classes, K. In our problem, if we know exactly the number of wireless devices then the number of classes in the feature space is known. Unfortunately, in most situations, we have no control of

the number of active (legitimate or illegitimate) devices. Therefore, the FGMM cannot model the feature space, and a more advanced statistical method needs to be adopted as discussed next.

2.1.3 Infinite Gaussian Mixture Model

IGMM can resolve the problem. It is an extension of FGMM by letting $K \to \infty$. It assumes that the number of classes is infinite but only a finite number of classes can be observed at a certain time. By doing that, IGMM virtually can fit any situation, not matter how big the observation set is and how variation over the time the number of classes is. Therefore, IGMM is a very powerful tool to model the generative procedure for many data set.

Figure 2.1(b) shows a graphical representation of the IGMM. The IGMM is identical to the FGMM except that the number of classes is infinity. This difference leads to variations between the two models as can be seen in the definition below. With similar notations as in the FGMM, the IGMM is defined as:

Definition 2.3. *Infinite Gaussian Mixture Model.*

$$\vec{w}|\alpha \sim \textit{Stick}(\alpha); z_i|\vec{w} \sim \textit{Multinomial}(\cdot|\vec{w}); \vec{\theta}_k \sim \vec{H}; \vec{x}_i|(z_i = k, \Sigma_k, \vec{\mu}_k) \sim G(\cdot|\vec{\mu}_k, \Sigma_k), \quad (2.4)$$

where $\vec{\theta}_j \sim \vec{H}$ *stands for:*

$$\Sigma_k \sim \textit{ Inverse Wishart}_{v0}(\Lambda_0); \vec{\mu}_k \sim G(\vec{\mu}_0, \Sigma_k/K_0), \quad (2.5)$$

and $\vec{w}|\alpha \sim \textit{Stick}(\alpha)$ *is a shorthand of:*

$$w'_k|\alpha \sim Beta(1, \alpha); w_k = w'_k \prod_{l=1}^{k-1}(1 - w'_j); k \to \infty. \quad (2.6)$$

The Inverse Wishart distribution in (2.5) is chosen [5], [6] because it is the conjugate prior for the normal distribution, which is instrumental in deriving a closed form solution for the posterior distribution of the indicators, $\vec{Z} = [z_1, \ldots, z_N]$. For your reference, the pdf of the Inverse Wishart distribution is given as follow:

$$p(\Sigma) = \frac{|\Lambda^{-1}|^{v/2}|\Sigma|^{-(v+D+1)/2}e^{-trace(\Lambda^{-1}\Sigma^{-1})/2}}{2^{vD/2}\Gamma_D(v/2)}, \quad (2.7)$$

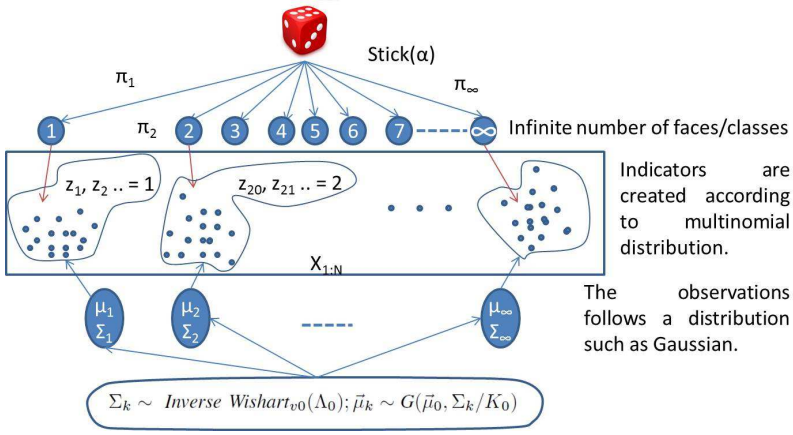


Figure 2.2 IGMM - Another view.

where Λ^{-1} is the scale matrix, D is the number of dimensions and v is the degree of freedom.

In the FGMM, the distribution of the weights is modeled as the Dirichlet distribution. However, when the number of classes goes to infinity, it is hard to sample w directly from the Dirichlet distribution. Instead, another process, called *the stick breaking construction* [4] is used and defined as follow. We know that $\sum_{i=1}^{K} w_k = 1$, hence, we start with a stick of length 1 and break it into two parts at w_1', which is sampled according to the Beta distribution in (2.6). Assign weight w_1 equal to the length of either one of the two parts in (2.6), and repeat the same process on the other part. The same procedure is repeated until a sufficient number of weights are obtained. Another more intuitive representation of IGMM can be found on Figure 2.2. Imagine that there is a dice with infinite number of faces, each face is corresponding to a class. In the generative model, each time the dice is rolled, a class is chosen with a certain weight. Each face has its own set of parameters, i.e. μ and $\mathbf{\Sigma}$. Given the parameters, the set of observations corresponding to the face/class will be generated.

2.2 Inference in IGMM

In this section, we describe the nonparametric Bayesian classification method. This is also referred to as the inference model. As mentioned before, now we are given N obseravtions, $\vec{X} = [\vec{x}_1, \vec{x}_2, ..., \vec{x}_N]$ from the above D-dimensional feature space. And we have to infer back the parameters, which is the purpose of the inference algorithm. Specifically, we have to solve the

following main questions: i) *how many devices have generated the data*, and ii) *which device each data point belongs to*. The parameter we need to infer here is the indicator vector $\vec{Z}$. Given $\vec{Z}$, we can answer the above questions.

Although IGMM is very well known, there can be various ways of applying it to solve for a specific problem. For example, in the classification problem, there are many approaches trying to solve the problem. In this chapter, we try to derive a closed form solution for the problem by integrating out the parameters $\vec{\mu}_k, \Sigma_k$, thus, developing a collapsed Gibbs Sampling approach. Other methods can keep the parameters in the sampling space. However, in this method, by integrating out the parameters, the sampling space is much reduced, leading to a far less complicated classification algorithm. Below, steps needed to implement the collapsed Gibbs Sampling method are described and finally, the collapsed Gibbs Sampling method will be explained in details.

Our goal is to find out the indicators, z_i, $i = 1, \ldots, N$, for each data point. For that purpose, given the priors, we need to derive an expression for the distribution of $\vec{Z}$. Then, a collapsed Gibbs sampler will be used to sample the distribution and determine the class label. Gibbs sampling [7] is a method to generate samples of two or more random variables from a joint distribution. Sampling from a univariate distribution is easy to implement. However, when there are multiple variables, it is very complicated to derive a closed form solution for the posterior due to the integrations needed to carry out. Moreover, direct sampling from a complex joint distribution is very difficult. The Gibbs sampler is an efficient method for the situation by assuming all the other variables are known except the current one. It is then much easier to retrieve a closed form for the univariate posterior distribution of the current variable. Below, derivations needed to carry out the Gibbs sampler will be described.

Recall that we need to find out the indicators, $\vec{Z}$, which is very complicated and in many situation, a closed form solution cannot be derived due to the required integration calculations. Now, instead of finding the joint distribution of $\vec{Z}$, we only need to find $P(z_i = k|\vec{Z}_{-i}, \alpha, \vec{\theta}, \vec{H}, \vec{X})$, which is much simpler. Here, z_i is the unknown variable while $\vec{Z}_{-i}$ is a vector all the other indicators. Apply the Bayesian rule, the distribution is:

$$P(z_i = k|\vec{Z}_{-i}, \alpha, \vec{\theta}, \vec{H}, \vec{X}) = P(z_i = k|\vec{Z}_{-i}, \alpha, \vec{\theta}_k, \vec{H}, \vec{x}_i)$$

$$\sim P(\vec{x}_i|z_i = k, \vec{Z}_{-i}, \alpha, \vec{\theta}_k, \vec{H})P(z_i = k|\vec{Z}_{-i}, \alpha) \tag{2.8}$$

$$\sim P(\vec{x}_i|\vec{\theta}_k)P(z_i = k|\vec{Z}_{-i}, \alpha), \tag{2.9}$$

where the "$\sim$" symbol hides the normalized factor in the Bayesian rule.

In (2.8), $P(\vec{x}_i|\vec{\theta}_k)$ is the likelihood and simply a Gaussian distribution. The only unknown term is $P(z_i = k|\vec{Z}_{-i}, \alpha)$, which will be determined in two steps. First, starting with a finite K, we apply the FGMM. Second, we explore the limit when K$\rightarrow \infty$ and apply IGMM.

2.2.1 Applying FGMM to Solve the Classification Problem

In this subsection, a closed form for $P(z_i = k|\vec{Z}_{-i}, \alpha)$ will be derived by assuming the number of classes is K. Hence, the feature space is modeled as the FGMM with $\vec{M}$ and $P(w|\alpha, \vec{M})$ described as in (2.3). To adopt the Gibbs sampling method, assume that we have already classified N data points, and just received a new one, $\vec{x}_{N+1}$. The probability of an assigning the new observation to the k^{th} cluster is:

$$P(z_{N+1} = k|z_{1:N}; \alpha, \vec{M}) = \int P(z_{N+1} = k|\vec{w})P(\vec{w}|z_{1:N}; \alpha, \vec{M})d\vec{w} \tag{2.10}$$

$$= \int P(z_{N+1} = k|\vec{w})P(\vec{w}; \alpha^*, \vec{M}^*)d\vec{w} \tag{2.11}$$

$$= E(P(z_{N+1} = k|\vec{w})) = E(w_k) = \frac{\alpha^* m_k^*}{\sum_{i=1}^{K} \alpha^* m_i^*} = m_k^*. \tag{2.12}$$

(2.10) is simply a marginal distribution where $\vec{w}$ is margined out. Result (2.12) is due to the definition of the expected value. In (2.12), $P(z_{N+1} = k|\vec{w})$ is actually w_k. As a result, we get the marginal probability of assigning the new data point to a represented class as in (2.12). m_k^* belongs to $\vec{M}^*$ and $\vec{M}^* = [m_1^*, ..., m_k^*, ..., m_K^*]$. $\vec{M}^*$ and α^* are updated prior parameters of the Dirichlet distribution after observing $z_{1:N}$ data points [8]. To find out the updated prior parameters, applying

the Bayesian rule, we have:

$$P(\vec{w}|z_{1:N};\alpha,\vec{M}) = P(z_{1:N}|\alpha,\vec{M},\vec{w})P(\vec{w}|\alpha,\vec{M}) \tag{2.13}$$

$$= A\prod_{k=1}^{K}\vec{w}_k^{n_k}x\prod_{i=1}^{K}\vec{w}_k^{\alpha m_k-1} \tag{2.14}$$

$$= A\prod_{k=1}^{K}\vec{w}_k^{\alpha m_k+n_k-1} \tag{2.15}$$

$$= Dir(\vec{w};\alpha^*,\vec{M}^*) \tag{2.16}$$

where $m_k = \alpha/K$, n_k is the number of data points in the same cluster, and A is a normalizing constant. m_k represents the number of data points that belong to class k^{th} initially. In (2.13), the distribution of $z_{1:N}$ is just a Multinomial distribution, and $\vec{w}$ follows the Dirichlet distribution. Since the Dirichlet distribution is the conjugate prior for the Multinomial distribution, the posterior distribution of the weights is the Dirichlet distribution with the updated prior parameters, α^* and $\vec{M}^*$ as in (2.16). According to [9], the updated prior parameters can be calculated as:

$$\alpha^* = \alpha + N \text{ and } \vec{M}^* = \frac{\alpha\vec{M} + N\hat{F}}{\alpha + N}, \tag{2.17}$$

where $\hat{F}$ is the empirical distribution. Since $\vec{M}$ and $\vec{M}*$ are sets of $\{m_k\}$ and $\{m_k^*\}$ respectively, one can easily replace m_k into (2.17) to get its updated value:

$$m_k^* = \frac{\alpha m_k + \sum_{i=1}^{N}\delta(z_i = k)}{\alpha + N}. \tag{2.18}$$

From (2.16) and (2.18), we have:

$$P(z_{N+1} = k|z_{1:N},\alpha,\vec{M}) = \frac{\alpha m_k + \sum_{i=1}^{N}\delta(z_i = k)}{\alpha + N} = \frac{\alpha/K + n_k}{\alpha + N}, \tag{2.19}$$

where n_k is the number of data points coming from the same k^{th} cluster, and data point $N + 1$ is not included.

2.2.2 Extending the FGMM to the IGMM

Now, we will explore the mixture by letting $K \to \infty$. When K approaches infinity, the FGMM becomes the IGMM. Equation (2.19) is equivalent to:

$$P(z_{N+1} = k | z_{1:N}, \alpha) = \frac{\alpha m_k + \sum_{i=1}^{N} \delta(z_i = k)}{\alpha + N} \approx \frac{n_k}{\alpha + N}. \tag{2.20}$$

The value on the right hand side of (2.20) is the probability that the $(N+1)^{th}$ data point belongs to the k^{th} cluster. One important property of the NPB is the exchangeability [10]. Applying this property, we can swap $N+1$ with any value of i without changing the joint probability. Hence, we have:

$$P(z_i = k | \vec{Z}_{-i}, \alpha) \quad = \quad \frac{n_{k,-i}}{\alpha + N - 1}, \tag{2.21}$$

where $n_{k,-i}$ is the number of observations assigned to the k^{th} cluster, excluding the i^{th} observation. The left hand side of (2.21) is the probability of assigning a data point to an existing device. The probability of assigning that data point to a new device is given by,

$$P(z_i \neq z_j, \forall j \neq i | z_{-i}, \alpha) = 1 - \frac{\sum_{j=1}^{K} n_{j,-i}}{\alpha + N - 1} = \frac{\alpha}{\alpha + N - 1}. \tag{2.22}$$

Clearly, this probability is equal to one minus sum of all probability of assigning to K existing classes. The summation in (2.22) is equal to $N-1$ because it counts all the data except for the current one. This is also called the Chinese Restaurant Process (CRP) [4]. In CRP, the number of tables (classes), is infinite and each table can serve an infinite number of customers (data points). The first customer coming in seats at the first table. The second one will seat either at the first table or a new one. The probability of seating at an occupied table is proportional to the number of customers already seated there in (2.21). And the probability of seating in a new table is proportional to α in (2.22).

From (2.22) and (2.21), one can design a Gibbs sampler to obtain samples of Z. Nevertheless, we see that there is no observed data in the above equations. That is because they are only priors. After observing $N-1$ data points, we can update our priors by applying the Bayesian rule. Using

(2.22) and (2.21) as priors in (2.8), we can update our indicators posterior knowledge after observing Z_{-i} data points. The probability of assigning a data point to a represented class is:

$$P(z_i = k|\vec{Z}_{-i}, \alpha, \vec{\theta}, \vec{H}, \vec{X}) = P(z_i = k|\vec{Z}_{-i}, \alpha)P(\vec{x}_i|\vec{\theta}_k) = \frac{n_{k,-i}}{\alpha + N - 1} Gaussian(\vec{\theta}_k). \quad (2.23)$$

The probability of assignment to unrepresented classes is,

$$\begin{aligned} P(z_i \neq j, \forall j \neq i|\vec{Z}_{-i}, \alpha, \vec{\theta}, \vec{H}, \vec{X}) &= P(z_i \neq j, \forall j \neq i|\vec{Z}_{-i}, \alpha)P(\vec{x}_i; \vec{H}) \\ &= \frac{\alpha}{\alpha + N - 1} \int_{\vec{\theta}} P(\vec{x}_i|\vec{\theta})P(\vec{\theta}|\vec{H})d\vec{\theta}. \end{aligned} \quad (2.24)$$

The integration is the likelihood of being assigned to any unrepresented classes. It is also the marginal probability since $\vec{\theta}$ is integrated out. The integration is analytically tractable, because we have chosen $P(\vec{\theta}|H) \sim$ Inverse Wishart$_{v0}(\Lambda_0)$ that is the conjugate prior of the Normal distribution. The above two equations are necessary and sufficient conditions to implement the Gibbs Sampling algorithm. However, in (2.23) we see that in order to sample the values of Z, we need to sample the values of the parameters in advance. If, somehow, we can integrate out the parameters, the sample space will be much reduced and the algorithm will be faster. Next, we adopt the Collapsed Gibbs Sampling Method [11] to improve the computation efficiency.

2.2.3 Collapsed Gibbs Sampling Method

In this subsection, the parameters $\vec{\theta}_k$ will be integrated out for the purpose of reducing the sampling space. At the end of this section, we obtain a closed form solution for the posterior distribution. The distribution does not depend on $\vec{\theta}_k$ and only depends on the hyperparameters. From Figure 2.1(b), applying the Bayesian rule, we have the joint distribution:

$$P(\vec{Z}, \vec{\theta}|\vec{X}; \alpha, \vec{H}) = P(\vec{\theta}|\vec{H})P(\vec{Z}|\vec{X}, \alpha) \sim P(\vec{\theta}|\vec{H})P(\vec{X}|\vec{Z}, \vec{\theta}; \vec{H})P(\vec{Z}|\alpha).$$

Now, consider the case when $z_i = k$, the joint distribution becomes

$$P(z_i = k, \vec{\theta}_k|\vec{X}; \alpha, \vec{H}) = P(\vec{\theta}_k; \vec{H})P(\vec{X}_k|\vec{\theta}_k; \vec{H})P(z_i = k; \alpha), \quad (2.25)$$

where $\vec{X}_k$ is a set of all data points belonging to the k^{th} cluster. The marginal joint distribution after integrating out the parameters is:

$$P(z_i = k|\vec{X};\alpha,\vec{H}) = \int P(z_i = k,\vec{\theta}_k|\vec{X};\alpha,\vec{H})d\vec{\theta}_k \tag{2.26}$$

$$= P(z_i = k;\alpha) \times \int P(\vec{X}_k|\vec{\theta}_k;\vec{H})P(\vec{\theta}_k|\vec{H})d\vec{\theta}_k \tag{2.27}$$

$$= P(z_i = k;\alpha) \times E\{P(\vec{X}_k|\vec{\theta}_k;\vec{H})\} \tag{2.28}$$

$$= P(z_i = k;\alpha) \times P(\vec{X}_k;\vec{H}). \tag{2.29}$$

Replace (2.25) into (2.26) to get (2.27), and (2.28) is derived by applying the definition of expectation value in (2.27). Given the $P(z_i = k|\vec{X};\alpha,\vec{H})$, we are now able to calculate

$$P(z_i = k|\vec{Z}_{-i},\vec{X};\alpha,\vec{H}) = P(z_i = k|\vec{Z}_{-i};\alpha)P(\vec{x}_i|\vec{X}_{k,-i};\vec{H}). \tag{2.30}$$

The first term in the above equation is given in (2.20). For the second term, because we choose the Inverse Wishart as the prior distribution for Σ_k and Gaussian distribution for $\vec{\mu}_k$, according to [6], we will have the multivariate Student-t distribution for $P(\vec{x}_i|\vec{X}_{k,-i};\vec{H})$:

$$P(\vec{x}_i|\vec{X}_{k,-i};\vec{H}) \sim t_{\upsilon_n - D+1}\left[\vec{\mu}_n, \frac{\Lambda_n(\kappa_n+1)}{\kappa_n(\upsilon_n - D+1)}\right], \tag{2.31}$$

where

$$\vec{\mu}_n = \frac{\kappa_0}{\kappa_0+N}\vec{\mu}_0 + \frac{N}{\kappa_0+N}\bar{X}, \tag{2.32}$$

$$\kappa_n = \kappa_0 + N,\ \nu_n = \nu_0 + N, \tag{2.33}$$

$$\Lambda_n = \Lambda_0 + S + \frac{\kappa_0 n}{\kappa_0+N}(\bar{X}-\vec{\mu}_0)(\bar{X}-\vec{\mu}_0)^T, \tag{2.34}$$

$$\bar{X} = (\vec{x}_1 + \vec{x}_2 + \cdots + \vec{x}_N)/N, \tag{2.35}$$

and D is the data dimension, $\vec{\mu}_l, \kappa_l, \nu_l$ and Λ_l are updated hyperparameters after observing $L-1$ samples. L is the number of observations in that particular cluster. To sample $\vec{Z}$, we do not have to sample the parameters. Instead, we can implement the Collapsed Gibbs Sampling. For the case of assignment to an unrepresented cluster, since in (2.24), the parameters are already integrated out, we only need to find $P(\vec{x}_i;\vec{H})$. Based on the expression of $P(\vec{x}_i|\vec{X}_{k,-i};\vec{H})$ given in (2.31),

Algorithm 2.1 Collapsed Gibbs Sampler

input : A feature space with N data points with unknown number of classes
output: A clustered feature space with K, the number of classes

Initialize: Start by seating the first customer at the first table, the next customer comes in will be seated either at an occupied table or seat at a new table. The new table will be indexed as one plus the total number of tables. Start with $K = 1$ **for** $j \leftarrow 1$ **to** M, *the number of iterations* **do**

  for $i \leftarrow 1$ **to** N **do**

    O = # of points having the same indicator $z_i^{(j)}$ **if** *O = 1* **then**

      $K = K - 1$ **for** $l \leftarrow i + 1$ **to** N **do**

        $z_l^{(j-1)} = z_l^{(j-1)} - 1$

    Sample $z_i^{(j)}$ according to (2.37) and (2.38), given $z_1^j \dots z_{i-1}^j, z_{i+1}^{(j-1)} \dots z_N^{(j-1)}$ **if** $z_i^{(j)} > K$ **then**

      Update $K = K + 1$;

the distribution can be determined as a multivariate Student-t distribution with the hyperparameters before updating, $\vec{\mu}_0, \kappa_0, \nu_0$ and Λ_0:

$$P(x_i; \vec{H}) \sim t_{v_0 - D + 1} \left\{ \vec{\mu}_0, \frac{\Lambda_0(\kappa_0 + 1)}{\kappa_0(\upsilon_0 - D + 1)} \right\}. \tag{2.36}$$

In conclusion, we have obtained two posterior distributions for the indicators. The first distribution is used in the case of assigning the data point to an existing cluster. From (2.21), (2.31) and (2.30), we have the form of the distribution:

$$P(z_i = k | Z_{-i}, X; \alpha, \vec{H}) \sim \frac{n_{k,-i}}{\alpha + N - 1} t_{v_n - D + 1} \left[\vec{\mu}_n, \frac{\Lambda_n(\kappa_n + 1)}{\kappa_n(\upsilon_n - D + 1)} \right]. \tag{2.37}$$

The second distribution is used in the case of assignment to a new device, which does not contribute any data point in the feature space. From (2.22), (2.36) and (2.30), we have the distribution:

$$P(z_i \neq j, \forall j \neq i | Z_{-i}, X; \alpha, \vec{H}) \sim \frac{\alpha}{\alpha + N - 1} t_{v_0 - D + 1} \left[\vec{\mu}_0, \frac{\Lambda_0(\kappa_0 + 1)}{\kappa_0(\upsilon_0 - D + 1)} \right]. \tag{2.38}$$

It is important to mention that the above probabilities sum up to 1 when k runs from 1 to the number of classes (including the possible unrepresented one).

The posterior we have built so far is essential to implement a Gibbs sampler to sample the values for the indicators. Algorithm 2.1 summarizes the Collapsed Gibbs sampler algorithm to classify the data points into different classes. As we can see, as soon as a new value for the current

indicator is obtained, it will be used to update the distribution of the next indicator. We repeat that for every data point and finally the algorithm will be converged. The number of iterations used in our simulation is $M = 250$, from which, the first 50 iterations are ignored and only the last 200 iterations are taken into account.

2.3 Evaluation Criterion and Performance Bounds

2.3.1 Classic Mean Shift Based Classification Method

MS is used as a baseline algorithm to compare against IGMM. It is a classical nonparametric classification methods. The idea of MS is to treat the physical location density of data points as the probability density. The dense regions represent the means of the underlying probability densities. By locating the dense regions, we can figure out the number of classes in the feature space. Applying the kernel density estimation approach, the density function can be estimated as [12]:

$$\hat{f}_{l,K}(\vec{x}) = \frac{1}{Nl^D} \sum_{i=1}^{N} K(\vec{x} - \vec{x}_i) = \frac{c_{k,D}}{Nl^D} \sum_{i=1}^{N} k\left(\left\| \frac{\vec{x} - \vec{x}_i}{l} \right\|^2 \right), \tag{2.39}$$

where l is diameter of a cube centered at $\vec{x}$, $c_{k,D}$ is a normalization factor and $K(\cdot)$ is called a kernel function [12]. A standard kernel function of a point will be 1 if the point is inside the cube, and be 0 otherwise. The mean of the density should be located at the point where the gradient of the density function is 0. The gradient of the density function is as follow [12].

$$\nabla \hat{f}_{l,K}(\vec{x}) = \frac{2c_{k,D}}{Nl^{D+2}} \left[\sum_{i=1}^{N} g\left(\left\| \frac{\vec{x} - \vec{x}_i}{l} \right\|^2 \right) \right] \cdot \left[\frac{\sum_{i=1}^{N} \vec{x}_i g\left(\left\| \frac{\vec{x} - \vec{x}_i}{l} \right\|^2 \right)}{\sum_{i=1}^{N} g\left(\left\| \frac{\vec{x} - \vec{x}_i}{l} \right\|^2 \right)} - \vec{x} \right], \tag{2.40}$$

where $g(x) = -k'(x)$. Equation (2.40) is a product of two components. The first component is proportional to the density function at point $\vec{x}$, while the second one is called the *mean shift vector*, $\vec{m}$. $\vec{m}$ is the difference between the weighted mean by using kernel profile, g and the center of the kernel, $\vec{x}$. This mean shift vector points toward the maxima in density function. Repeat the following actions until the gradient is 0, or in other words, the mean shift is 0, we will move to the estimated mean point:

1. Calculate the mean shift vector, $\vec{m}$.

2. Update $g(x)$ according to new $\vec{x}_i$: $\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{m}$.

MS is fast only in a low dimensional space. In a high dimensional space, it faces a serious problem of searching for neighbors of a given point, so called the problem of *multidimensional range searching*. The method also has difficulties if two classes are close to each other, leading to possible multiple local maxima in the region between them. In this case, the method might fail to detect the correct number of classes as illustrated in the simulation results.

2.3.2 Kullback-Leibler Divergence

To evaluate the performance, we use a distance metric called Kullback-Leibler Divergence (KLD) to represent the hardness of the classification problem. Given two D-variate Gaussian distributions, f_0, f_1, the KLD can be specified as:

$$KLD = \frac{1}{2}\big(trace(\Sigma_2^{-1}\Sigma_1) + (\vec{\mu}_2 - \vec{\mu}_1)^T\Sigma_2^{-1}(\vec{\mu}_2 - \vec{\mu}_1) - ln(\frac{det\Sigma_1}{det\Sigma_2} - D)\big), \tag{2.41}$$

where D is the number of dimensions.

2.3.3 Evaluation Criterion

Three metrics are used to evaluate IGMM and MS:

1. Type 1: the hit rate of detecting a PUE attack, defined as the percentage of correct counts over the total number of trials.
2. Type 2: the hit rate of assigning every single feature point to its correct cluster, defined as the number of feature points assigned to their original classes over the total number of feature points.
3. Type 3: the false alarm rate of assigning a feature point to a *valid* cluster (device) other than the correct one.

Of the three metrics, Type 1 is the most important one since it characterizes the PUE attack detection performance. Note the last two metrics differ if spurious classes are generated in the classification process. Type 2 is equivalent to P_{Hit} in (2.51) and Type 3 is equivalent to P_{FA} in (2.50). In the simulation, we follow the generative models in Section 2.1 to generate sample data

with initial parameters set according to the equations in Section above. Means and covariance matrices of the classes are varied in different settings, leading to different KLD given in (2.41). The larger the KLD between two distributions, the easier it is to separate them.

When there are multiple classes, there exists bounds of the hit rate in assigning a point to its original cluster. The bounds play an important role in evaluating the performance of the methods. Below, an upper bound and a lower bound of the hit rate will be derived for the case of two classes. The analysis can be extended to handle more than two classes.

2.3.4 Upper Bound for the Hit Rate

We denote the Gaussian distributions of two classes as f_0, f_1, with means $\vec{\mu}_0, \vec{\mu}_1$ and covariance matrixes Σ_0, Σ_1. We further define H_0 as hypothesis zero, associated with f_0 and H_1 as hypothesis one, associated with f_1. Denote $\pi = \{\pi_0, \pi_1\}$ the prior probability set for hypothesis $\{H_0, H_1\}$ respectively. According to H. Kobayashi [13], the upper bound for the hit rate is:

$$P_{Hit} \leq 1 - \pi_0 \pi_1 \rho^2 \tag{2.42}$$

where the Bhattacharyya coefficient, ρ, is [14]:

$$\rho = \exp\left\{ -\frac{1}{8}(\vec{\mu}_1 - \vec{\mu}_0)^t \left(\frac{\Sigma_1 + \Sigma_0}{2} \right)^{-1} (\vec{\mu}_1 - \vec{\mu}_0) + \frac{1}{2} ln \frac{det\frac{\Sigma_1+\Sigma_0}{2}}{det\Sigma_1 det\Sigma_0} \right\}. \tag{2.43}$$

2.3.5 Lower Bound for the Hit Rate

Using Hoeffding's inequality [15] and Kullback-Liebler Divergence [16], we can calculate the bounds for such probabilities, which can be used as a criteria to evaluate the performance of classification algorithms.

The Hoeffding's inequality theorem is defined bellow.

Theorem 1. *If $U_1, U_2, \ldots, U_n$ are independent and $a \leq U_i \leq b$ for $i = 1, \ldots, n$, then*

$$P\left(\frac{1}{n} \sum_i U_i - E[U] > \epsilon \right) \leq e^{-2n\epsilon^2/(a-b)^2}. \tag{2.44}$$

In our work, we define U as: $U_i = \log \frac{f_1(x_i)}{f_0(x_i)}$, where $x_i, i = 1, \ldots, n$ are the observations. The fraction is the likelihood ratio between two distributions. If both the distributions are bounded,

$0 < c \leq f_0, f_1 \leq d < \infty$, the bounds for U can be specified as:

$$\log \frac{c}{d} \leq \log \frac{f_1(x_i)}{f_0(x_i)} \leq \log \frac{d}{c}. \tag{2.45}$$

The false alarm can be expressed mathematically as:

$$P_{FA} = P\left(\frac{1}{n}\sum_i U_i > 0 | H_0\right), \tag{2.46}$$

which means given the observations are generated from the hypothesis zero, it is decided that they are from the hypothesis one. From equation (2.46), we have:

$$P_{FA} = P\left(\frac{1}{n}\sum_i U_i - E[U|H_0] > -E[U|H_0]|H_0\right). \tag{2.47}$$

Now, we can apply the Hoeffding's inequality theorem:

$$P\left(\frac{1}{n}\sum_i U_i - E[U|H_0] > -E[U|H_0]\right) \leq e^{-2nE^2[U|H_0]/(\log\frac{d^2}{c^2})^2}. \tag{2.48}$$

The term $E[U|H_0]$ turns out to be the KLD of p_1 from p_0:

$$E[U|H_0] \quad = \quad \int f_0(x) \log \frac{f_1(x)}{f_0(x)} dx = -KLD(f_0||f_1). \tag{2.49}$$

Replace (2.49) into (2.48), we have:

$$P_{FA} \leq e^{-2nKLD^2(f_0||f_1)/(\log\frac{d^2}{c^2})^2}. \tag{2.50}$$

Given the limited feature space, the bounds of the distribution, c, d, can be easily determined. The KLD can be calculated based on the means and the variances of the distributions. Hence, the upper bound for the false alarm of assigning a point to the correct cluster can be determined. Similarly, the lower bound for the probability of hit rate can be calculated as:

$$1 - e^{-2nKLD^2(f_1||f_0)/(\log\frac{d^2}{c^2})^2} \leq P_{Hit}. \tag{2.51}$$

The bounds calculated above are based on the likelihood ratio hypotheses test. The performances of other methods might not be limited by the bound. It only provides a means to evaluate

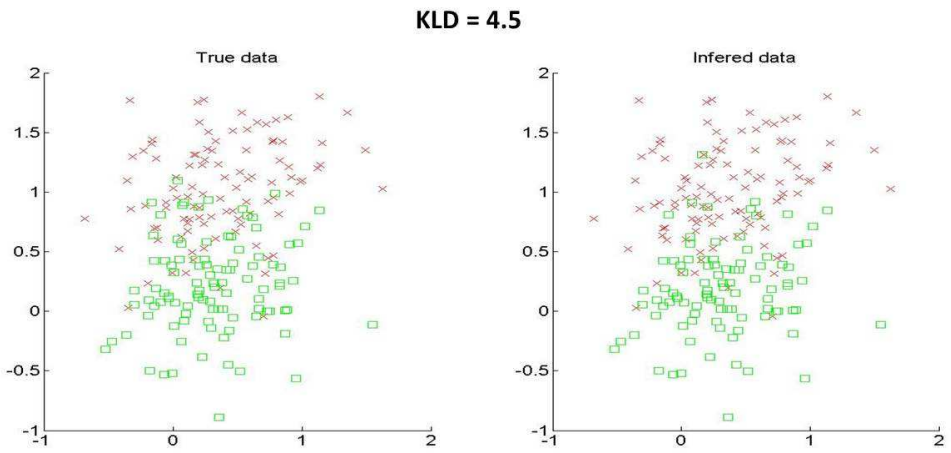


Figure 2.3 IGMM Output with 2 classes.

different approaches. And since the bounds are calculated with the knowledge of two classes, it eliminates errors caused by misinterpreting the number of classes in the nonparametric approaches. Hence, with the same settings, the likelihood ratio test is expected to outperform the nonparametric methods.

2.3.6 Illustration of the power of IGMM

In this subsection, to demonstrate the power of IGMM, we implemented a simulation with two set of observations following two multivariate Gaussian distributions. The KullbackLeibler distance between the two distributions is 4.5, which is very short. The number of observations is unknown to IGMM as well as the labels of the the observations. The result of IGMM in classifying the observations without the prior knowledge is given on Figure 2.3.

As one can see, although the two clusters are close to each other and overlapping excessively, the Gibbs sampler still achieves a high performance. Not only can it figure out the number of classes, it also have a high performance in sampling the label of each observation using Gibbs sampler. That is because of the joint distribution of the label is well approximated by the conditional distributions Student-t. MS cannot achieve such a high performance and may come up with only one cluster, since there is only one densest region on the plot. Further simulation results, in which we compare MS and IGMM performance with the performance bounds, will be described in Section 3.4.1.

Chapter 3

Identifying Primary User Emulation Attacks in Cognitive Radio

With the ongoing growth in wireless services, demands for radio spectrum have significantly increased. Cognitive radio (CR) [17]- [18] systems have been proposed to efficiently exploit the spectral holes unoccupied by licensed users to alleviate the spectrum scarcity problem. CR or secondary users (SUs) are wireless devices that can intelligently monitor and adapt to their environment, and hence, they are able to share the spectrum with the licensed primary users (PUs), operating whenever the PUs are idle. However, the cognitive nature of system introduces an entire new suite of threats and tactics that cannot be easily mitigated. One severe attack to CR systems is the primary user emulation (PUE) attack, suggested in [19]- [20]. In the PUE attacks, an attacker mimics the signals of the PUs, resulting in false alarms during the spectrum sensing phase at well-behaving SUs even though the PUs are actually not present. The malicious attackers can thus significantly degrade the performances of the well-behaving SUs.

There exist some works to alleviate the PUE attack [19]- [20]. However, most of them require the knowledge of the location and/or transmission power of devices. These parameters vary when PUs are moving or change their configurations. Furthermore, they can be easily emulated by an attacker if the attacker can change its position and/or transmission power. In this chapter, we propose a new method, called DECLOAK, to identify the PUE attacks. Our approach is named after a utility in Linux to discover hidden processes. It mainly comprises two components. Firstly, DECLOAK utilizes device dependent radio-metrics as fingerprints. A radio-metric is a component of radio signal, examples are amplitude, frequency and bandwidth. Each device creates a unique set of radio-metrics in its emitted signal due to hardware variability during the production of the antennas, power amplifiers, ADC and DAC circuits, etc. As a result, radio-metrics cannot be altered post-production, and thus provide a reliable means for identifying the real PUs and the PUE attackers. Secondly, DECLOAK utilizes nonparametric Bayesian classification. Based on these fingerprints, the feature space of a single device is modeled as a multi-variable Gaussian distribution

with unknown parameters, and the feature space of multiple devices as an infinite Gaussian mixture. The approach is unsupervised classification with an unbounded number of mixtures. Specifically, we define a prior of device features using the stick breaking process. Based on the properties of the stick breaking process, we apply the collapsed Gibbs sampling algorithm to sample from the posterior distribution, and determine the number of active devices. Third, we collect device IDs, e.g. MAC addresses. If more than one physical devices share the same ID, PUE attacks are identified. Our proposed method is passive and requires no modification of PU hardware. It is also based on unsupervised classification, and thus does not require any training data. We explore several different settings with different degrees of similarity among device features.

This rest of this chapter is organized as follows: In Section 3.1, related work is investigated. The system model is described in Section 3.2. The performance of the algorithm is evaluated in Section 3.4, followed by conclusion in Section 3.5.

3.1 Related Works

Several approaches have been investigated in literature to combat PUE attacks. They mainly fall into two categories: detection and defense mechanisms. In [19], Chen *et al.* proposed a location based authentication scheme in TV white spaces. If the signal from a transmitter has similar characteristics as TV broadcasters, localization based on RSS measurements is performed, which are matched against a known location database to determine whether the attacks have occurred. Clearly, such a scheme assumes that PUs are stationary and channel variations are not significant. A new method based on invariant channel parameters was recently proposed in [21]. Although it is claimed that the method is not location based, channel characteristics are in general dependent on the locations of devices. When an attacker is in the vicinity of legitimate PUs, the method might fail to detect the attack. In [22], the authors studied the impact of PUE attacks on the classifier performance in CR systems, and concluded that it is relatively easy to "fool" conventional supervised and unsupervised classifiers. To mitigate PUE attacks, the authors suggest the use of multiple classifiers and a combination of both offline (supervised) and online (unsupervised) classifiers. In [23, 24], Jin

et al. presented a Wald's sequential probability ratio test (WSPRT) to detect the PUE attacks. The probability density function (pdf) of the received signal from malicious users is obtained using the Fenton's approximation. The basic premise of the technique is that attackers' transmission power levels are significantly different from that of the PUs. It is also assumed the malicious nodes cannot change their transmitted power and the number of malicious node is fixed. On the defense front, Thomas *et al.* [20] provided a Bayesian game analysis of emulation attacks in CR systems, and demonstrate that depending on radios' beliefs about the fractions of PUs in the system, a policy maker can control assurance of PUE attacks by adjusting the gains and costs associated with performing and checking of PUE attacks. The work is different from our work since it does not focus on detecting PUE attacks. The work instead tries to cope with the attacks when they are detected. In [25], based on the assumption that there are multiple channels in the cognitive radio system, the authors propose an approach in which an honest SU intelligently and randomly selects a channel to sense and access at each time slot, and thus probabilistically mitigate the effect of PUE attacks.

Beside the works targeted at PUE attack detection, there have been some works in wireless device identification. One of the prominent works is [26]. It is close to ours in feature selection. However, there are two major differences between the works. First, [26] utilizes supervised classification methods while DECLOAK is based on unsupervised classification. Second, the classification methods in [26] are solely based on the average value of each feature, which can be easily replicated as pointed out in [27]. In [27], the authors implemented an attack on physical-layer identification methods. They proved experimentally that the average values of the selected features in [26] can be reproduced using a high-end arbitrary waveform generator. The result shows that certain radiometric myth can be spoofed. Radiometric fingerprint is considered hardware specific, thus approaches such as [26] are insufficient in identifying devices.

Nevertheless, in DECLOAK, we consider the distribution of the feature points in the feature space rather than the average values of features. Although a malicious device can spoof the average values of a specific feature, it cannot reproduce exactly the distribution of the feature itself, let alone the multivariate distribution of multiple features. In a feature space, from the same set of values on

each axis, we can create a large number of classes, each with different combination of the values. Each cluster corresponds to a single multivariate distribution. Hence, even if a malicious node can modify its signal to have the same average values of the original signal, it still cannot reproduce the multivariate distribution, which is the signatures of legitimate devices in our application. In addition, even if the malicious node is able to change the central frequencies [27], it is not likely to reproduce exactly subfrequencies since the frequency divider characteristics are device dependent. Our DECLOAK approach utilizes second-order cyclostationary features, which is calculated upon the subfrequencies of the transmitted signal. Hence, DECLOAK is robust to the attack.

3.2 Feature Extraction for Device Fingerprint

In this section, we introduce the features used in DECLOAK. To distinguish among different wireless devices, we need to determine the fingerprints, which is a group of unique features of RF devices. These features can be extracted from the transmitted signals, uniquely determined by the transmitter's characteristics, and not affected by receiver design. In addition, locations of the transmitters are also a good identifier for devices as two devices are likely to be separated by some distance. Combining both the RF fingerprints and the transmitter location leads to a strong identity for the device.

IGMM is defined in Definition 2.3 and next, we will demonstrate how the IGMM can be utilized in this application. First, as we can see, the number of wireless devices is unknown and can be assumed to be infinity initially. Each device creates a cluster of data points in the feature space. Each cluster has a distribution, which can be represented by a parameter vector $\vec{\theta}_k$. The parameter vector includes two components, $\vec{\mu}_k$ and Σ_k. $\vec{\mu}_k$ can be understood as an aggregate of the means of the four features. The same interpretation can be applied to Σ_k as the covariance matrix. Due to many effects during the production that contribute to the variability of the features, we can apply the law of large number to assume that $\vec{\mu}_k$ also follows a Gaussian distribution. The mixing weight, $\vec{w}$, i.e., the proportion of the whole data points each device contributed to is created according to the stick breaking process. α represents our confidence in the model. The larger the α, the stronger our

confidence in choosing the base distribution, $\vec{H}$ and the more concentrated the distributions of $\vec{\theta}_k$'s are around the base distribution.

In this chapter, we limit the scope by considering OFDM as the transmission modulation technique. Other techniques can be investigated in a similar way. OFDM is advantageous in minimizing the Inter-Symbol Interference (ISI) caused by multipath propagation or bandwidth-limited channels, hence, is widely used in many wireless standards. The specific properties of OFDM signal will be discussed in the subsection below. Then, based on these properties, a signal extraction process will be proposed to derive the unique features for device fingerprinting.

3.2.1 OFDM System Model

In OFDM, the bandwidth of transmitted signal is divided into N_s subcarriers, and data are transmitted in parallel. The frequency spacing is $\Delta f = 1/T_d$ with T_d as the time duration of one OFDM symbol transmission. Based on the channel condition, information (in the form of binary bits) is first modulated using schemes such as Binary Phase Shift Keying (BPSK), Quadrature Phase Shift Keying (QPSK) and Quadrature Amplitude Modulation (QAM) to represent symbols. By using Inverse Discrete Fourier Transform (IDFT), the symbols are then multiplexed to generate OFDM signals, which can be expressed in the following form:

$$y_b(t) = \sum_{k=1}^{N_s} Y_k e^{j2\pi k\Delta f t}, \tag{3.1}$$

where Y_k is the k^{th} symbol embedded in the k^{th} subcarrier. Without loss of generality, we assume that the amplitude of any symbol Y_k is 1. The signal is then converted to a carrier band signal with carrier frequency f_c and transmitted on a multipath fading channel (such as a Rayleigh channel):

$$h(t) = \sum_{p=1}^{N_P} A_p \delta(t - \tau_p), \tag{3.2}$$

where τ_p, A_p, and N_P are the delay time, channel gain in the p^{th} path, and the number of paths, respectively.

Beside the transmitted signal and the channel model, several specific designs have been in-

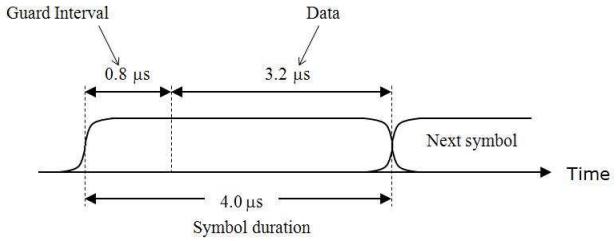
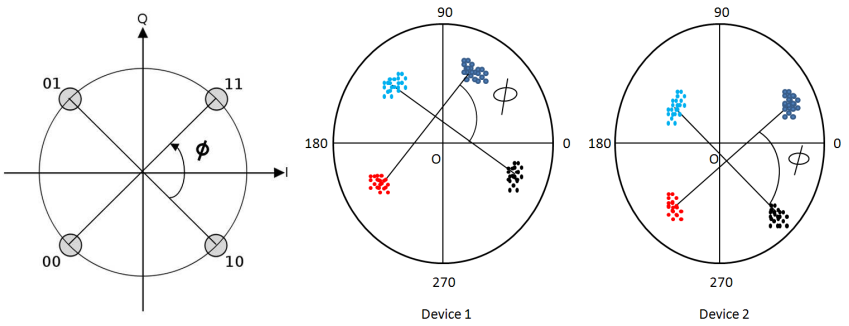


Figure 3.1 802.11a/g PHY frame structure.

corporated to combat multipath fading. The first design is the length of the PHY frame. Figure 3.1 illustrates the structure of the IEEE 802.11a/g physical layer frames. A Guard Interval (GI) is appended to be no shorter the longest possible path delay that may lead to ISI. The length of GI, denoted as T_{CP}, is $0.8\mu s$. Together with the data segment of length $T_d = 3.2\mu s$, a PHY frame is formed with symbol time $T_s = 4.0\mu s$ [28]. The second design is the locations of the pilots. There are totally 64 subcarriers (N_s = 64) but only 52 out of those are used, and the others are the guard bands. Among the 52 occupied subcarriers, 48 are used for data transmissions and the other 4 subcarriers at locations -21, -7, 7 and 21 are used for pilots. The pilots are purposely inserted in the signal to estimate the channel. However, in DECLOAK, they are utilized to exploit the periodicities embedded in the signal.

3.2.2 Extracted Features

Based on the specific signal properties as well as the PHY frame structures of OFDM signals, we propose to use the following features for identification purposes: i) the carrier frequency difference, ii) the phase shift difference, iii) the second-order cyclostationary and iv) the amplitude of the received signal. The first three features form fingerprints for devices. The last feature depends on location of the devices and is not device dependent. However, it helps to improve the overall performance of the DECLOAK algorithm. In [26], it has been demonstrated experimentally that the frequency difference and the phase shift difference are device dependent. The third feature, the second-order cyclostationary, strongly depends on the oscillator and the frequency divider in each device. Since the oscillator is device dependent and so is the frequency divider, the third feature is also device dependent. In our approach, we assume all the features follow Gaussian distribu-



(a) Ideal phase shift in QPSK (b) Phase shift difference in the constellation of two Tx devices

Figure 3.2 Illustration of Phase Shift Difference for constellation of symbols.

tion. However, we also show that the performance of DECLOAK is not very sensitive to the exact distribution of the feature space. Below, we describe the features used and how they are derived.

3.2.2.1 The Carrier Frequency Difference (CFD)

The first feature, denoted as Δf_c, is defined as the difference between the carrier frequency of the ideal signal and that of the transmitted signal. Δf_c is a strong feature since each wireless transmitting device has its own oscillator, and each oscillator creates a unique carrier frequency. For OFDM application with low mobility, the Doppler effect is not severe. Moreover, due to reflection, the Doppler effect will cause the carrier to smear on both sides of the frequency. Consequently, the transmitter oscillator is the major factor for this feature. We assume that Δf_c follows a Gaussian distribution $\Delta f_c \sim N(0, \sigma_2^2)$.

3.2.2.2 The Phase Shift Difference (PSD)

Although OFDM uses different modulation techniques and each technique produces a specific constellation, most of the constellations share some commonalities. For example, the phase shifts from one symbol to the next one are created in the similar way in hardware and are transmitter dependent. Thus, for the sake of simplicity, we use QPSK as an example and consider feature extracted from the constellation of QPSK as shown in Figure 3.2(a). In QPSK, four symbols with different phases are transmitted and each symbol is encoded with two bits. The phase different between two consecutive symbols is ideally 90^o. However, the transmitter amplifiers for I-phase and

Q-phase might be different. Consequently, the degree shift can have some variances. Figure 3.2(b) shows an illustrative example of device signal constellations. The constellation may deviate from its original position due to hardware variability, and different devices may have different constellations. Therefore, we introduce the phase shift ϕ as one feature in our classification method. Due to the possible I/Q ambiguity in QPSK, we consider the smaller of the two possible angles.

3.2.2.3 The Second-Order Cyclostationary Feature (SOCF)

The aforementioned features are generic to most radio technologies. For OFDM signals, specific features can be derived. We consider the use of the second-order cyclostationary statistics [29] for identification. To decide if a signal is wide sense (or second order) stationary, its autocorrelation function is generally used. Similarly, the signal can be determined to be second order cyclostationary by considering its Cyclic Autocorrelation Function (CAF),

$$R_y^\alpha(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} y(t+\tau/2)y^*(t-\tau/2)e^{-j2\pi\alpha t}dt. \tag{3.3}$$

If $R_y^\alpha(\tau) \neq 0$ when $\alpha \neq 0$, then the signal is said to exhibit second-order cyclostationary at cycle frequency α and time lag τ.

In the above equation, we can see that inside the integration is a complex exponential term. As $T \to \infty$, $R_y^\alpha(\tau)$ is always 0 except when the complex number becomes a real number. In that case, T will be canceled out, and the limit becomes a real number. Thus, if $R_y^\alpha(\tau) \neq 0$ when $\alpha \neq 0$, then the signal is said to exhibit second-order cyclostationarity at cycle frequency α and time lag τ. Below, we will explain in more details why at $\alpha \neq 0$, it is possible that $R_y^\alpha(\tau) \neq 0$.

Each modulation technique has a different set of cycle frequencies. Consider (3.1), replacing $k\Delta f$ by f, we have:

$$y(t) = \sum_f Y(f)e^{j2\pi ft}, \tag{3.4}$$

where $XY(f)$ is the Fourier coefficients of $y(t)$. Replacing (3.3) with (3.4), we have the CAF [30]:

$$\begin{aligned} R_y^\alpha(\tau) &= \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}\left[\sum_{f_1} Y(f_1)e^{j2\pi f_1(t+\tau/2)}\right] \\ &\cdot \left[\sum_{f_2} Y^*(f_2)e^{-j2\pi f_2(t-\tau/2)}\right] e^{-j2\pi\alpha t}dt \\ &= \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{T/2}\left[\sum_{f_1}\sum_{f_2} e^{j\pi(f_1+f_2)\tau}\right. \\ &\cdot \left. Y(f_1)Y^*(f_2)e^{j2\pi(f_1-f_2-\alpha)t}\right]dt. \end{aligned} \tag{3.5}$$

If $y(t)$ is periodic, its fourier transform, $Y(f)$ peaks at those frequencies, and equals 0 otherwise. From (3.5), we see that $R_y^\alpha(\tau)$ is nonzero when $\alpha = f_1 - f_2$ where f_1, f_2 are possible embedded frequencies of signal $y(t)$. Or in other words, the complex number $e^{j2\pi(f_1-f_2-\alpha)}$ now becomes a real number. In this case, we will have nonzero CAF and nonzero α, as explained above. As a result, $\alpha = f_1 - f_2$ is the cyclic frequencies. If the signal has a periodicity embedded structure, we can always find out α , and α can be interpreted as the differences between the periodicities.

As mentioned above, there are many embedded periodicities in an 802.11a/g OFDM signal. For example, every $3.2\mu s$, a GI is inserted at a length of $0.8\mu s$. Also, the pilot signals use some of the subcarrier frequencies, providing another group of cycle frequencies. Based on those periodicities, the cycle frequency of OFDM signal can be calculated as [31]:

$$\alpha = \pm\left\{2k\Delta f + \left[-\frac{\beta}{G/2}, \left(1-\frac{\beta}{G/2}\right)\right] f_s\delta(\beta)\right\}, \tag{3.6}$$

where $\beta = \mod(k, G/2)$, and k is the location index of the pilots ($k = -21, -7, 7, 21$). $G = 3.2\mu s/0.8\mu s = 4$ is the ratio of the length of data transmission time over GI. $\Delta f = 1/T_d = 312.5$ KHz and $f_s = 1/(T_d + T_{CP}) = 250$ KHz. $\delta(\beta)$ is zero when $\beta = 0$, and 1 otherwise. In (3.6), $[\cdot,\cdot]$ indicates that either the left term or the right term can be taken into account. The cycle frequencies are thus:

$$\alpha = \pm\{0, 4.25, 4.5, 8.75, 13, 13.25\}\text{MHz}. \tag{3.7}$$

Theoretically, at the above values of α, CAF should be maximum. However, due to hardware variability during production, f_1 and f_2 are not exactly the same as their designated values. As the result, each device is associated with a different set of α values, or equivalently different CAF values at the above α values. Hence, the value of CAF at each cycle frequency can be used to identify devices. In [31], it has been shown that at $\alpha = 4.25$MHz, the difference in the CAF among devices is the biggest experimentally. For the sake of simplicity, we only consider $\alpha = 4.25$MHz. We define the third feature as the values of CAF at $\alpha = 4.25$MHz, where α is the cycle frequency. We assume the measured CAF follows a Gaussian distribution $R_y^{\alpha}(\tau) \sim G(\mu_3, \sigma_3^2)$, with mean μ_3 and variance σ_3^2.

3.2.2.4 Feature 4: The Received Signal Amplitude

Location is a useful feature in identifying devices as demonstrated by several works on device identification [19,21,32]. Compared to other techniques, OFDM is especially suitable for employing location based identification methods due to the fact that OFDM is primarily designed for low mobility scenarios. Under low mobility, the channel gain does not change very significantly. Hence, the magnitude of the received signal can effectively serve as one dimension to identify devices. Though the absolute value of the amplitude is generally unknown, the signals transmitted from the same device over a short duration tend to have a similar amplitude. The amplitude of received signal is proportional to the channel gain, A_p. In a Rayleigh multipath channel, the channel gain takes the following form:

$$A_p \sim d^{-\beta}|h|, \;\; |h| \sim CN(0, \sigma_1^2),$$

where d is distance from a transmitted device to the sensing device. β is the path loss exponent, and $|h|$ is the fading component. As the result, the fourth feature, A_p is defined as $A_p \sim CN(0, \sigma_1^2 d^{-2\beta})$ where CN stands for the Complex Gaussian distribution. It is important to mention that we do not infer the locations of devices directly, instead, we only use A_p as one feature dimension for the purpose of identification. It should be noted that, when the PUE attackers are at close proximity of the PUs, this feature alone is insufficient to distinguish them.

The proposed scheme in this chapter can be easily extended to incorporate other features. The four features above constitute a data point in the 4-dimensional feature space. The data points recorded over time from the same device will concentrate in a region in the feature space and follow a specific distribution. Those from different devices will likely form different classes, or equivalently follow different distributions. The information on the number of classes or ideally the number of unique devices is key to the detection of identity spoofing. For instance, if multiple cluster labels map to a single device ID, then we can infer the existence of a PUE attack.

Note that a *cluster* is modeled as a distribution with a certain unique parameter set. A parameter set, in turn, represents a unique *class*. Thus, in this chapter, the two terms are used interchangeably depending on different contexts.

3.2.3 DECLOAK Detection Scheme

Given the indicators, $\vec{Z}$, from the output of the Gibbs Sampler, we propose the following procedure to detect PUE attacks (Figure 3.3). At first, spectrum is sensed to detect if there are active PUs in their assigned frequency bands. If there are, the received signal is then sampled and processed in two paths. In the first path, IDs of PUs (including the malicious ones) are extracted (e.g., the MAC address field in the 802.11a/g MAC frame). In the second path, the device dependent fingerprints are extracted based on the proposed model. If an attacker joins the system, it has to declare the same MAC address as a PU. Thus, a PUE attack can be detected by comparing different device dependent fingerprints with their associated IDs. If there are two or more fingerprints associated with the same ID, an alarm will be raised. Note that one advantage of the proposed solution is that we do not need to maintain a database of features for the legitimate PUs, which may be subject to changes due to equipment upgrades, relocation, addition or termination of devices, etc.

3.3 Implementation Considerations

3.3.1 Selection of Hyperparameters

In this section we discuss some considerations in implementing DELOAK. First, we will describe a data-driven approach to set the hyperparameters in the IGMM model. In theory, regardless

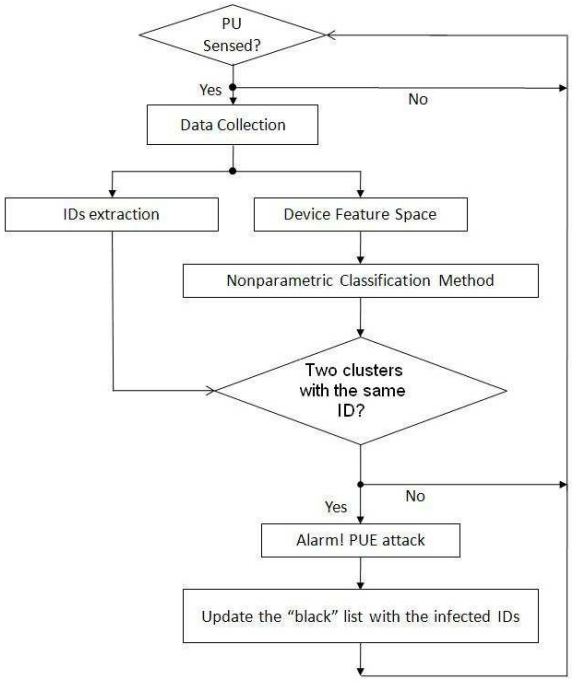


Figure 3.3 DECLOAK Detection Scheme.

of the values of the hyperparameters, the algorithm always converges to the ground truth given we have infinite number of observations. However, in practice, the number of observations is limited. Furthermore, the computation complexity of the Gibbs sampler methods grows significantly as the number of observations increases. Thus, for fast convergence, it is desirable to set the hyper parameters as close to the truth as possible.

In IGMM, we have the control over the following hyperparameters: $\alpha, \vec{\mu}_0, \kappa_0, \Lambda_0$ and v_0. α encodes the number of classes. If α is big compared to the number of observations, we are likely to have more classes and vice versa. As implied by (2.37) and (2.38), with big a α, the probability of assigning an observation to an unrepresented class is higher than that to a represented class. In our experiment, α is chosen to be small since from our prior knowledge, we know that we do not have too many devices in the limited range of sensing device.

In Definition 2.3, we see that the mean vector of a class follows a Gaussian distribution with mean $\vec{\mu}_0$ and covariance matrix Σ_k/κ_0. Thus, the two hyperparameters $\vec{\mu}_0, \kappa_0$ reflects the mean locations of classes. $\vec{\mu}_0$ represents the initial mean values of the features. For example, for the CFD,

the mean value can be set at 0. Generally, $\vec{\mu}_0$ can be set to the mean values of all observations in the 4-D feature space. κ_0 is associated with the dispersion of the classes, if κ_0 is high, then the classes tend to be closer to each other. On the other hand, if κ_0 is small, we will have a big variation in Σ_k/κ_0, and thus, the classes are sparser. Based on our prior knowledge about the feature space, we can choose $\vec{\mu}_0$ and κ_0 accordingly.

Again, from Definition 2.3, we see that the covariance matrix of a cluster depends on the two hyperparameters: v_0 and Λ_0. These are the parameters of the Inverse Wishart distribution. The first parameter, v_0, is the degree of freedom, and the second parameter, Λ_0, is the scale matrix for the distribution. By changing Λ_0, we initialize the variances of the marginal . For example, since the crystal oscillator in each device is very precise, it is likely that we will have a small variance for the CFD. Hence, we set the value for the corresponding element on the diagonal of Λ_0 to a small value. Unlike CFD, the received signal amplitude has a big variation due to the nature of wireless channel. Thus, the corresponding element on the diagonal of Λ_0 should be set to a large value. Finally, v_0 represents our confidence about Λ_0. If v_0 is big, it is unlikely that the covariance matrix or a cluster will be much different from Λ_0 and vice versa.

3.3.2 Handling Device Mobility

When devices are fast moving or there are multiple devices at the same location, amplitude is not an effective feature. More specifically, when the mobility is high, the same device may create multiple classes in the 4-D feature space over time. On the other hand, when multiple devices are at close proximity, the amplitude of the received signals may be similar. In both cases, we can remove the amplitude feature and perform classification using the other three features. We call this algorithm DECLOAK-3D. When devices are stationary, DECLOAK-3D serves as a lower bound of the performance of DECLOAK-4D. The gap between the two schemes reflects the contribution of the amplitude feature.

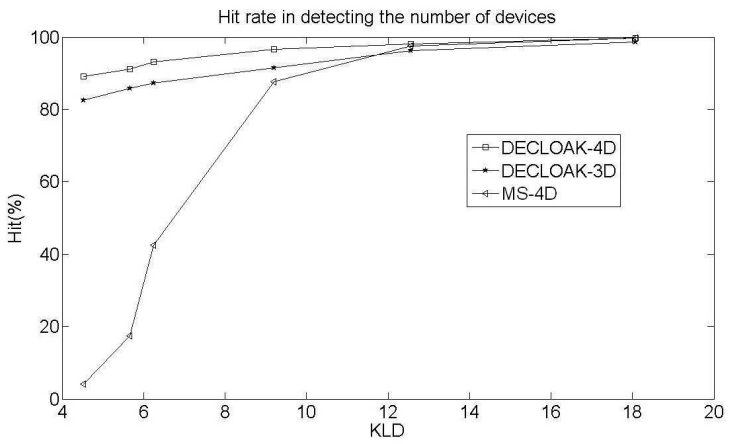


Figure 3.4 Two devices: a comparison of Type 1 performance in DECLOAK and MS.

3.4 Simulation Result

In this section, the performance of DECLOAK will be compared with that of a baseline method, the Mean Shift based Classification method (MS) [12]. Performance bounds for the classification problem formulated from the previous section will also be calculated for comparison. Two cases are considered. In the first case, there are only two active devices, resulting in two classes in the feature space. KLD is varied to change the hardness of the problem. In the other case, the number of devices is varied from one to one hundred to evaluate the scalability of the algorithm. After that, we dedicate a subsection to study the effect of mobility on performance result. Finally, the case of non-Gaussian distribution is investigated. Since we assume all the feature distributions follow Gaussian distribution, it is important to see how the algorithm performs when the distribution is not Gaussian. We first give a brief introduction of MS below.

3.4.1 Two-Device Case

This set of experiments serve the purpose of evaluating the performance of the algorithm by varying the hardness of the problem. In this setting, there are only two active devices (the number of devices is unknown to the algorithm), resulting into two classes in the feature space. We vary KLD to change the problem hardness. Since the feature space consists of 4 features, we name the algorithm DECLOAK-4D and the Mean Shift algorithm as MS-4D.

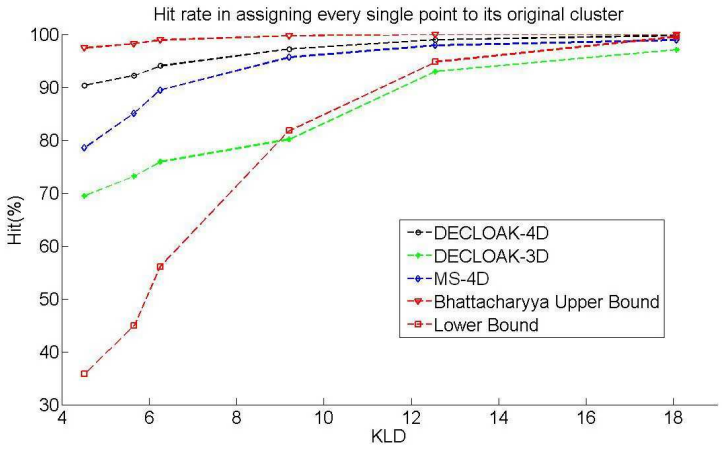


Figure 3.5 Two devices: a comparison of Type 2 performances in DECLOAK, MS and the bounds.

As expected, in Figure 3.4, DECLOAK-4D outperforms the MS-4D in most of the values of KLD except when $KLD = 18.1$. When $KLD = 18.1$, the two classes are well separated and there is almost no overlapping. As the result, the MS achieves a high Type 1 hit rate of 99.94%. However, the MS performance drops dramatically when KLD decreases. When $KLD = 4.5$, the hit rate is only 4% while DECLOAK-4D can still achieve a hit rate of 90%. Note that at $KLD = 4.5$, the two classes are almost overlapping each other. Specifically, mean of one cluster is on the boundary of the other cluster. Hence, the performance is very desirable. A low degree of separation between two devices may be the result of noises in the measurements data. Thus, DECLOAK-4D is more robust to noisy measurements.

The Type 2 results are given in Figure 3.5. In this aspect, DECLOAK-4D outperforms MS-4D again. Though both are within the bounds, DECLOAK-4D is closer to the upper bound compared to MS. Both type 2 and type 3 results of MS and DECLOAK-4D are shown in Figure 3.6. Please note that the curves in the figure are not ROC curves, since we change $Diff$ at every step. It can be seen from the figure that DECLOAK-4D maintains a high Type 2 hit rate while keeping the Type 3 false alarm reasonably low, compared to MS-4D.

3.4.2 Varying the Number of Devices

In the second scenario, we consider a more complex setup where the number of devices changes over time up to 100 in total. Each device's feature cluster has a diameter around 0.5 and

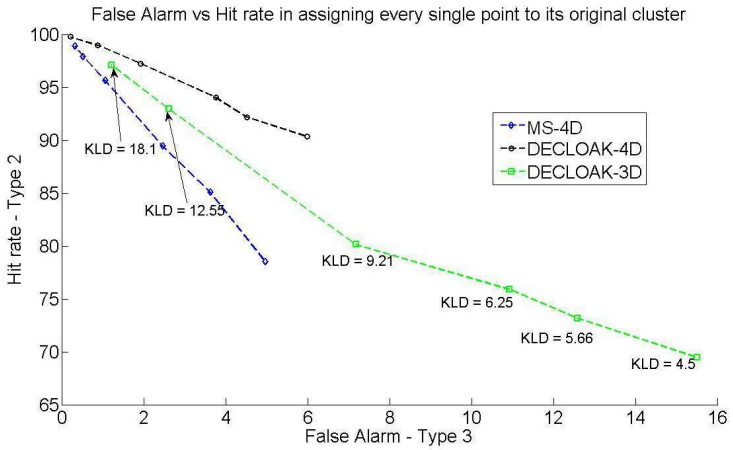


Figure 3.6 Two devices: Type 2 vs Type 3.

a mean randomly selected in a space of $10 \times 10 \times 10 \times 10$. Note that KLD can be very low since there is a chance in which some classes may locate in close proximity in the feature space. In Figure 3.7, when the number of devices increases, the detection accuracy decreases. However, even at $N = 100$, when the probability of having two or more classes at the same location is very high, the Type 1 hit rate in DECLOAK-4D is still as high as 95.5%. In contrast, at $N = 100$, MS-4D can achieve a Type 1 hit rate of 46.2%. The Type 2 performances of DECLOAK-4D and MS-4D are also given in Figure 3.8. Similar observations can be made. Changing the number of devices from 99 to 100, we notice that the algorithm takes approximately $1365s$ to detect there is a new device. The number of observations is kept at 2500 and we run the program on a single core computer with CPU speed at $2.33GHz$. On average, each device contributes 25 data points in a period of approximately 25 minutes. Hence, it is reasonable to collect data for 25 minutes and run the algorithm offline while keep collecting data. We use a personal computer to run the algorithm, but in the case of a super computer, the running time should be much reduced. Considering the situation in which there are 99 users and a new attacker joins the system, we notice that after receiving 25 new observations of the attacker from the sensing device, the algorithm can detect the attack and raise an alarm. In combination with the results displayed on Figure 3.7, we can conclude that with a super computer, we can immediately detect a new device with a hit rate of 96%. In the case the new device has the same ID as the ID of another PU, the new device will be determined as an attacker. Otherwise, we

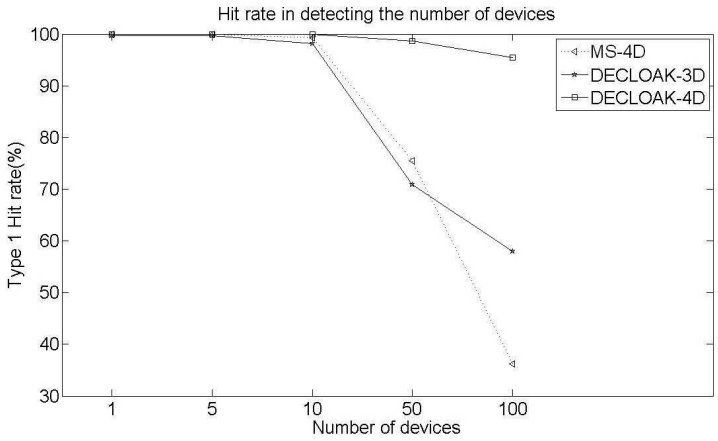


Figure 3.7 Varying the number of devices: Type 1 hit rate.

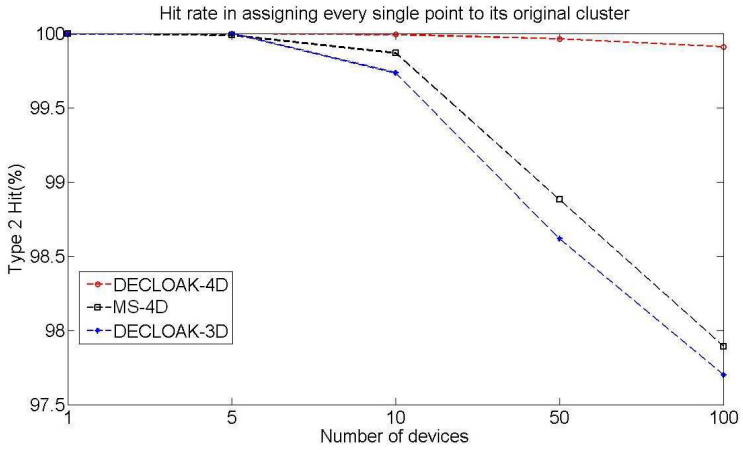


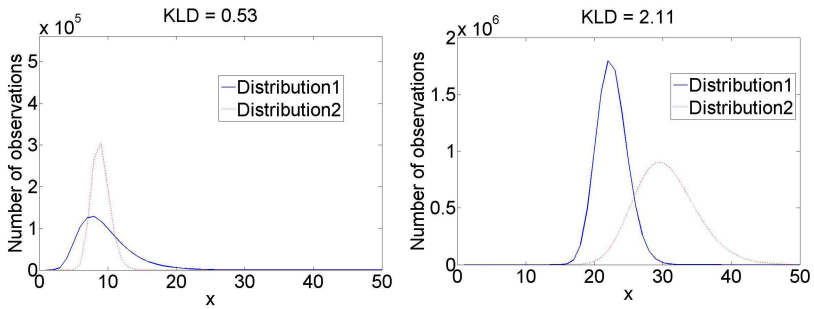
Figure 3.8 Varying the number of devices: Type 2 hit rate.

can claim the new device as a legitimate PU.

DECLOAK focuses on detecting PUE attacks. If there is a PUE attack, an alarm will be raised, as explained in Figure 3.3. However, since we do not store fingerprints of users, we cannot distinguish legitimate PUs from attackers in the case of an attack. Further action by system administrators is required to determine which user is legitimate.

3.4.3 Effect of Amplitude Feature

In this section, we evaluate the effect of amplitude feature. As discussed in Section 3.3.2, with high mobility or when devices are at close proximity, DECLOAK-3D only utilizes the CFD, PSD, SOCF features. As shown in Figures 3.4, 3.5, 3.6 for the case of two devices and Figures 3.7, 3.8 for the case of varying number of devices, DECLOAK-3D generally outperforms MS-4D



(a) KLD = 0.53 (b) KLD = 2.11

Figure 3.9 Two devices: One-dimension amplitude feature follows log-normal distribution.

or performs similarly as MS. Note that in the comparison, MS uses 4 features. As expected, in all scenarios, DECLOAK-4D outperforms DECLOAK-3D since devices are stationary. However, the amplitude feature play a more important role when the KLD is small or the number of devices is large.

3.4.4 Performance in the Case of Non-Gaussian Distribution

In practice, the assumption of Gaussian mixture model may not always hold. For example, in wireless communication, the gain of a slow fading channel typically follows log-normal distribution. In this section, we evaluate the sensitivity of DECLOAK when the underlying distribution is non-Gaussian. Specifically, we consider two devices with the one-dimension amplitude feature following log-normal distribution. The other features are omitted to single out the effect of the non-Gaussian distributed. Two cases with KLD = 0.53 and 2.11, respectively, are evaluated. As shown in Figure 3.9(a) and 3.9(b), at KLD = 0.53, the two distributions overlap a lot with each other. DECLOAK still achieves a high Type 1 hit rate, 82.1%, while MS can only achieve 45%. In the case of KLD = 2.11, the Type 1 hit rate of DECLOAK is 88% while that of MS is 47%. From the simulation, we can see that though the performance of DECLOAK degrades for non-Gaussian mixtures, it is relatively robust, and significantly outperforms the MS method.

3.5 Conclusions

This chapter proposed DECLOAK, a nonparametric Bayesian approach for detecting PUE attacks. DECLOAK exploits the OFDM signal features, the Infinite Gaussian Mixture Model for

modeling, and a collapsed Gibbs sampling method for classification. It has been shown through simulation studies that DECLOAK significantly outperformed a baseline method, the Mean Shift based classification method, and approached the upper bound performance. Besides, the unsupervised nature of the method is very desirable in practice, since it is either impossible to store all wireless device signatures in one single sensing station or required to have a complicated collaborating network between multiple sensing stations in a large-scale wireless network. Though the algorithm is specifically designed for PUE attack detection in CR, it can be easily extended to many other security applications.

Chapter 4

LOIRE: Lightweight Online Inference of Recurring and New Indoor Places of Mobile Users

LBS are services provided to improve users' experiences based on their locations. There are several important aspects in LBS. First, the most important task is to identify users' places. For outdoor, this information is trivial given GPS signal from users' smartphones. However, for indoor environment, this information is missing. We, thus, focus on the problem of indoor location identification. Second, given the location information, it is then of interest to predict the users' future whereabouts. These are the two major objects of the subsequent chapters.

Understanding and modeling users' daily activities can find applications in many domains. An important aspect of daily activities is the places a person visits and the duration he/she stays in each place. It has been reported that human mobility has high predictability. Based on 3-month-long cellular data, Song *et al.* [33] find that only 7% of the time, an individual appears to choose his/her location at random, while 93% of the time, a person's whereabout can be predicated accurately. Such predictability can be exploited in targeted advertisement [34], energy saving in micro-climate control [35, 36], delivery of anticipatory notifications about traffic jams, etc. Given that majority of the day humans stay indoor, identifying indoor places complements existing studies of human mobility, which primarily focuses on outdoor activities or macro-level places (e.g., shopping mall, work, home, etc.) [37].

4.1 Introduction

The goal of this work is to infer new and recurring indoor places a person visits using a small number of measurements. This problem differs from indoor localization in several aspects. First, transient locations (e.g., doorway) are typically of little interest to identifying important places. This gives rise to opportunities for energy savings by keeping track of only places that a person stays for sufficiently long periods of time. Second, different places are associated with distinctive

labels (e.g. Place A, B, C, etc.) or logical identifiers rather than physical coordinates. This allows us to adopt a profiling-free (unsupervised) approach without the need to obtain training data such as fingerprints for each target location. Third, one main design goal is to minimize the number of measurements needed in the identification (and thus the energy expenditure), while most prior work in indoor location is primarily concerned with location accuracy. Admittedly, once coordinates of a user is known, the corresponding place label can be inferred. However, we argue that existing localization approaches are either not yet ready for practical deployment or are too costly (in energy consumption or processing overhead) for the simpler problem of inferring new and recurring indoor places addressed in this work.

To illustrate the utility of inferring important places, let us consider the following two applications:

- *Micro-climate control:* In this application, a room is heated up or cooled down in anticipation of the arrival of its occupants. Such an application can benefit from the knowledge of the places a user visited, the length of stay in a place and the estimation of the likelihood of his future whereabout to improve his comfort level. Note that occupancy information alone is insufficient as it provides the current physical presence of the user.
- *Co-location tracking:* It has been shown that more interactions among co-workers tend to improve productivity in work place [38]. Generally, co-location is a good indication of direct interaction. Additionally, for applications such as friendship suggestions, it is of interest to find out if users visit common places at different times.

Note that neither applications require physical coordinates of places and yet can benefit from the knowledge of places that a person frequents over time.

Our proposed solution, LOIRE, combines accelerometer and WiFi scans on smart phones to infer important indoor places. The incorporation of accelerometers serve two purposes. First, it detects movements and triggers WiFi scanning when a person remains stationary sufficiently long. Second, by segmenting time into periods of different movements such as being stationary, walking,

and running, we immediately obtain a collection of (unlabeled) places, means of transit and a rough distance estimate from one to another. In this work, we focus on the places alone. Since all WiFi measurements collected throughout the stationary period come from the same place, we can utilize such knowledge to design efficient algorithms for detecting new places and classifying recurring places.

Using WiFi scans as location features pose several challenges, some of which are not well recognized in literature. First, WiFi signals tend to fluctuate as the result of multipath effects in indoor environments. Second, received signal strength (RSS) measurements from different devices can differ a lot. However, offline calibration of heterogeneous devices are not always feasible. Third, continuous collection of WiFi signals from multiple access points is power consuming. Fourth, due to the existence of many access points (APs) in typical office buildings or a residential neighborhood, a single WiFi scan may contain the RSS readings from only a subset of possible APs within the range. Compound with the fact that APs outside the range cannot be scanned, we observe situations where both missing at random (MAR) and missing not at random (MNAR) occur [39]. Existing approaches to deal with missing data in WiFi-based localization employ one or more of the following techniques, namely, i) data imputation by substituting missing data with a fixed value; ii) selection of top k RSS readings from each scan; and iii) use of set differences such as the Jaccard distance and Tanimoto distance. While these approaches can alleviate the impact of missing data, they treat MAR and MNAR non-discriminatively.

In this chapter, we address all the challenges in a probabilistic framework. RSS readings from a single AP at different locations are modeled as Gaussian mixtures. Our first contribution is to systematically handle incomplete RSS data, which allows grouping places with similar RSS readings despite MAR and MNAR. In an effort to reduce the power consumption of WiFi scans, our second contribution is to devise an online classification mechanism that accumulates *just enough* measurements and stops the WiFi interface from scanning until the user moves again. The online approach first determines whether a new place is visited, and if not, it selects one of the previously visited places as the label for the current place. LOIRE is non-parametric and assumes no knowledge

Table 4.1 Comparison of indoor place identification algorithms.

Approaches	Profiling-free	Sensors	# of WiFi scans per session/zone
SurroundSense	No	WiFi, acoustic, vision	n/a
SensLoc	No	WiFi, Accel, GPS	39/n/a
WILL	YES*	WiFi	n/a
LOIRE	Yes	WiFi, Accel	30/150
ARIEL	Yes	WiFi, Accel	3.5/80

*WILL requires the knowledge of the number of places

of the number of places a user visits. It is profiling-free and requires no manual on-site survey. LOIRE has been validated using multi-week long traces collected from multiple smart phones from two buildings on campus. It can accurately identify a revisited place and a new place 92.2% and 100% of the time. The rest of the chapter is organized as follows. In Section 4.2, we give a brief review of existing literature pertaining to our work. Then, we will present an overview of LOIRE in Section 4.3.

4.2 Related Works

There is a significant body of work on wireless indoor localization. A recent survey can be found in [40]. In this section, we discuss major indoor place identification works listed on Table 4.1. Generally, there are two categories of approaches in wireless indoor localization or place identification, namely, profiling and profiling-free. In profiling approaches [41–43], fingerprints for each location are first obtained through manual survey to construct a fingerprint map, which is later used to find the best match with the new measurements at unknown locations. In SurroundSense [41], fingerprints are generated from WiFi signal strength, ambient sound measurement, acceleration, color and light readings. SurroundSense is shown to be very accurate in identifying places though the good performance comes at the expense of utilizing sensing modalities with high energy consumption such as camera.

SensLoc [44] is an energy-efficient profiling approach for sensing everyday places. It bares some similarity with LOIRE in the use of accelerometer to detect motion to trigger WiFi scanning. In SensLoc, input Wi-Fi scans are treated as vectors so that the Tanimoto coefficient can be used. A group of scans within a window, the response rate of each AP (defined as the percentage of

beacons observed) are used to determine the fingerprint. Signals from unobserved beacons are imputed with zero. The approach can identify correctly 96% of revisited places. However, the performance is sensitive to the threshold chosen for the Tanimoto distance between fingerprints. And most importantly, SensLoc requires users to label all visited places.

Profiling-free localization solutions [45, 46], in contrast, do not require users to collect fingerprints from different locations beforehand. WILL [46] uses RSS from APs as fingerprints and computes a pairwise similarity score of any two fingerprints from their Euclidean distance. The distance is then used in K-means clustering to group measurements into virtual rooms. Missing data are not explicitly dealt with in WILL. Furthermore, in K-mean clustering, the number of clusters (or, the number of places) needs to be specified before hand. Thus, it cannot detect a new place, which makes it less practical.

ARIEL is a non-parametric approach for indoor localization, which does not require the number of locations to be known a priori. It defines a WiFi session as a set of signals collected when users are stationary. Ariel utilizes n-gram [47] to determine the distance between the WiFi sessions Sessions are grouped into one zone if, i) they are within a certain distance threshold, and ii) their quantity exceeds a minimum number of points. The zone-based clustering is followed by a motion-based clustering, which groups zones together into rooms. ARIEL assumes that the order of RSS from APs is mostly fixed within one location, and the order is independent of the device characteristics. However, we find through our measurement study that the first assumption is often violated. Moreover, ARIEL requires a total of 30 samples per session and 5 sessions per zone. This leads to considerable energy consumption in WiFi scanning.

Main contributions LOIRE differs from existing indoor localization and place identification approaches in three important aspects. First, it is a profiling-free, non-parametric approach that does not require site survey and the knowledge of the number of places. Second, it deals with missing data in WiFi scans in a systematic manner. Third, LOIRE treats energy as the first class resource and minimizes not only how frequent WiFi scanning is conducted but also its duration. These aspects will be described in details in the subsequent sections.

4.3 Overview of LOIRE

In LOIRE, two sources of data are collected from a mobile device: three-axis accelerometer readings and WiFi scanning. Accelerometer data is collected continuously while WiFi scans are obtained only as needed. Using the accelerometer data, we first classify a user's states as stationary or moving. If the user is stationary for T_s time, where T_s is a pre-specified system parameter, WiFi scans are started until LOIRE decides, whether the user has returned to a previously visited place and the same label is used; or he/she is at a new location, in which case, the user is *optionally* asked to provide a label. Incorporating user inputs not only makes it possible to associate a physical location with its logical identifier (e.g., "my office", "kitchen area on the first floor", etc.) but also allows the user to make corrections if LOIRE mistaken an old place as a new one. LORIE is tuned such that the probability of identifying a new place as old is minimized while the probability of identifying an old place as new is within a tolerable range. Once the label is decided, WiFi scanning is stopped to conserve power.

LOIRE can run on individual phones if users only choose to keep the information for their own use as in the example application of scheduling delay-tolerant data. It can also run on a centralized server, which collects data from multiple users and provides additional feedback for co-location tracking.

4.3.1 Design challenges

The main challenges in the design of LOIRE arise from the stochastic nature of WiFi signal measurements and limitation of WiFi interfaces on smart phones as sensing devices. Most of the challenges are well documented in literature including time varying RSS measurements and device heterogeneity. In this subsection, we focus on the problem of incomplete and provide empirical evidences from our own measurement study.

Incomplete data Table 4.2 lists three RSS measurements from different APs on a single device. The first two observations are collected in the same location (P_1) consecutively in time. The third observation is collected in another location approximately $26 feet$ away (P_2). One can observe

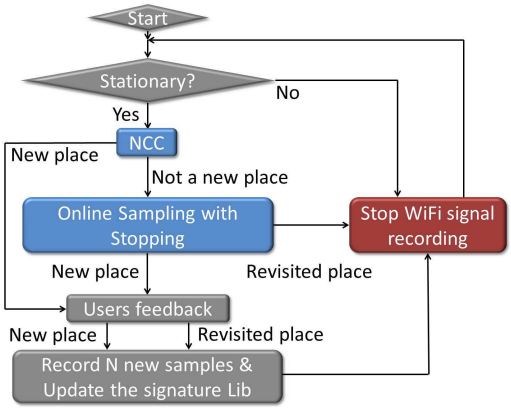


Figure 4.1 LOIRE algorithm.

Table 4.2 Example of observations.

Observation 1 from Room 1

AP	1001	1002	1074	1073	1050	1075
RSS	-67	-69	-89	-76	-69	-76

Observation 2 from Room 1

AP	1001	1002	1074	1073
RSS	-66	-66	-89	-76

Observation 3 from Room 2

AP	1001	1002	1068	1070	1073	1076	1077
RSS	-63	-63	-86	-89	-96	-91	-87

that the number of APs observable in these three observations are all different. Some APs are in common at both places, while some are not. This is primary because some APs are too far away from one place. If the received power falls below the sensitivity threshold, the respective AP cannot be detected. This type of missing data is called MNAR. However, we also find that even for observations from the same place, measurements from APs may be missing (e.g., AP 1050, 1075 in observation 2). Such missing appears not due to low RSS (e.g., AP 1050) but rather at random, and we call them MAR. For WiFi scans, imputation that replaces missing data with a fixed value (e.g., -99dBm) is effective for MNAR but not for MAR.

4.3.2 Summary of solution approaches

Table 4.3 outlines the techniques we propose to address the above challenges in achieving energy-efficient inference of recurring and new indoor places. Figure 4.1 summarizes the flow diagram of LOIRE. In particular, we model RSS measurements from different places as Gaussian

Table 4.3 Summary of techniques.

Challenges	Solution approaches
RSS variations	Gaussian mixture model; large prior for variance
MAR	Interpolation in time
MNAR	Non-parametric categorical clustering (NCC)
Energy efficiency	Motion detection; Infinite Gaussian Mixture Model (IGMM) and Sequential Probability Ratio Test (SPRT)
Device heterogeneity	Signal strength difference (SSD) [48]

mixtures. To combat MAR and MNAR, we first group RSS measurements into coarse-grained clusters according to the list of APs observable using non-parametric categorical clustering (NCC). RSS measurements within one coarse-grained cluster share similar dimensions in the feature space. An online sampling procedure with stopping is then applied to the numerical RSS features to further group observations into fine-grained cluster each corresponding to one place. Users are optionally asked to provide the labels when new places are identified. Past observations and their labels are stored in the signature library. The signatures are updated when LOIRE misidentifies an old place as a new place. This may happen when RSS measurements at a place change significantly due to WLAN reconfiguration and/or environmental changes (e.g., furniture rearrangements). The rest of the chapter is organized as follow. A novel framework to resolve the problem of missing data is described in details in Section 4.4. Toward the energy efficient approach, we propose an online sampling algorithm for IGMM and a Quickest Detection stopping rule in Section 4.5. The algorithm is fully evaluated in Section 4.6. And finally, a conclusion is drawn in Section 4.7.

4.4 Handling Missing Data

Let $\boldsymbol{X} = [\vec{x}_1, \vec{x}_2, ..., \vec{x}_N]$ be a set of N observations. Each observation $\vec{x}_i$, also called a feature vector, has a variable length and is defined as

$$\vec{x}_i = [AP_1 : RSS_1,\ AP_2 : RSS_2, ...,\ AP_n : RSS_n]^T, \tag{4.1}$$

where n is the total number of unique APs observable.

Not all APs can be observed at all locations all the time. Thus, $\vec{x}_i$'s may have missing entries.

As discussed in Section 4.3.1, there are two types of missing data in RSS measurements, MAR and MNAR. MAR is caused by missing beacons from APs[1] and/or imperfection in WiFi scanning on mobile phones. MAR is more evident when many APs exist in the surroundings. MNAR, on the other hand, is caused by signal attenuation over large distances. Unlike MAR, which is undesirable, MNAR contains useful information for identifying the places. For example, two locations at two ends of a large building can see completely different sets of APs, while two adjacent locations observe the same set of APs but at different magnitudes. Therefore, we need to treat MAR and MNAR differently.

4.4.1 MAR

To mitigate MAR, we observe that RSS measurements are highly correlated at the same location within a short period. Thus, we can remove missing data by interpolation using the value of adjacent RSS readings from the same AP. More specifically, let $x_{i,j}$ be the predicted missing values in the ith measurement from AP_j. $x_{i,j}$ is calculated as:

$$x_{i,j} = \frac{\sum_{t=1}^{m} x_{i+t-m/2,j} o_{i+t-m/2,j} w_t}{\sum_{t=1}^{m} w_t o_{i+t-m/2,j}}, \tag{4.2}$$

where m is the number of the taps of the averaging filter; w_t is the coefficient of each tap and set to $1/m$; and $o_{j,t}$ is a binary indicator: it is 0 if data is missing, and 1 otherwise at time t.

A small value of m can preserve sudden changes in RSS. In the experiments, we choose $m = 2$, which means that interpolation is applied only for missing values that have at least one out of its 4 nearest samples observed. For MNARs, missing values typically appear consecutively, and thus are not interpolated and need to be treated separately.

4.4.2 MNAR

MNAR contains location-dependent information. However, the list of visible APs alone are not sufficient for fine-grained identification. Our measurement study shows that locations in the area of a building tend to share the same set of APs. Thus, The magnitude of RSS readings is crucial in differentiating the locations. The objective of handling MNAR is to group observations sharing

[1]Transmissions of beacon messages are subject to CSMA, and therefore, may not be always carried out.

similar sets of APs together, which then allows further classification based on magnitude values as will be discussed in the next section.

Since observations are of variable length, the Jaccard distance is used as a measure for the dissimilarity between two observations defined below,

$$J(\vec{x}_i, \vec{x}_j) = 1 - \frac{\vec{x}_i \cap \vec{x}_j}{\vec{x}_i \cup \vec{x}_j}, \tag{4.3}$$

where $\cap$ represents the set intersection and $\cup$ stands for the set union. In other words, the Jaccard distance is calculated as the fraction of the number of disjoint APs over the number of all available APs in the two observations. For example, the Jaccard distances between the first two and the last two observations in Table 4.2 are $4/7$ and $3/8$, respectively.

Given the Jaccard distance, we propose the non-parametric categorical clustering algorithm, a kernel density estimation approach based on the Mean Shift (MS) algorithm [49] to cluster the observations. It is non-parametric as no prior knowledge regarding the number of clusters is required; it uses categorical information about the set of APs. The basic idea is to use the pairwise Jaccard distance to build a density function of the observations, and then group the observations that have small distances to each other into one cluster. We use the following uniform kernel function to evaluate the density:

$$\kappa(\vec{x}) = \begin{cases} c(l - \vec{x}^T\vec{x}), & \text{if } \vec{x}^T\vec{x} \leq l, \\ 0, & \text{otherwise,} \end{cases} \tag{4.4}$$

where

$$c = \pi^{-D/2}\left(\frac{D+2}{2}\right)\Gamma\left(\frac{D+2}{2}\right), \tag{4.5}$$

and D is the dimension of the observations. The density function can be estimated as:

$$\hat{f}_{l,\kappa}(\vec{x}) = \frac{1}{Nl^D}\sum_{i=1}^{N}\kappa(\vec{x} - \vec{x}_i), \tag{4.6}$$

where l is the diameter in the Jaccard coordinate of a cube centered at $\vec{x}$, denoted as $C_{\vec{x}}$. Note that the argument of the kernel function is $(\vec{x} - \vec{x}_i)$, hence, the kernel covers all the observations within

the ball $C_{\vec{x}}$ that have the Jaccard distance to $\vec{x}$ smaller than l. Given the density function, the densest region should locate at the point where the gradient of the density function is 0. Let k be the number of observations in $C_{\vec{x}}$. The gradient of the density function is as follow,

$$\nabla \hat{f}_{l,\kappa}(\vec{x}) = \left(\frac{D+2}{l^2} \frac{k}{N v_l} \right) \left[\frac{1}{k} \sum_{\vec{x}_i \in C_{\vec{x}}} (\vec{x}_i - \vec{x}) \right], \tag{4.7}$$

where

$$v_l = \frac{l^D \pi^{D/2}}{\Gamma[(D+2)/2]}. \tag{4.8}$$

(4.7) is a product of two components. The first component is proportional to the density function at point $\vec{x}$, and the second component is called the *mean shift vector*, $\vec{m}$. $\vec{m}$ is the difference between the arithmetic mean, $\frac{1}{k} \sum_{\vec{x}_i \in C_{\vec{x}}} \vec{x}_i$ and the center of the kernel, $\vec{x}$. This mean shift vector points toward the local maxima in the density function. In other words, if one replaces the center of the kernel with the arithmetic mean, one reaches the local densest region with gradient 0.

Since the observations are categorical (namely, consisting of the collection of access points), the arithmetic mean cannot be computed directly. Instead, we resort to the pair-wise Jaccard distance and approximate the arithmetic mean with the geometric median defined as follows:

$$g = \arg\min_{\vec{x}_j} \sum_{i=1}^{N} J(\vec{x}_i, \vec{x}_j). \tag{4.9}$$

In other words, the geometric mean is the observation that has the smallest sum distances to all the other observations. If the observations follow the Gaussian distribution, the geometric median is identical to the arithmetic mean for a large number of observations.

The algorithm starts from an arbitrary observation, and groups all the observations within the coverage of a ball with diameter l and centered at the chosen observation to form a cluster. Next, it moves to the geometric median of the cluster, and forms another ball with diameter l until the geometric median does not change between iterations. This process is repeated for all observations. At the end of the procedure, the number of clusters will be determined as the number of the means found. Observations associated with each mean form one cluster.

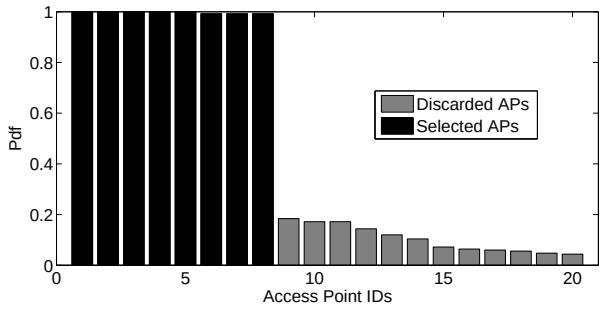


Figure 4.2 Histogram of APs in the scan result.

After NCC, observations in the same cluster share similar lists of APs. We further perform feature selection and choose APs that appear most frequently (e.g. more than 95% of the time). Figure 4.2 shows the frequency of appearance of different access points in one cluster. We observe that there exists a dichotomy – either an AP appears close to 100% of the time or way below 50%.

To this end, we mitigate the effect of missing data by i) imputation via interpolation, ii) feature selection, and iii) grouping observations using NCC. The first two techniques target at MAR and the third one at MNAR. Observations in the same cluster have feature vectors of the same length and can then be compared numerically.

4.5 Online Identification of New and Recurring Indoor Places

As discussed in the previous chapter, NCC provides coarse-grained clustering, which may not be suitable for place identification. Further fined-grained classification based on the magnitude of RSS is needed. In this section, we limit the discussion to observations from the same coarse-grained cluster as the result of NCC.

As discussed in Section 4.3, RSS measurements may vary a lot over time. The RSS measurements at a location can be modeled as a Gaussian distribution with unknown mean and variance. The Gaussian distribution assumption has been adopted in many other works [50]. When the observations are drawn from multiple places, they follow a Gaussian mixture with unknown number of components as the number of places is not known. The IGMM [10] is a generative stochastic model that can adapt model complexity (e.g. the number of components) to the data available. It is suitable for Bayesian inferences when the number of clusters are unknown and the clusters may overlap in

the feature space. In this section, we first give an overview of IGMM. Due to the high computation complexity, traditional inference model for IGMM is not suitable for running on mobile devices. Hence, we extend the framework to the online setting in the next section, where incoming observations are classified on the fly and decisions regarding whether the observations come from a new place or one of the old places are made once a sufficient number of observations have been collected.

4.5.1 IGMM

In this section, we briefly discuss about IGMM and its interpretation in the context of indoor locations. Formally, the definition of IGMM is given below.

Definition 4.1. *IGMM is a Gaussian mixture model associated with the following parameters,*

$$\vec{w}|\alpha \sim \textit{Stick}(\alpha); z_i|\vec{w} \sim \textit{Multinomial}(\cdot|\vec{w}); \vec{\theta}_k \sim \vec{H},$$
$$\vec{x}_i|(z_i = k, \Sigma_k, \vec{\mu}_k) \sim N(\cdot|\vec{\mu}_k, \Sigma_k), \tag{4.10}$$

where $\vec{\theta}_j \sim \vec{H}$ *stands for*

$$\Sigma_k \sim \textit{ Inverse Wishart}_{v0}(\Lambda_0); \vec{\mu}_k \sim N(\vec{\mu}_0, \Sigma_k/K_0), \tag{4.11}$$

and $\vec{w}|\alpha \sim \textit{Stick}(\alpha)$ *is a shorthand of:*

$$w'_k|\alpha \sim Beta(1, \alpha); w_k = w'_k \prod_{l=1}^{K-1}(1 - w'_k).K \to \infty. \tag{4.12}$$

$\alpha, \vec{H}, \vec{T}, \kappa_0, \vec{\mu}_0, \nu_0, \Lambda_0$ are called hyper parameters. The $Stick(\alpha)$ is a *stick breaking process* [4] used to generate an infinite number of weights, $\vec{w}$, that sum up to 1. $\vec{w}$ is the mixing weight vector, which defines the contributions of each distribution in the observations set. $z_i = k$ indicates that $\vec{x}_i$ belongs to cluster k with probability $p(z_i = k) = w_k$. The observations are assumed to follow the k^{th} joint Gaussian distribution that can be represented by $\vec{\theta}_k$, a parameter set of a cluster. Specifically, $\vec{\theta}_k$ includes a set of mean vector and covariance matrix for each cluster. The mean vector and the covariance matrix in turn are assumed to follow a Gaussian distribution and an Inverse Wishart distribution, respectively.

4.5.2 Online sampling with stopping

Traditionally, a Markov Chain Monte Carlo (MCMC) method such as the Gibbs sampling [5] can be adopted to sample labels for the observations in IGMM. The Gibbs sampling procedure for IGMM iterates through all samples and assigns labels for each sample. The computation complexity grows with the sample size. Thus, it is not suitable for online use when samples are classified as they come in. Furthermore, the Gibbs sampling procedure ignores the fact that the batch of samples associated with the same location should have the same label.

Sequential hypothesis testing [51] is a framework for hypothesis test where the sample size is not fixed in advance. Instead, data are evaluated as they are collected, and further sampling is stopped in accordance with a pre-defined stopping rule. The existing algorithms for sequential hypothesis testing are not directly applicable when there are multiple hypotheses – a new place or one of the old places. It further requires that both hypotheses have known distribution. Let $p_j(\vec{x}_i)$ be the *posterior probability* of the ith WiFi session $\vec{x}_i$ belong to the jth place, $j = 1, 2, \ldots, K, K+1$. $K+1$ represents the new place. To apply the sequential probability ratio test (SPRT), we consider only two hypotheses, a new place and the most likely old place. More specifically, the first hypothesis (H_1) associates the new observation with a new place with probability $f_1(\vec{x}_i) = p_{K+1}(\vec{x}_i)$; The null ($H_0$) hypothesis is associated with the distribution with the maximum posterior probability $f_0(\vec{x}_i)$ =max$\{p_k(\vec{x}_i)\}_{1:K}$, i.e., the most likely old place that the new observation belongs to. The cumulative log posterior probability ratio is computed as:

$$C = \sum_{i=1}^{N} \log \frac{f_1(\vec{x}_i)}{f_0(\vec{x}_i)}, \tag{4.13}$$

with N is the number of observations. When C exceeds a threshold $c_+ > 0$, a decision is made that the new observations belong to a new place. On the other hand, when C is below the threshold c_-, a decision is made that the new observations come from one of the old places that maximizes the cumulative log posterior probability:

$$k^* = \arg\max_k \left\{ \sum_{i=1}^{N} \log[p_k(\vec{x}_i)] \right\}_{1:K}. \tag{4.14}$$

WiFi scanning is stopped as soon as the threshold is exceeded and the decision is made. $c-$ and $c+$ determines the trade-off between misclassification errors and classification delay. We do not yet have a closed-form formula for determining $c-$ and $c+$ given the misclassification error bounds, which will be part of future work.

What remains to be resolved is how to update the posterior probabilities $p_j(\vec{x}_i), j = 1, 2, \ldots, K, K+1$. Let $\boldsymbol{S}_{1:K}$ represent the set of all location signatures and K be the total number of signatures collected so far. Each place has a unique corresponding signature, and thus, the number of places is also K. Let L_k be the number of observations stored for place k. Define the number of stored observations in all location signatures M, namely $M = \sum_{k=1}^{K} L_k$. Let $\vec{s}_{jk}$ represent the j^{th} observations from location k, and $\{x_i\}_{1:N}$ be the set of N newly collected observations. The posterior probabilities $p_j(\vec{x}_i)$ can be computed from the IGMM generative model in Definition 4.1 as below.

- **Known places:** The probability of identifying observation $\vec{x}_i$ belongs to a *known location* k can be determined below:

$$P(z_i = k|\vec{x}_i, \boldsymbol{S}_{1:K}; \alpha, \vec{H}) \sim \frac{n_k}{\alpha + M}$$
$$\times t_{v_k - D+1}\left[\vec{x}_i|\vec{\mu}_k, \frac{\Lambda_k(\kappa_k + 1)}{\kappa_k(\upsilon_k - D + 1)}\right]. \tag{4.15}$$

- **A new place:** The probability of assigning the observation to a new place, $K+1$ is as follows:

$$P(z_i = K + 1|\vec{x}_i, \boldsymbol{S}_{1:K}; \alpha, \vec{H}) \sim \frac{\alpha}{\alpha + M}$$
$$\times t_{v_0 - D+1}\left[\vec{x}_i|\vec{\mu}_0, \frac{\Lambda_0(\kappa_0 + 1)}{\kappa_0(\upsilon_0 - D + 1)}\right], \tag{4.16}$$

where

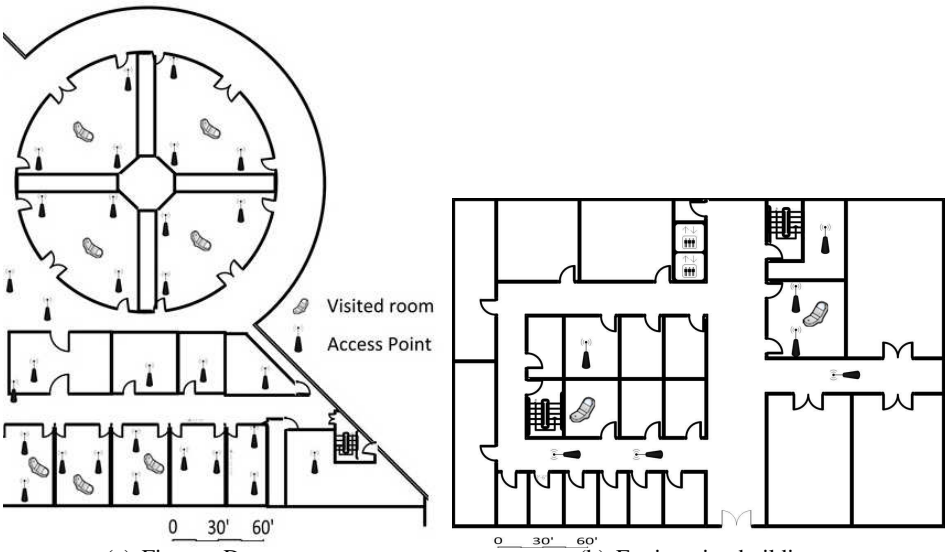
$$
\begin{aligned}
\vec{\mu}_k &= \frac{\kappa_0}{\kappa_0 + L_k}\vec{\mu}_0 + \frac{L_k}{\kappa_0 + L_k}\bar{\boldsymbol{S}}_k, \\
\kappa_k &= \kappa_0 + L_k,\ \nu_k = \nu_0 + L_k, \\
\Lambda_k &= \Lambda_0 + Sum + \frac{\kappa_0 L_k}{\kappa_0 + L_k}(\bar{\boldsymbol{S}}_k - \vec{\mu}_0)(\bar{\boldsymbol{S}}_k - \vec{\mu}_0)^T, \\
\bar{\boldsymbol{S}}_k &= (\vec{s}_{1k} + \vec{s}_{2k} + \cdots + \vec{s}_{L_k k})/L_k, \text{and} \\
Sum &= \sum_{j=1}^{j=L_k} (\vec{s}_{jk} - \bar{\boldsymbol{S}}_k)(\vec{s}_{jk} - \bar{\boldsymbol{S}}_k)^T. \qquad (4.17)
\end{aligned}
$$

Based on the above two posteriors, we sample the label for each observation on the fly. Initially, the probability of assigning an observation to a new place entirely depends on the hyper-parameters as in (4.16). As more observations become available and assigned to a new place (proportionally to the posterior probability), the parameters associated with the $K + 1^{th}$ cluster will be updated with (4.17) reflecting the distribution of the new data. The main advantage of this approach is as follows. In contrast to maximum posterior assignment, with sampling, there is always a chance for an observation to be assigned to a hypothesis associated with a lower probability. The observation is used to update the parameters, significantly boosting the chance of assigning subsequent observation to the true label. To this end, we summarize the proposed approach for online identification of new and revisited places in Algorithm 4.1.

Algorithm 4.1 Online identification algorithm

input : A set of newly observed data $\{x_i\}_{1:N}$, a set of location fingerprints $\{\boldsymbol{S}_k\}_{1:K}$.
output: Decision about the location the observations belong to. The location can be one of the identified locations or a new place.

Initialize:
- Set a threshold c_+, c_- to make the decision - C = 0 - i = 1 **begin**
  Utilize NCC to determine a small set of clusters, $\{\boldsymbol{S}_k\}_{1:K}$, that the observations may belong to The next steps involve with the above clusters only **while** $c_+ > C > c_-$ **do**
    Sample label z_i for $\vec{x}_i$ using (4.15) and (4.16) Update the parameters using (4.17) Calculate C according to (4.13) given the updated parameters i = i+1
  if $C \geq c+$ **then**
    Create a new signature for location $\boldsymbol{S}_{K+1}$ Label the new observations with $K+1$ Stop collecting WiFi signals
  else
    Identify the location k^* using MAP in (4.14) Stop collecting WiFi signals



(a) Finance Department (b) Engineering building

Figure 4.3 Floor plans.

Table 4.4 Summary of parameters.

N (Number of samples/place)	80
c_+ (SPRT upper threshold)	100
c_- (SPRT lower threshold)	-100
l (NCC threshold)	0.58

4.6 Evaluation

In this section, the performance of LOIRE is evaluated using experimental traces from smart phone devices. The values of the parameters used in the evaluation process is summarized in Table 4.4.

4.6.1 Experimental setup

Data collection Traces have been collected for two weeks from four Android phones: the HTC Nexus One, the HTC Sense, the Samsung Galaxy Nexus and the Samsung Nexus S. Experiments were conducted within two buildings in the University of Houston, the Engineering building (EB) and the Finance building (FB). Figure 4.3 shows the floor plans of the two as well as the AP layout and locations visited by the users (marked by "V"). Three students, two from the ECE department and the other one is from the Finance department, participate in the experiments. The two ECE students are PhD candidates, who spend most of their time in their lab and carry one phone each. The Finance student is undergraduate student and he carries the other two phones. Altogether, nine

locations have been visited and 70 visits were made to the locations. Of the seven locations in the FB, four are auditoriums while the remaining three are small classrooms. The labs and the classrooms in the EB are of roughly the same size as the classrooms in the FB.

Evaluation methodology In determining the accuracy of LOIRE in identifying the places, we randomly permute the order of the visits and evaluate the *hit rate* of the last visits, defined as the percentage of time the correct old place or a new place is identified. Other than the accuracy, we also define two missing probabilities, the first one is the missing in identifying an old place as a new place, and the second one is the missing in identifying an old place as another old place. When coming to a new place, N samples are collected. When returning to an old place, if the algorithm misidentifies it as a new place (based on the user feedback), N new samples are collected for future use.

4.6.2 Motion detection

In the experiments, we collect data corresponding to 5 different actions when a user is walking. This includes a phone held in the user's hand, pocket, and backpack, and being used for calls or reading news. In addition, accelerometer data is collected in 5 stationary states, i.e., the phone is put on the desk, in the pocket when the user is sitting and standing, in hand for reading, and in conversation. The data is divided into training and testing data. The precision of different algorithms in classifying whether a user is stationary or not is summarized in Table 4.5. Of the algorithms, Perceptron [52] consistently outperforms the other classifiers. Thus, we choose perceptron in subsequent experiments to determine the onset of WiFi scanning.

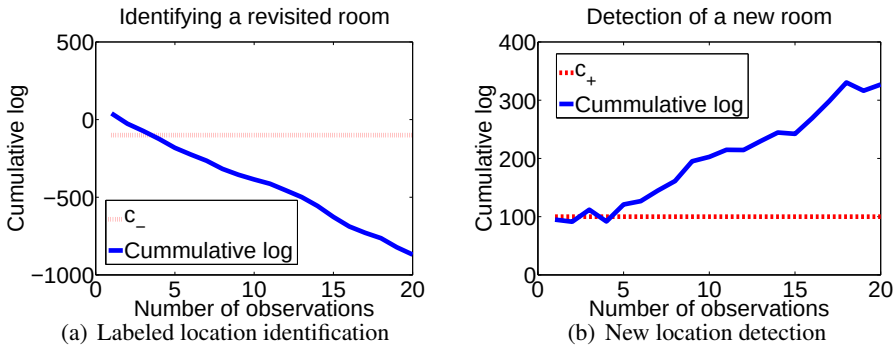
Table 4.5 Classification performance with different methods.

Naive Bayes	Decision tree	Perceptron	Boosting
95.5%	95.5%	96.7%	95.2%

4.6.3 Place identification

In this section, we evaluate the performance of LOIRE using traces collected from all the devices. We find that in the experiments the auditoriums in the FB and the two places in the EB have separable sets of APs. Therefore, NCC is sufficient to identify these places. However, the three small rooms in the FB share similar APs, in which case, fine-grained clustering is needed.

Choice of parameters Figure 4.4 shows the ratio of log posterior probability between the hypothesis that a user visits a new place and returns to an old place defined in (4.13). In particular, in



(a) Labeled location identification (b) New location detection

Figure 4.4 Sequential probability ratio test. The flat line indicates a threshold of $c_+ = -c_- = 100$.

Figure 4.4(a), as a user revisits an old place, C decreases quickly as more observations are made. In contrast, as shown in Figure 4.4(b), C increases (though not monotonically) as more samples are available when a user visits a place for the first time.

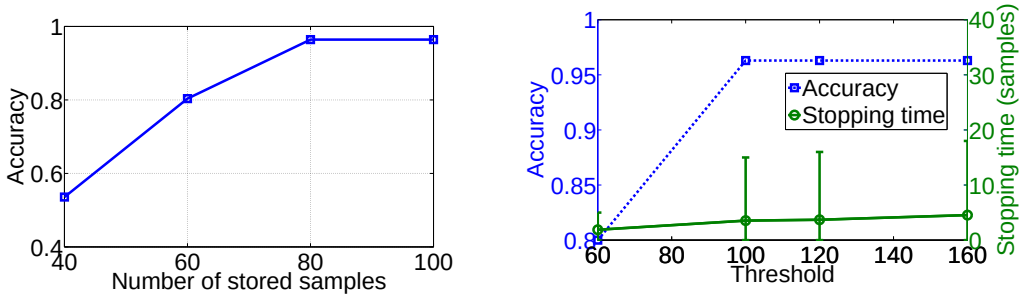
To find the number of samples to be stored for a new location, we vary N and show in Figure 4.5(a) the accuracy of LOIRE, defined as the average hit rate of identifying both new and old places. With more observations stored at each location, LOIRE has better characterization of the locations. One can see that for the data we have, 80 samples per location are sufficient to achieve the highest performance. For this reason, we choose $N = 80$.

Next, we vary the threshold value in the SPRT and evaluate its effect on the identification accuracy and stopping time. Here the accuracy is defined as the sum hit rates of identifying old and new places. It is always desirable to have the stopping time as small as possible by lowering the threshold. However, a smaller threshold would reduce the algorithm accuracy. In Figure 4.5(b), the stopping time is measured by the number of newly collected samples when C crosses the thresholds, where decisions can be made. Information about the confidence interval of the stopping time at each threshold is also given. For examples, at $c_+ = 160$, the stopping time ranges from 1 to 18. One can see that a threshold $c_+ = -c_- = 100$ can achieve a high accuracy while maintaining a reasonable stopping time (approximately 3.5 samples on average). To this end, from the experiments, we fix $N = 80$ and $c_+ = -c_- = 100$.

Under the above settings, LOIRE can achieve the accuracy of 92.2% in identifying revisited zones and 100% hit rate in detecting new zones. Of the 7.8% miss-classification of a revised place, 7.2% comes from incorrectly identifying a revisited place as a new place, and only 0.6% is due to incorrectly identifying it as another revisited place. Since users are promoted to enter its label when a new place is encountered, the first source of errors can be mitigated.

Table 4.6 Performance Comparison among LOIRE, LOIRE-U, and ARIEL

	Revisited zones			New zones
	Accu-racy	New	Different zone	Accuracy
ARIEL (400 samples)	84.6%	15.4%	0%	100%
LOIRE-U (80 samples)	87%	13%	0%	100%
LOIRE (80 samples)	92.2%	7.2%	0.6%	100%



(a) Accuracy vs N.

(b) Accuracy and stopping time vs $c_+(-c_-)$

Figure 4.5 Impact of parameter settings. Accuracy is the sum hit rate of correctly identifying a new and a revisited place.

We also evaluate the performance of LOIRE-U (unsupervised LOIRE) with the same settings as above, except that users' feedback are not utilized. As the results, LOIRE-U scores 87% hit rate in identifying revisited zones and 100% hit rate in detecting new zones. All missed classifications are from determining the revisited zone as a new zone.

For comparison, the zone detection part of ARIEL [45] has been implemented and tested it with our experimental data. We tune the parameters of ARIEL to achieve the best results. Specifically, we choose the minimum number of points $MinPts = 1$ and the distance threshold $Eps = 0.09$. The number of WiFi scans per session is set to be 400. Under these settings, ARIEL achieves 84.6% in identifying revisited place and 100% hit rate in detecting new places. All miss-classifications come from identifying a revisited place as a new place. Notice that this is the result achieved with 400 samples/session, while LOIRE-U/LOIRE can achieve comparable hit rate with approximately 3.5 samples/session and 80 samples/zone. Table 4.6 summarizes the performance of LOIRE and ARIEL on the same data set.

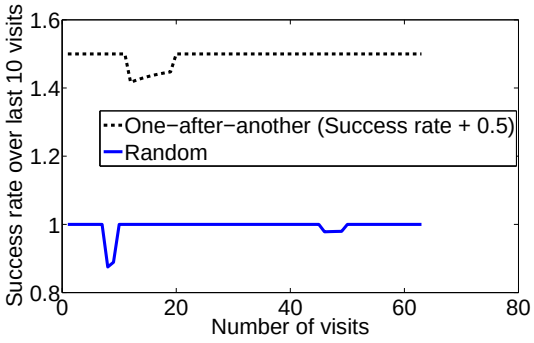


Figure 4.6 Success rate to correctly identify places over last 10 visits.

Cross-validation using multiple phones We next conduct experiments using the traces from two co-located phones. Figure 4.6 shows the percentage of correctly identified places among last 10 visits as more visits are made to different places. To evaluate the impact of different phones, we consider two setups. In the first setting (*random*), measurements from the two phones interleave randomly. For example, we may have [phone 1, place 1] [phone 2, place 2], [phone 1, place 2], etc. This emulates the real world scenarios where measurements from different phones may be reported from different users at different times from different places. In the second setting (*one-after-another*), measurements from the two phones follow one another. In the Figure 4.6, to distinguish the two curves, 0.5 is added to the success rate for the one-after-another setting. We observe that in both cases, as more visits are made, the success rates approach 1.

4.6.4 Energy saving

To demonstrate how much energy saving can be achieved by stopping WiFi scans and turning off a WiFi device, we measure the power consumption on each phone using the Monsoon power monitor. Power consumption is measured in 4 operating modes: both WiFi and accelerometer are turned off, only accelerometer is turned on, only WiFi is turned on and both are turned on. Table 4.7 summarizes the power consumption.

As one can observe, the accelerometer does not consume much power, approximately $1mw$. On average, the power consumption for WiFi scanning is $126.2mW$. While LOIRE only requires 3.5 and 80 samples when the user visits an old or a new place, respectively, ARIEL [45] needs approximately 30 and 400 samples to achieve a comparable performance in the two cases, respectively. This translates to an energy saving of $0.327J$ and $4J$ per place at the WiFi beacon rate of $10Hz$ with LOIRE.

Table 4.7 Power consumption for different phone models (mw).

	No sensor	Acc	WiFi	Both
HTC Sense	414.9	415	587.2	588.3
Galaxy Nexus	636.5	636.14	686.4	689.8
Nexus S	489.3	491.77	647.89	650.99

4.7 Conclusions

In this chapter, we proposed a lightweight energy-efficient solution to identify new and revisited indoor locations based on WiFi scans and accelerometer readings. Using traces collected from four smart phones in two buildings, we demonstrated that LOIRE could handle missing and noisy data effectively, without manual survey or prior knowledge of the number of places. Two case studies were conducted showing the utility of LOIRE. As future work, we plan to develop applications built on LOIRE, such as the micro-climate control and the co-location tracking, and may integrate a future whereabout location prediction algorithm into LOIRE.

Chapter 5

Predict Users' Future Whereabouts: Opportunistic Data Transfer in Cellular Networks

The number of mobile Internet users is growing rapidly, as well as the capability of mobile Internet devices. As a result, the enormous amount of traffic generated everyday on mobile Internet is pushing cellular services to their limits. We see great potential in the idea of scheduling the transmission of delay tolerant data, which are not required to be sent immediately, to times when the RSS is high. However, such scheduling requires good network condition prediction, which has not been effectively tackled in previous research. In this chapter, we propose a Dynamic Hidden Markov Model (DHMM) to model the time dependent and location dependent network conditions observed by individual users. The model is dynamic since transition matrix and states are updated when new observations are available. On the other hand, it has all the properties of a Hidden Markov Model. DHMM can predict precisely the next state given the current state, hence can provide a good prediction of network condition. DHMM has two layers, the top layer is RSS and the bottom layer consists of states, defined as a mixture of location, time and the signal strength itself. Since the state is defined as a mixture, it is hidden and the number of states is also not known a priori. Thus, the Nonparametric Bayesian Classification is applied to determine the hidden states. We show through simulations that when combined with a Markov decision process, the opportunistic scheduling can reduce transmission costs up to 50.34% compared to a naive approach.

Based on Cisco's estimate, mobile Internet traffic will grow to 6.3 exabyte (10^6 terabytes) in 2015 [53]. All wireless service providers are facing paramount challenges in serving the increasing traffic on cellular networks. However, *not all traffic is created equally*. While many applications, such as voice call and on-line gaming, demand real-time service, there exists an increasingly large number of applications that are delay tolerant times-scale, called *delay-tolerant* data. In addition, data from a large cellular network shows that there exists a significant lag between content generation and user-initiated upload time, more that 55% uploaded content on mobile network is at least 1 day old [54]. In addition to existing delay-tolerant data, service providers are also considering dynamic pricing to incentize certain applications and users to be more flexible in transmissions time. Such flexibility can be leveraged to improve network resource utilization through opportunistically schedules when network resource is abundant.

The delay tolerance of such jobs varies from subseconds to hours. For example, web browser

and Multimedia Messaging Service (MMS) messages can tolerate subseconds to seconds of delay. Emails can tolerate seconds to tens of seconds of delay. Content update, such as Facebook, Twitter, and Rich Site Summary feeds, can tolerate hundreds of seconds of delay. Content precaching and uploading can tolerate minutes to hours of delay. OS/application updates can endure even longer delay. However, current cellular networks more or less treat all traffic as equal, which results in performance degradation during peak load period.

The idea of opportunistic transmission schedule for delay-tolerant application is first explored in [55], where a simple histogram-based predictor is employed. The performance of the opportunistic transmission scheme clearly depends on how well the future network can be predicted. In this chapter, we develop a more elaborate model that better captures inherent user profile and provides more accurate estimation of future network condition. The proposed model is motivated by the predicability of human mobility patterns [56, 57]. Since network conditions location and time dependent, user experienced quality of service is predictable as well. However, building a user profile is inherently hard. Though the observations of network condition can be collected, we do not know how many states there are and which state (or cluster) the observations reside in. Even for a given location, network condition is dynamic due to small scale fading and network dynamics (e.g., load). Furthermore, there are distinctive modes in human mobility, e.g., human behaves totally different during weekdays and during weekends.

To address these challenges, we propose a dynamic hidden Markov model to model the user experienced network condition as the result of user mobility, available access networks, and time-varying nature of the wireless channel. A Nonparametric Bayesian Classification (NBC) method is used to determine the state space. Observations in the Dynamic Hidden Markov Model (DHMM) consist of three features: time, location and signal strength. Since users normally follows a stable schedule, their locations are closely correlated with time and duration they are at the locations. Moreover, even within the same coarse grained location such as in a building, their mobilities or the background noises have a strong correlation over time. As a result, signal strength also has a strong correlation with location and time. NBC can be applied efficiently in this scenario to locate closely related observations and group them into corresponding classes. Each cluster is associated with a state in our DHMM. Among many variations of NBC, the Infinite Gaussian Mixture Model (IGMM) is the most widely applicable model due to its robustness to many types of distribution as well as its high performance in classifying the observations. IGMM is a generative model describing the way the observations are created. With the DHMM, opportunistic scheduling decision can be

made based on the predicted network condition and requirements of the application.

Our contributions are summarized as follows:

1. We present a Dynamic Hidden Markov Model to capture user network condition profiles. The model is dynamic because the states and transition matrix are updated each time new observations are obtained. This is necessary to accommodate the situation when users move to new locations, we need to determine the states in the model to reflect the new change. If the state is not recognized, the DHMM simply does not operate since one requirement of DHMM is the complete knowledge of states.
2. We develop a non-parametric Bayesian approach to unsupervised classification with an unbounded number of mixtures to determine the state space of user profile. Specifically, we define a prior over the likelihood of devices using the Dirichlet distribution [4]. Based on the properties of the Dirichlet distribution, we derive a Gibbs sampling algorithm that can be used to sample from the posterior distribution, and determine the number of classes.
3. Based on the developed DHMM model, we apply the Markov Decision Process (MDP) to decide the optimal schedule of user transmission subject to a delay requirement.

It is worth mentioning that DHMM model is general and not limited to the opportunistic scheduling. Other potential applications include location-based advertisements where the payoff not only depends on location but also on other contextual information (such as preferences and past behavior). The effectiveness of the proposed method is validated using synthetic data and the results show great improvement over a naive method.

The rest of the chapter is organized as follow. In Section 5.1, we discuss related work. Opportunistic data transmission mechanism is described in Section 5.2. Section 5.3 introduces the DHMM system model. Section 5.4 explains why NBC fits the application. To evaluate the proposed framework, simulation results are presented in Section 5.5. And finally, conclusions are drawn in Section 5.6.

5.1 Related Works

The history of a user's network condition can be used to predict his own future network condition. In [56,57], the authors point out that individual human travel patterns are highly predictable. Since network condition experienced by a user, such as available networks and signal strength, is location dependent, it also has predictability. While network condition is predictable for individual users, it may differ across different users. Hence individual user profile is crucial to obtain a good

estimation of future network condition.

The idea of leveraging delay tolerant data in resource management has been studied in the literature. In [58], the authors exploit delay tolerant data for energy saving on mobile phones by choosing from different network interfaces (cellular, WiFi) if available. An online link selection algorithm based on the Lyapunov framework is proposed. In [59], the authors propose to use WiFi network whenever possible to offload data from 3G connections. They propose to use recent WiFi availability to predict the future availability. In both papers, simple first-order predictors that utilize past observations to predict future network connectivity or link transmission rate in near future.

The Markov (or hidden Markov) model is a powerful tool for prediction, and many recent works have employed it for the mobility prediction. In [60], the vehicular position location prediction is performed by the hidden Markov models (HMMs) trained with prediction data to model the strength of the received signals for particular areas. In [61], location context is used in the creation of a predictive model of users future movements based on the Markov models. In [62], by investigating the mobility problem in a cellular environment, a Markov model is constructed to predict future cells of a user.

Recently, there has been some works using nonparametric Bayesian classification. For example, in [63], the authors proposed a Nonparametric Bayesian Hidden Markov Model to build a static HMM for speech recognition, . In contrast, our work considers a dynamic HMM, and users' profiles are utilized to schedule effectively users' access. Several recent works apply the Bayesian nonparametric schemes in different applications. In [64], spectrum access in cognitive radio networks is modeled as a repeated auction game subject to monitoring and entry costs. A Bayesian nonparametric belief update scheme is devised based on the Dirichlet process. In [65], an approach is proposed that enables a number of cognitive radio devices which are observing the availability pattern of a number of primary users. The cognitive radios then cooperate and use the Bayesian nonparametric techniques to estimate the distributions of the primary users activity pattern, assumed to be completely unknown.

In this chapter, the proposed DHMM is different from other works in the following aspects. In DHMM, the states are not solely determined by the location. They are combinations of time, location, and signal strength. By collecting experiment data from wireless users, we observe that even within the same location, signal strength may still vary significantly. Thus, additional information other than the location information is necessary to achieve better signal prediction. Another important contribution lies in the dynamic nature of the model, the states and transition matrix

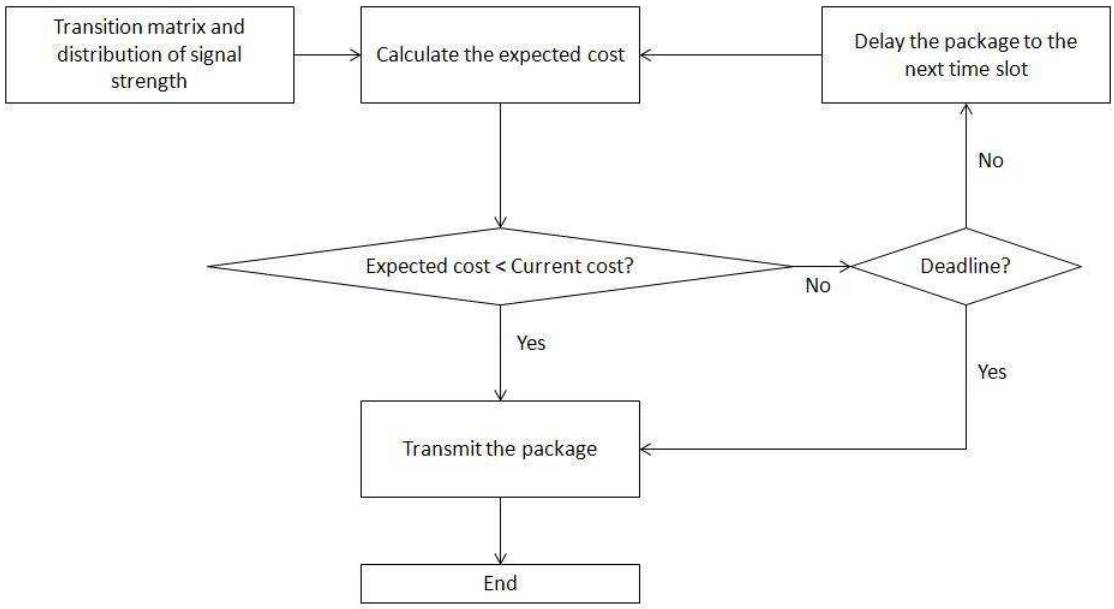


Figure 5.1 Decision making process.

are incrementally updated to reflect the recent trend of users. Finally, different from the previous schemes, in this chapter, we employ the nonparametric Bayesian approach to determine the states of the DHMM for scheduling transmission in wireless network.

5.2 Opportunistic Scheduling

In this section, we briefly discuss the mechanism to make scheduling decisions for an instance of data transmission given its deadline. We assume time is slotted. When delay tolerant data becomes available, the protocol is started at time slot 1. A deadline is specified by the application layer. The data has to be transmitted before the deadline, otherwise user experience will be harmed. In our framework, we call this deadline *horizon* M. The goal of the opportunistic scheduling is to minimize the network resource usage, while bounding the delay of data transmission. In each time slot, we make a decision whether to transmit right away or further delay the transmission based on the current network condition (e.g., RSS_i), time until deadline, and user network profile. If the deadline is imminent M, the data needs to be transmitted regardless of the network condition. Transmission at any time slot incurs an instantaneous network resource cost, which is a function of the network condition at that time, notated as L_i. The expected cost when using the optimal policy is C. The cost at a time slot i is defined as $1/RSS$ where RSS is the signal strength. The reward function is defined as the signal strength. Hence, the more the signal strength we gain by rescheduling, the more the reward we receive and the less the cost we have to pay. The notations

are summarized as follows:

- M: Horizon, deadline for data transmission.
- L_i: Network resource consumption (cost) for transmitting the job at time slot i, calculated based on RSS_i.
- $C_i(S_i)$: Expected network resource cost from time slot i on using an optimal policy given current state is S_i.

If a Markov model for RSS_i is given, we can use the standard MDP technique to decide the optimal transmission time. Because MDP is a mature and well-known technique, we only sketch the main steps here due to space limitation. At each step, the action at time slot i can be decided by

$$\text{Action} = \begin{cases} \text{Transmit}, & L_i \leq C_{i+1}(S_i), \\ \text{Delay}, & L_i > C_{i+1}(S_i). \end{cases} \tag{5.1}$$

The decision process is illustrated on Figure 5.1. To find $C_i(S_i)$ for each time slot, we use the backward induction. Starting from the last time slot, if we are at the time slot M, due to the deadline, we have to transmit regardless, the expected cost is the expectation of resource cost at that time slot is $E(RSS_M|RSS_i)$. In this process, it is critical to estimate future network conditions. To address this challenging issue, we propose the DHMM model and NBC approach in the subsequent sections.

5.3 Dynamic Hidden Markov Model

HMM is widely used in the literature for signal strength prediction. The states are normally defined as locations that are inherently limited as mentioned in Section 5.1. In our work, we base on the predictability of signal strength and its strong correlation with location and time to build our model, the DHMM.

Figure 5.2 illustrates the model. The bottom layer consists of the hidden states of the model, notated by S_i. Each state is defined as a multivariate joint distribution of time, location and signal strength. Although each state consists of a set of observations, it is still hidden because from the observations, we cannot differentiate which state generates the observations. The signal strength, location and time associated with a state are strongly correlated. Hence, they tend to be grouped together in a small cluster. User movement, including time of arrival and departure as well as visited locations, is predictable. As a result, the state, which comprises the two features, is predictable. In

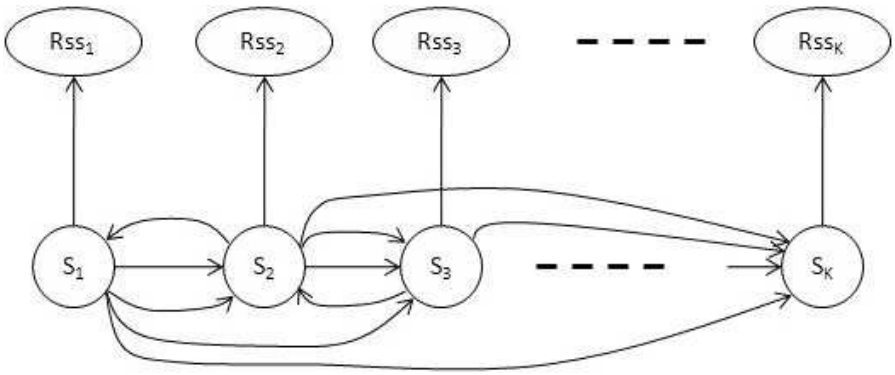


Figure 5.2 Dynamic hidden markov model

the model, states are associated with a cluster of observations, hence, the two concepts can be used exchangeably depending on the context.

The top layer is the RSS value, notated by RSS_i. Since each state generates its own distribution of RSS, prediction of the next state leads to the prediction of RSS following the respective distribution. In the context of scheduling user access, a distribution of the next possible states is input to a standard MDP to calculate the expected value of reward until the deadline. From the MDP result, if it is more beneficial to wait then the transmission will be delayed until we receive higher rewards. Otherwise, it will be transmitted immediately. In the next Section, we describe a procedure to determine the states of DHMM as well as the current state of an observation. We note that historical network conditions are readily available locally on mobile devices, and thus collecting this information incurs little extra cost.

At this point, the states are defined, however there are two questions left. The first question is how to determine the states and the number of the states given the observations. It is important to emphasize that the number of states or classes are not known, hence, traditional classification algorithms are not appropriate. And the second question is how to determine the current state of a user. Only by inferring about the current state can we calculate the transition probability and expected reward. The NBC is the best solution the questions since it does not require prior knowledge about the number of classes to be known as a priori.

5.4 IGMM

In an HMM, it is required that the number of states is known a priori. However, this assumption does not always hold in practice. For example, when a user comes to a new place, he creates a new state in the model and this new state is not known by training. Such a situation makes HMM not appropriate for modeling a user behavior. We, instead, apply IGMM, to dynamically determine

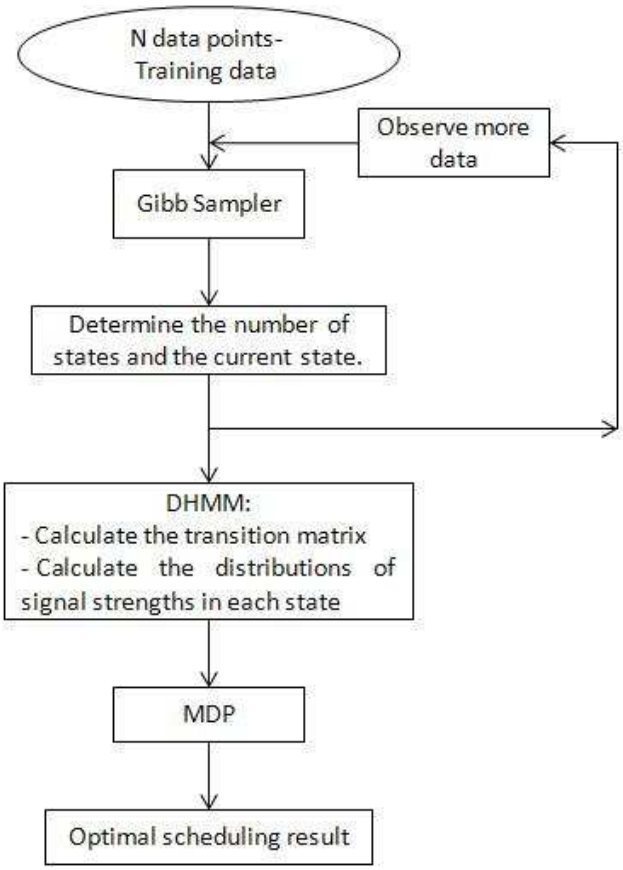


Figure 5.3 Opportunistic scheduling framework.

the states. Next, we briefly introduce IGMM highlighting its connection to the proposed DHMM.

5.4.1 Determining the States

Given the model and the observations, infer the model parameters using the inference algorithm described in Chapter 2. A Gibbs sampling algorithm [5] is implemented to sample the indicators from their posterior distributions. The indicators here in our application are the labels of the states. To this end, we are now in the position to outline the opportunistic scheduling based on DHMM as illustrated in Figure 5.3. First, the training data are processed, and states are determined by the Gibbs sampler. When a new observation is available, we run the Gibb sampler again to determine the observations' states. After identifying the states, the transition matrix will be updated. Given the current state and the updated transition matrix, the future reward will be calculated until the deadline. If the cost of transfer in the current state is greater than the expected cost in the future and the deadline has not been reached, we defer the transmission. Otherwise, data will be transmitted immediately.

As we can see, the states in DHMM are dynamically updated as more observations becomes

available. This property is a significant improvement since we are able to determine the current state associated with the observation and incorporate the new observation into the model when the user moves to a location that has never been recorded. Given the current states, one can easily update the transition matrix in the DHMM model to reflect the recent dynamics.

5.5 Simulation Results

To evaluate the proposed method, we generate synthetic data as follows. A random number of locations were selected, each location is specified by a two-tuple of longitude and latitude. Every day, the user follows a schedule and stays in each location for a Normal distributed random length of time. Totally, there are 14 locations. The longitude and latitude means for the locations are chosen randomly in the range of $[0, 14]$ with variance in each location set to 1. Signal strength mean in each location is selected randomly in the range of $[0, 30]$ and has a standard deviation of 1.55dBm. Deadline is chosen to be 20. After generating the synthetic data, we divide it into two parts, a training part and an online part. Transmission cost is defined as the inverse of the signal strength. Figure 5.4 shows the output of the Gibbs sampler. Since we can only show a 3D picture, latitude, time and RSS were chosen. Observations within a rectangle are from the same location while those within an eclipse are classified as one cluster or state by the Gibbs sampler. Although there are only 10 locations, Gibbs sampler outputs 14 states since at the same location, there can be more than one signal strength profiles over the time.

The Gibbs sampler has high classification accuracy and agility when new observations are present. As shown in Figure 5.4, although there are only three observations marked by "x", they are grouped together to form a new state. Thus, whenever a user enters a new new location, the DHMM will be immediately updated with the new state.

Each scenario is run 800 times using traces from the online part of the synthetic profile to make the results statistically meaningful. Performance of the DHMM is compared with that of two other methods. The first method is a naive approach, which simply transmits data immediately regardless of future rewards. The second method is the UPDATE algorithm in [55]. Every time, all three algorithms are used to schedule the transmission on the same synthetic trace, and the total transmission cost of each test is collected. In Figure 5.5, we show the Cumulative Distribution Function (CDF) of the transmission costs in all the tests run. CDF is used to show the distribution of transmission cost among all the tests. An average reduction of 50.34% of network cost is achieved when applying the DHMM and MDP compared to the naive one. Our proposed framework also outperforms the UPDATE algorithm by 27.5%. As shown in the figure, over 93% of the transmis-

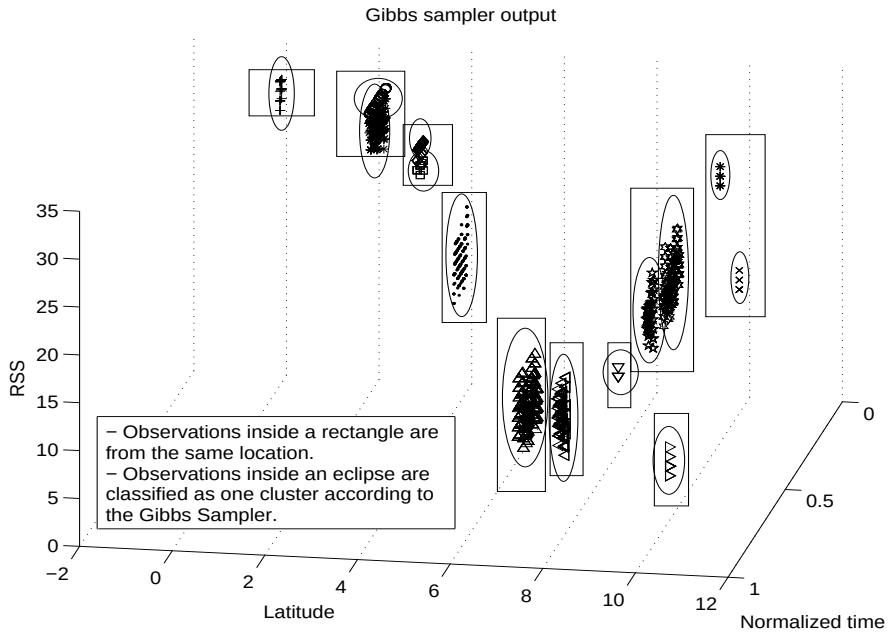


Figure 5.4 Classification result.

sions scheduled by DHMM have cost less than 0.04, while that rate of the UPDATE algorithm is 71% and the naive method is even worse with 52% of the transmissions have cost less than 0.04.

5.6 Conclusions

When the number of wireless smart devices increases rapidly, network load also increases exponentially since data transfers are bigger in both size and quantity. However, not all traffic are created equally. In this chapter, we propose to exploit the delay-tolerance of certain data transfers to opportunistically schedule such jobs to times when the network condition is more favorable. To achieve this goal, we proposed an DHMM model and the NBC approach to build an adaptive model of user profiles based on observations. Through simulations, we showed that the proposed scheme improves over the naive method by 50.34%. The results presented in this chapter demonstrated the promises of exploiting user and application characteristics in better resource management in cellular data networks. As future work, we plan to collect real-world measurement data from mobile users to test the proposed framework.

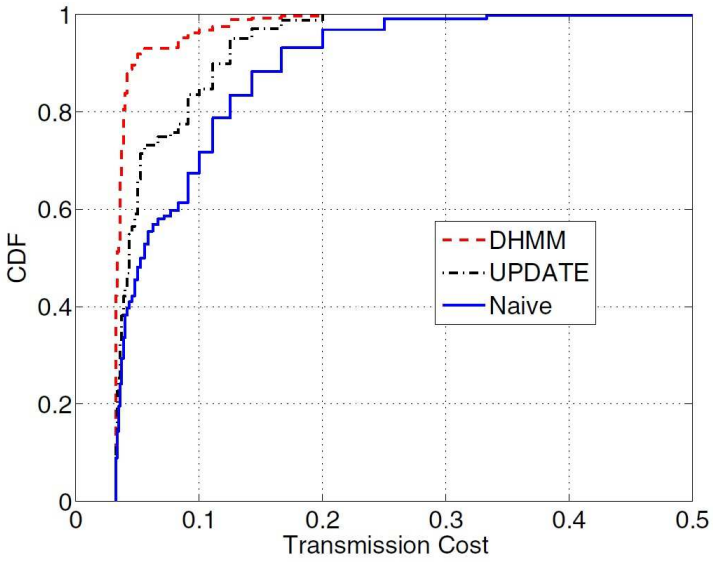


Figure 5.5 CDFs of transmission cost for DHMM, UPDATE and naive methods.

Chapter 6

Predict Users' Future Whereabouts: Extracting Typical Users' Moving Patterns Using Deep Learning

When GPS devices are widely integrated into smart phones, researchers stand a big chance of collecting massive location information, that is necessary in studying users' moving behavior and predicting the next location of the users. Once the next location of a user can be determined, it can serve as input for many applications, such as location based service, scheduling users access in a mobile network or even home automation. One important task in predicting the next location is to identify typical users' moving patterns. In this chapter, we propose a novel method to extract the patterns using deep learning. Experiment results show significant performance improvement of the proposed method compared to the classical principal component analysis method.

6.1 Introduction

Predicting user movement has been always an interesting topic for researchers in the past decades. However, due to the lack of smart hand held devices, it has never surged until smart phone with integrated GPS was invented recently. There are many works which focus on location prediction, however, most of them are based on the Markov model [66] [67] [68]. The Markov model is powerful, but it poses some problems such as expressing different patterns over different times or exploiting embedded frequencies in human behavioral patterns [2]. For example, at 6am during the weekdays, an officer normally goes to his office, but on the weekend, at the same time, he more likely stays at home. Markov model treats these two situations in the same way, the probability of going to office when the user is at home at 6am during the weekends is as high as that probability during the weekdays. Hence, if typical users' moving patterns can be extracted, they will be very useful by combining with the Markov model with different moving patterns for different days. The proposed approach can also serve as a stand-alone solution if we want to predict the next possible visiting locations in the rest of the day given location information from midnight to noon [69].

Existing efforts in identifying users' moving patterns can be divided into two major approaches. The first approach is from frequentist statistics [70] [71], in which the most frequent moving sequences of visiting places are extracted. Nevertheless, this approach lacks flexibility. For example, when a user stops by a gas station on the way going from home to office, the user breaks the stored moving sequence, e.g. [Home $\rightarrow$ Office], and consequently, the system fails to predict the

next place. In addition to that, locations are only considered to be in the same sequence if and only if they are in the same time slot [71]. This condition does not fit the real scenarios, many factors can affect users' schedules.

The other approach is based on principle component analysis (PCA) to extract eigenbehaviors [69] or eigenplaces [72]. An eigenbehavior is a principle component of our daily moving patterns. In [69], three principle patterns were extracted, including weekday, weekend, and no-signal patterns using PCA. The authors then tried to predict future locations given the observed data and obtained promising experiment results. However, compared to deep learning, which has multiple layers, PCA has a huge disadvantage of shallow structure since it consists of only two layers, additionally, it assumes linear transformation. With multiple layers, and using non-linear transformation, deep learning is more powerful and flexible since it is able to combine many layers and complex mixtures to generate observations. In addition to that, only three eigenbehaviors might not be sufficient enough to construct different users' movement patterns. We believe that the results can be further improved by extracting more typical patterns and applying a "deeper" structure.

From the human point of view, given the users daily trajectories, we can tell approximately how many typical moving patterns there are. Our brain can extract multilayer representations of the trajectories, and at the highest abstract level it concludes a number of typical moving patterns. The question is how to reverse engineer the human brain which has multiple layers of expressions. Theoretical results suggest that we should use a deep structure to model this complicated extraction process [73] [74]. Nevertheless, there was not any closed form solution for inference in the deep learning due to the complicated structure and the lack of an efficient algorithm to set up initial weights until recently. Hinton et al. [75] proposed a method to pretrain the model layer by layer and set the prior to cancel out the explain away effect. The DBN proves to outperform shallow architectures such as PCA in many applications [75]. We distinguish our method from the others in the following aspects:

First, to the best of our knowledge, it is the first attempt to apply deep learning in extracting users' typical moving patterns. Since deep learning outperforms PCA by far, we expect to achieve a better result compared to the result in [69]. Second, in our approach, to extract the patterns, users do not have to manually label their logical places, which was a requirement in [69]. And finally, we apply deep learning to some collected users trajectories and prove that the deep learning can reconstruct the trajectories with much less error compared to the classical PCA.

The rest of the chapter is organized as follows. In Section 6.2, the process of collecting and

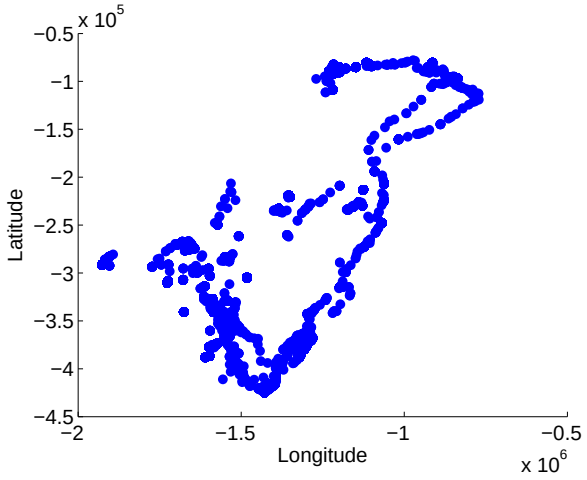


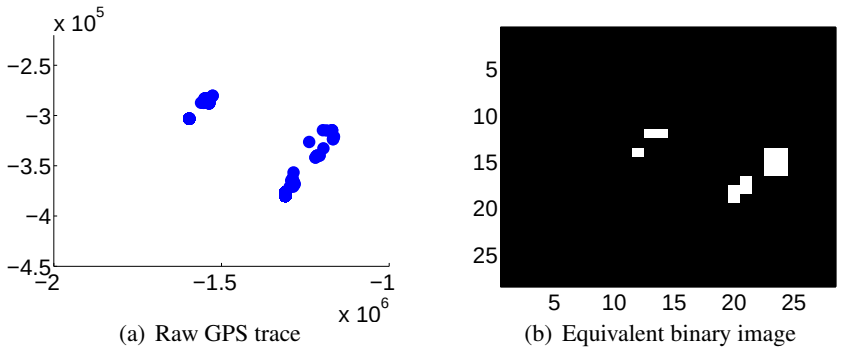
Figure 6.1 6-week trajectory.

representing the trajectories to feed to the input of the deep learning is described. In Section 6.3, we discuss about the deep learning and its basic components, the Restricted Boltzmann Machine (RBM). Experiment results are shown in Section 6.4, and finally, the conclusion is drawn in Section 6.5.

6.2 Data Collection

An Android application was written and installed on the HTC Wildfire S smart phones to collect GPS signal every 30 seconds. The data consists of GPS coordinates and recorded timestamps. A group of volunteers live in different cities were chosen to bring the phones on their normal daily routines. GPS is always turned on on the phone and locations are collected periodically when the phone is not connected to the wall charger. Data is then automatically uploaded to a dropbox account at the end of the day. A trajectory of a user collected in 38 days is shown on Figure 6.1.

Based on the timestamps, GPS data in the same day is grouped together to form a data set. The area that includes all the GPS measurements shown on Figure 6.1 is divided into 784 small cells, corresponding to a (28x28) pixels image. One example of a trajectory for a day is illustrated on Figure 6.2 where the raw trace on the left is converted to a binary image on the right. For now, we assume the image pixel values are binary. Within a day, if the user shows up at a specific cell, then the corresponding cell on the image is assigned a value of 1. Otherwise, the cell value remains at 0. Furthermore, the image is reshaped to create a 1x784 vector. This vector will serve as the input for the deep learning described in the next section.



(a) Raw GPS trace (b) Equivalent binary image

Figure 6.2 Converting raw trace to binary image.

6.3 Deep Learning Basics

Deep learning is a generative model that consists of multiple layers of hidden stochastic latent variables or features. In this chapter, we use a variant of deep learning called Deep Autoencoder (DAE) [75]. DAE stacks multiple layers of RBM, thus, in a few subsections below, RBM is first described and then the generative model of DAE is introduced.

6.3.1 Restricted Boltzmann Machine

The Boltzmann machine is a parallel computational model supports "weak" constrains [76]. Units in the Boltzmann machine are stochastic and have energy assigned to them. The RBM is a restricted version of the Boltzmann machine. It does not have any intralayer connection between the hidden nodes, which leads to possible inference algorithms. RBM is an undirected graph that consists of two layer: a hidden layer and a visible layer as well as one bias unit that is always on. Hidden nodes are not connected and are conditionally independent given the observations. This important property makes inference tractable in the RBM. The RBM is illustrated on Figure 6.3, where v is the visible unit or observation, h represents hidden units and b is the bias unit.

For illustration purpose, we assume that both the observations and the hidden nodes are binary. The activation energy of unit i is:

$$E_i = \sum_j w_{ij} x_j, \tag{6.1}$$

where the index of the sum runs over all units connected to hidden unit i, x_j represents the j^{th} observation and w_{ij} is the weight of the connection between the hidden unit and the observation unit. With activation energy E_i, the probability to turn unit i on is a logistic sigmoid function:

$$p_i = 1/[1 + \exp(-E_i)]. \tag{6.2}$$

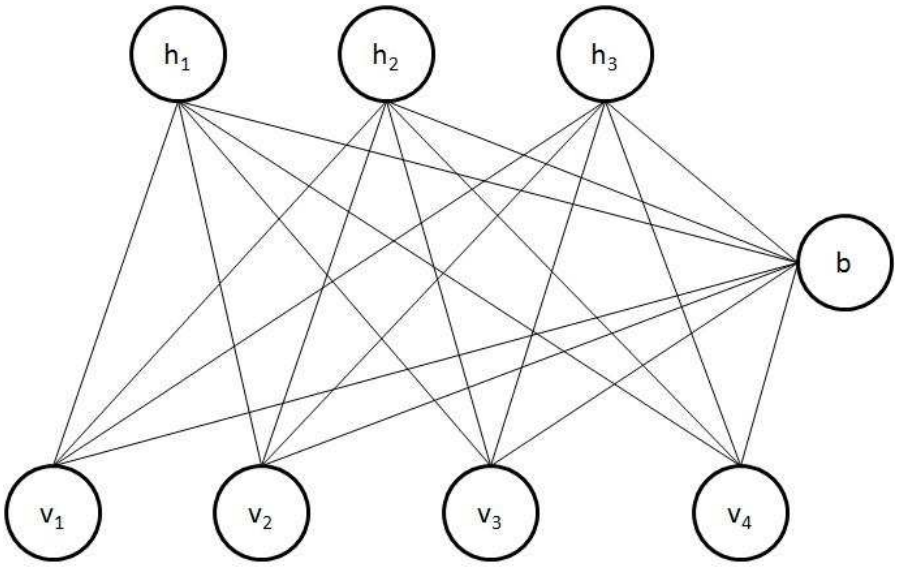


Figure 6.3 Restricted Boltzmann Machine

The larger the value of the activation energy, the more likely unit i will be turned on. A weight of a connection represents a mutual connection between the two units. For example, if both units are very likely to be on or off at the same time, then the weight between them should be positive and large. On the other hand, if they are very unlikely to have the same value then the weight should be small and negative. For example, in our application, we have 7 typical patterns, or equivalently, 7 hidden units on the top layer. The hidden units are corresponding to 7 days in a week and are the output of the algorithm. It is essential to mention that each hidden unit is not a trajectory. It is instead only one hidden node with an activation energy, which is proportional to the probability that the unit is turned on, and a set of related weights. Nevertheless, if the hidden unit is turned on, it can generate a full trajectory by following the weights and activation energies of the lower level. If we apply a trajectory collected on Monday to the visible unit, it is very likely that the Monday hidden unit at the top layer will be turned on and the weight of the connection between them is high.

If we name the vector of visible nodes **v** and the vector of hidden nodes **h**, then the energy of any joint configuration (**v**,**h**) is [77]:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in \mathbf{v}} a_i v_i - \sum_{j \in \mathbf{h}} b_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{6.3}$$

where a_i, b_i are the bias associated with visible units and hidden units, respectively, and w_{ij} is the weight between a hidden unit j and a visible unit i. The probability of any configuration (**v,h**) is:

$$p(\mathbf{v}, \mathbf{h}) = \frac{\exp[-E(\mathbf{v}, \mathbf{h})]}{Z}, \tag{6.4}$$

where Z is a normalizing factor to limit the probability in the range of $[0, 1]$. Given the join probability of the configuration, it is easy to find the marginal probability of a set of visible units:

$$p(\mathbf{v}) = \frac{\sum_h \exp[-E(\mathbf{v}, \mathbf{h})]}{Z}. \tag{6.5}$$

This probability represents the chance that the model with the determined values of weights will generate the observations set **v**.

In the training phase, it is expected that the probability is maximized by varying the weights. Thus, the optimum weights can be obtained by taking the derivative of the probability with respect to the weights:

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} =< vi.hj >_{data} - < vi.hj >_{model}, \tag{6.6}$$

where the brackets $< \ . \ >_{data}$ stands for the expectation of the product of the weights and the observed data and $< \ . \ >_{model}$ is the same expectation for the model observations that is generated according to the model. Obviously, when the derivative is 0, or in other words, when the training data and generated data are similar, we obtain the optimum performance. Hence, the learning algorithm is simply to update the weights with the value in (6.6).

$$\Delta w_{ij} = \epsilon(< vihj >_{data} - < vihj >_{model}), \tag{6.7}$$

where ϵ is a learning rate. The bigger the ϵ is, the faster the algorithm converges, but the coarser grain of the optimal values the algorithm can achieve. For this reason, ϵ should be chosen conservatively to achieve the best performance. Experiment results suggest that the value of ϵ should be set at 0.1.

6.3.2 A Deep Auto-Encoder for Typical Users' Moving Patterns Extraction

A DAE is an artificial neural network used to learn multilayer representations of an input data. Figure 6.4 describes the DAE for our application. It consists of 4 hidden layers. The lowest hidden layer has 500 hidden units, the next one has 400, the 3^{rd} layer has 300 and the top layer has 7 hidden units. Except for the top layer, all the other hidden layers are binary. 0 means the hidden unit is turned off while 1 means the hidden unit is turned on. The top layer has only 7 hidden units since human has 7 typical moving patterns corresponding to 7 days in a week.

Our purpose is to find the weights that minimize the error. However, optimization is over sensitive to initial chosen weights values. The performance is very poor if a large initial weights are chosen, while with small initial weights, optimization is intractable. Recently, Hinton et al. [75] have shown that by pretraining the network layer by layer, we can obtain a good initial weights val-

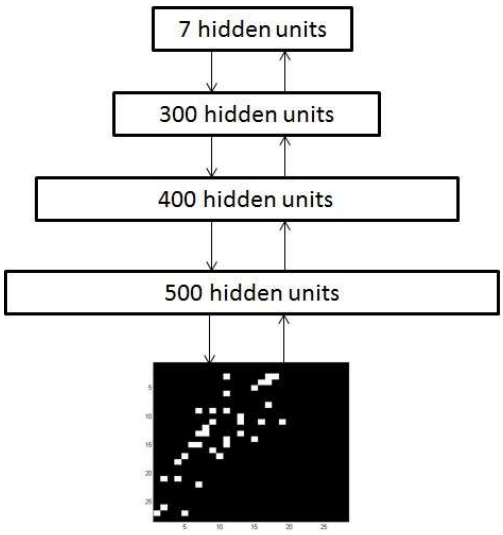


Figure 6.4 A Deep Auto-Encoder Network for a user trajectory.

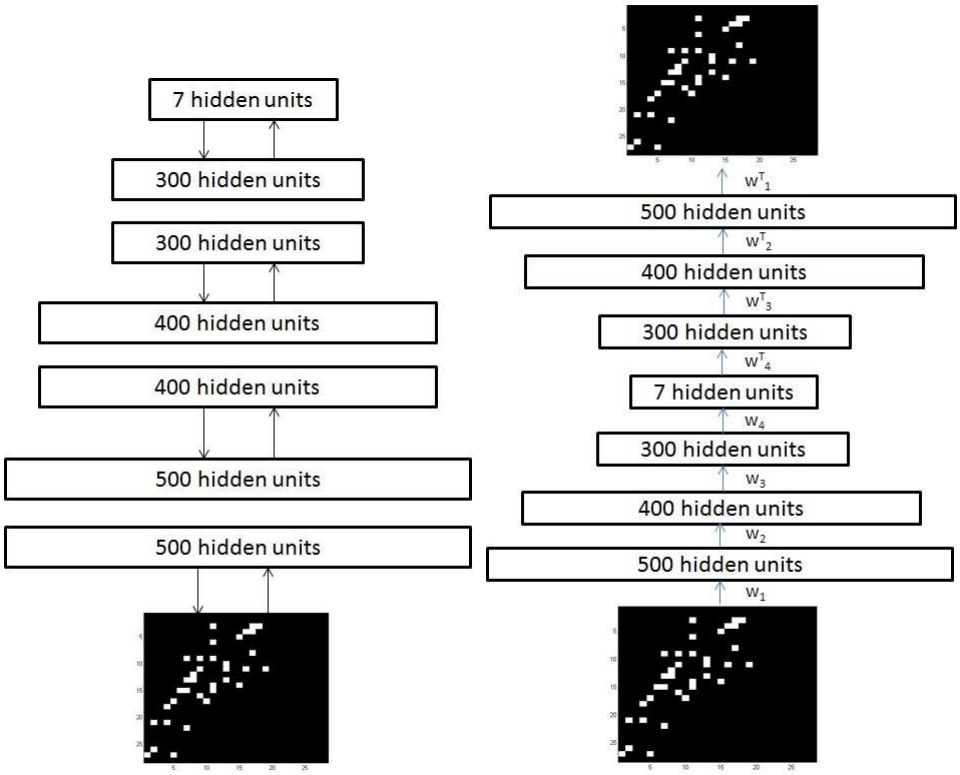
ues to minimize network errors. After that, a fine-tuning phase is applied to find the local maxima. In the next two subsections, we will describe shortly these two phases.

6.3.3 Pretraining Phase

Figure 6.5 shows the pretraining phase that include two steps. At the first step shown on Figure 6.5(a), for each layer, the hidden units are separated from their upper level hidden units and connected to the lower level hidden units to form an RBM. Hidden units in every RBM are binary except for the top layer, where the hidden units are drawn from a Gaussian distribution with the mean equals to the activation probability. The binary input of the trajectories is fed to input of the first RBM. As described in the previous section, the activation probability can be calculated according to (6.2). In turn, the hidden units of the first RBM now become visible units in the second RBM. The same process is repeated until the activation probabilities of the top layer hidden units are determined.

The step described above is a bottom to up process to achieve activation probabilities for all the hidden units. Now, given the probabilities and the weights, we implement the second step, shown on Figure 6.5(b). In this step, a process from top to bottom is followed to generate the model "visible" units at the bottom layer. The difference between the model "visible" data and the actual data gives us a hint to update the weights according to (6.7). On Figure 6.5(b), the top trajectory is created by the model while the bottom trajectory is the training data.

After this step, we obtain coarse grain optimum values for the weights. To further improve the result, a fine-tuning process is implemented using backpropagation in the next subsection.



(a) Training for each RBM (b) Unrolling the network to generate model data

Figure 6.5 Pretraining phase.

6.3.4 Fine-tuning Phase

Backpropagation is a supervised method to train deep learning. Since the top hidden units have been learned from the above pretraining phase, we first forward the training data to the input of the network and calculate the difference between the inferred top-layer hidden unit and the learned top-layer hidden. The difference at the top layer is then fed backward to go through the network from the top to the bottom layer using the leaned weights, producing differences at every layer. The differences at each layer are later used to update the weights of the corresponding hidden layers. The above process is repeated until the differences are under some threshold or the algorithm reaches its maximum number of iterations.

In this chapter, we use a backpropagation method called conjugate gradients. Basic backpropagation methods update the weights in the steepest descent direction. While it is sufficient enough to find minima, it is proved that the convergence time is not optimum since the direction of the deepest descent direction may circle around toward a global minima. The conjugate gradients method searches for different directions that are pedicular or conjugate to each other so that optimization in one direction does not affect the optimization in other directions. By searching in different direc-

Algorithm 6.1 Extracting typical patterns algorithm.

Initialize: Weights are chosen randomly in the range from [-0.1, 0.1] **for** $j \leftarrow 1$ **to** M, *the number of pretraining iterations* **do**
    + Train each RBM from bottom to top + Unroll the network to generate data + Calculate the difference between the original trajectory and the model trajectory + Update the weights
for $j \leftarrow 1$ **to** N, *the number of fine-tuning iterations* **do**
    + Fine tune the whole deep network by setting initial weights to the weights obtained from the pretraining phase
+ Store the top hidden units, which are the typical patterns

tions, the algorithm can reach to the minima much faster. We based on the code provided by [75] to perform the algorithm.

6.3.5 Typical Pattern Extraction Algorithm

Algorithm 6.1 summarizes our algorithm to extract typical patterns. Weights are initialized by randomly choosing values from the range of $[-0.1, 0.1]$. Then, pretraining phase is implemented to find initial weights for the next phase, the fine-tuning phase. After learning all the weights and the typical patterns, a test trace is fed to the network and a corresponding model trace is then regenerated. The difference between the test trace and the regenerated trace is calculated to measure the error, which is numerically described in the next section.

6.4 Experiment Results

In this section, we will evaluate the performance of deep learning in reconstructing user trajectory. Random users with trajectories of 38 days were selected. Of the 38 days, 30 days are used for training and the other 8 days are used for testing. After the learning process, weights are learned and then, a test trajectory is applied to the input of the network and used to reconstruct a model trajectory. We will prove that with the learned weights of the hidden units, we can successfully reconstruct the tested trajectories and compare the result with PCA based on the mean square error. The mean square errors of the two algorithms are calculated as the difference between the pixels of the reconstructed trajectory and the input trajectory.

Figure 6.6 shows an input trajectory and its corresponding reconstructed one by deep learning. Notice that the number of dimensions is reduced from $28x28$ to 7. Define the compression ratio as the ratio of the number of the dimensions originally needed to describe the trajectory to the number of dimensions extracted at the top level. We can see that even at a large compression ratio, deep learning performance is still noticeable, especially when compared with PCA. The reconstructed trajectory is not deviated far away from the original one. The mean square error between the two

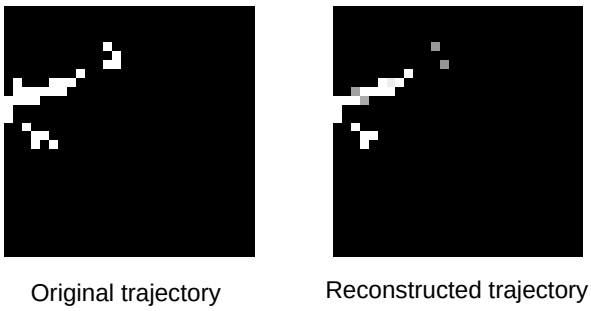


Figure 6.6 Reconstructed trajectory using deep learning.

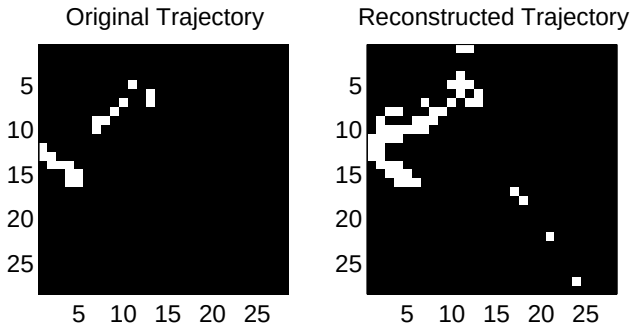


Figure 6.7 Reconstructed trajectory using PCA.

trajectories is 5.3127.

To compare the result with PCA, the same set of training and testing data are used. 7 principle components are extracted to reconstruct the trajectories. As illustrated on Figure 6.7, PCA has much worse performance compared to deep learning. The mean square error of the PCA method is $6.282 * 10^3$, compared to 5.3127 if reconstructed using the deep learning method. Obviously, deep learning beats PCA by far. The noticeable improvement in performance that the deep learning can achieve is due to its deep structure while PCA only has a shallow structure of two layers.

We further test the algorithm with a various hidden units on the top layer and show the result on Figure 6.8. As we can observe, when the number of hidden units in each layer increases, the performance also increases. Obviously, when the number of hidden units increases, the deep network is more flexible and powerful since it has more branches to generate different set of trajectories. However, the complexity increases at the same time as a trade off. Hence, depending on applica-

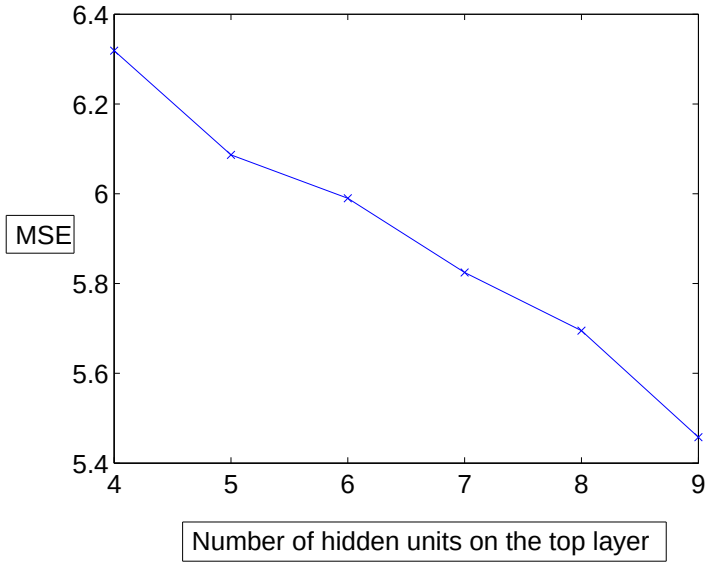


Figure 6.8 Varying the number of hidden units on the top layer.

tions, the deep network can be changed to meet both the performance requirements and complexity requirements.

6.4.1 Possible Future Applications

In our preliminary work, we only extract typical users' moving patterns. However, many applications can be developed based on our result. For example, at the beginning of the day, we can used the learned extracted moving patterns to generate a set of the most possible locations that a user may visit during the day. In combination with signal strength profile collected at the locations, we can effectively schedule users' access to the network. Delay tolerant packet transferring can be delayed until the time the user has a strong signal strength instead of transferring the packets immediately.

Another application is to predict next possible locations given the locations observed from beginning of the day until the current time. As soon as new data are collected, they are used to feed to the deep network to learn the weights of the typical patterns. The weights are then matched with the stored weights of different days and the most similar moving pattern will be selected to generate possible future locations for the rest of the day. This application is specifically tested by Eagle et al. in [69]. Since deep learning significantly outperforms PCA, which is used to extract typical patterns in [69], we would believe that our proposed algorithm can significantly improve their results.

6.5 Conclusions

In this chapter, we have implemented a 4-layer deep network to learn typical users' moving patterns. At the first step, we pretrain the network by learning it layer by layer using RBM learning algorithm. Then, a backpropagation is deployed to fine-tune the network. Real traces were collected and transferred into binary images. The performance evaluated with the traces is shown to be significantly improved compared to the traditional methods. Our preliminary result also suggests some promising applications and can be extended to many other fields.

Chapter 7

Conclusions and Future Works

7.1 Conclusions

In this thesis, we have proposed two frameworks for wireless device security and wireless device location based service. In the first framework, we proposed an algorithm to detect the primary user emulation attack in cognitive radio, named DECLOAK. DECLOAK exploits hardware dependent features, increasing the security for devices. It is an unsupervised approach that requires no labor. We compared its performance to the performance bounds and showed that it is close to the upper bound. Though the algorithm is specifically designed for PUE attack detection in CR, it can be easily extended to many other security applications.

Then, we resolved some technical core problems in the LBS. First, a lightweight energy-efficient solution to identify new and revisited indoor locations based on WiFi scans and accelerometer readings was developed. The algorithm is named LOIRE. LOIRE has the capability of handling missing and noisy data effectively. It takes advantage of a batch sampling approach and significantly improves the speed comparing to other sampling schemes like Gibbs sampler. Moreover, it proposes a Quickest Detection approach to turn off the WiFi scans soon after confidence are collected enough to save the smartphone energy. Experimental results showed that LOIRE could handle missing and noisy data effectively, without manual survey or prior knowledge of the number of places, and consume much less energy compared to other state-of-the-art approaches. LOIRE setup a basic framework and can be utilized in many applications involved with LBS.

Finally, to complete the LBS framework we proposed two prediction approaches. In the first approach, we proposed the Dynamic Hidden Markov Model, which is a Hidden Markov Model with the states determined dynamically. The model can involve itself with the growing number of places that the users visited. We investigated the performance of the scheme in a particular application, where delay tolerant data are scheduled to be transmitted in the most appropriate time slot, calculated based on the transition matrix of the model. Through simulations, we showed that the proposed scheme improves the overall throughput of the system by 50.34% compared to a naive approach. The results presented in this chapter demonstrated the promises of exploiting user and application characteristics in better resource management in cellular data networks. In the second prediction approach, we have implemented a 4-layer deep network to learn typical users' moving patterns. The algorithm consists of two steps, a pretrain step using RBM, and a backpropagation

step to finely tune the network. Tested on the real traces, the approach demonstrated considerable improvement in extracting features, compared to other approaches such as PCA. Our preliminary result also suggests some promising applications and can be extended to many other fields.

Overall, most of the methods we proposed are unsupervised, and therefore, require no or little users' interactions. We have also resolved one of the hardest problem in Machine Learning, the incomplete data problem, invented an online sampling approach to replace the highly complex Gibbs samplers, and proposed two novel approaches in predicting users' future whereabouts. The algorithms are light-weight and energy-efficient whereas maintain high performances. Together with the proven success in experiment evaluations, they make the Location Based Service framework readily available to be implemented on mobile devices on small scale.

7.2 Future Works

7.2.1 Future improvements on the theory side

7.2.1.1 From the security perspective

DECLOAK has shown to be highly effective in detecting primary user emulation attacks in Cognitive Radio. However, it is not limited to that only application. DECLOAK has the ability of detecting two types of attacks, the Sybil attack and the Masquerade attack. These are the two types of attacks where the attackers may clone other legitimate devices identities or generate multiple identities to exploit network resources. With the ability to detect the number of devices, DECLOAK can be applied to detect an attack in this case. In the future, we also consider to extract more unique features that can well represent the devices.

Currently, the sampling approach in DECLOAK is Gibbs sampler, which is heavy and requires $O(n^2)$ running time. In the future, we may investigate another sampling scheme to replace it and reduce the complexity of the algorithm. A variational sampling approach may be considered due to it simplicity. Or a batch sampling approach would be suitable if we know the label of the a whole set of observations.

7.2.1.2 From the LBS perspective

Theoretically, we plan to integrate other sensory data such as the earth magnetic field and lighting data to further improve the indoor location identification in LOIRE. Room structures are stable, unless it is rearranged. The structure of a room are established by a metal frame, surrounded by electrical wires and other furniture that contain metals. These settings are unique for each lo-

cation and therefore, the measured earth magnetic field is unique for each location. Moreover, magnetometer is commonly found in smartphones and there is no need for hardware modification. Thus, the magnetic field could potentially be a good feature. Other than the earth magnetic field, a lighting sensor would be of beneficial. With a fixed location of light bulbs within a room, the lighting condition within a room will be different to that of a hallway. Therefore, lighting condition can be used to detect entrances and departures of a room.

We are also working on the problem of Quickest Detection to quantify a theoretical threshold for the problem of new place decision making. Currently, an empirical threshold was chosen. However, to guarantee false positive and false negative of the place identification problem, we need to find the distribution of the log likelihood ratio of the two hypotheses. Based on the distribution of the ratio, one can easily quantify a threshold to guarantee a false alarm rate.

An important future work is to integrate LOIRE with the HDMM. Currently, HDMM was only tested with synthetic data. The places identified in LOIRE play the role of states in the HDMM. Once the states are correctly identified, applying HDMM using the states are straightforward. After the integration is finished, a complete framework for LBS is well established.

For the prediction part, we would further develop the prediction algorithm based on Deep Learning and exploit possible low complexity and high precision prediction algorithms. With the extracted features and the learned network, we would further test the model by using partial trace of a day and check if the network can generate a precise full day trace. We would also incorporate other features such as time into the model. And most importantly, we would increase the resolution of the model to accommodate a bigger map.

7.2.2 Future improvements on the application side

On the application side, it is desired to implement the LBS on smartphones on a large scale to completely test the algorithms. We considering to develop applications based on the current LBS framework and integrate different types of services to enhance users' experiences. The first application can be a micro-climate control application. The system automatically detect the appearance of users and automatically adjust the temperature accordingly. Given predicted location, the application can also preheat/precool the locations before the users actually arrive. Another application can be of interest is the geofencing application. In geofencing, it is very important to determine a fine grained location information, i.e. up to a room level. For example, within a shopping mall, an Apple store may want to deliver a coupon only to its customers within the store, not to the Victoria

Secret store next to it. The applications will be available to all operating systems and will be free for every users. By attracting more users, we will have a chance to test the framework on large scale and improve users' experiences.

Overall, there are many interesting applications/extensions can be built/investigated upon our proposed security and LBS framework. With its high performance and well tested features, the framework established an important step toward the fields.

Bibliography

[1] M. Rogowsky. (2013) More than half of us have smartphones, giving apple and google much to smile about. [Online]. Available: http://www.forbes.com/sites/markrogowsky/2013/06/06/more-than-half-of-us-have-smartphones-giving-apple-and-google-much-to-smile-about/

[2] B. Clarkson, *Life Patterns: Structure from wearable sensors, PhD thesis*. Boston, MA: MIT Media Lab, 2002.

[3] C. M. Bishop, *Pattern recognition and machine learning*. New York, NY: Springer, 2006.

[4] Y. W. Teh, *Dirichlet processes*. Encyclopedia of Machine Learning, Springer, New York, 2007.

[5] F. Wood and M. J. Black, "A nonparametric bayesian alternative to spike sorting."

[6] H. S. S. A. Gelman, J. B. Carlin and D. B. Rubin, *Bayesian data analysis, 2nd edition*. London, UK: Hall/CRC Chapman, 2003.

[7] P. Resnik and E. Hardisty. (2010) Gibbs sampling for the uninitiated. [Online]. Available: http://www.cs.umd.edu/~hardisty/papers/gsfu.pdf

[8] C. E. Rasmussen, "The infinite gaussian mixture model," in *Proc. of Neural Information Processing Systems (NIPS) 2000*, Denver, CO, Nov. 2000, p. 554560.

[9] A. Ranganathan. (2004) The dirichlet process mixture model. [Online]. Available: http://biocomp.bioen.uiuc.edu/journal_club_web/dirichlet.pdf

[10] Z. Ghahramani, "Nonparametric bayesian methods," in *Proc. UAI Conference*, Edinburgh, Scotland, Jul. 2005.

[11] J. S. Liu, "The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem."

[12] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis."

[13] H. Kobayashi and J. B. Thomas, "Distance measures and related criteria," in *Proc. the 5th Annual Allerton Conference on Circuit and System Theory*, Nagoya, Japan, Oct. 1967, pp. 491–500.

[14] F. Nielsen and S. Boltz, "The burbea-rao and bhattacharyya centroids."

[15] W. Hoeffding, "Probability inequalities for sums of bounded random variables."

[16] T. M. Cover and J. A. Thomas, *Elements of information theory*. New York, NY: Wiley-Interscience, 1991.

[17] J. M. III and G. Q. Maguire, "Cognitive radio: Making software radios more personal."

[18] D. N. E. Hossain and Z. Han, *Dynamic Spectrum Access in Cognitive Radio Networks*. UK: Cambridge, 2009.

[19] J. M. P. R. Chen and J. H. Reed, "Defense against primary user emulation attacks in cognitive radio networks."

[20] B. J. B. R. W. Thomas, R. S. Komali and P. Mahonen, "A bayesian game analysis of emulation attacks in dynamic spectrum access networks," in *Proc. IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Singapore, Apr. 2010.

[21] C. C. Z. Chen, T. Cooklev and C. P. Raez, "Modeling primary user emulation attacks and defenses in cognitive radio networks," in *Proc. Performance Computing and Communications Conference (IPCCC)*, Phoenix, AZ, Dec. 2009, pp. 208–2150.

[22] K. Bian and J.-M. Park, "Security vulnerabilities in ieee 802.22," in *Proc. the Fourth International Wireless Internet Conference (WICON)*, Hawaii, Nov. 2008.

[23] Z. J. S. Anand and K. P. Subbalakshmi, "An analytical model for primary user emulation attacks in cognitive radio networks," in *Proc. IEEE International Symposium of New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Chicago, IL, Oct. 2008, pp. 1–6.

[24] S. A. Z. Jin and K. P. Subbalakshmi, "Mitigating primary user emulation attacks in dynamic spectrum access networks using hypothesis testing."

[25] H. Li and Z. Han, "Dogfight in spectrum: Jamming and anti-jamming in cognitive radio systems," in *Proc. IEEE Conference on Global Communications (Globecom)*, Hawaii, Nov. 2009.

[26] M. G. V. Brik, S. Banerjee and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. the 14th ACM International Conference on Mobile Computing and Networking*, San Francisco, CA, Sep. 2008.

[27] S. C. B. Danev, H. Luecken and K. E. Defrawy, "Attacks on physical-layer identification," in *Proc. the 3rd ACM Conference on Wireless Network Security (WISEC)*, New York, NY, Mar. 2010.

[28] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. IEEE Standard 802.11, 2007.

[29] W. A. Gardner, *Statistical spectral analysis: A non-probabilistic theory*. Prentice Hall, 1988.

[30] T. A. T. F. K. Maeda, A. Benjebbour and T. Ohya, "Recognition among ofdm-based systems utilizing cyclostationarity-inducing transmission," in *Proc. IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Dublin, Ireland, Apr. 2007.

[31] I. A. K. Kim, C. M. Spooner and J. H. Reed, "Specific emitter identification for cognitive radio with application to ieee 802.11," in *Proc. the IEEE Global Telecommunications Conference (GLOBECOM '08)*, New Orleans, LA, Nov. 2008.

[32] W. T. J. Yang, Y. Chen and J. Cheng, "Determining the number of attackers and localizing multiple adversaries in wireless spoofing attacks," in *Proc. the 28th Conference on Computer Communications*, Janeiro, Brazil, Apr. 2009.

[33] C. Song, Z. Qu, N. Blumm, and A.-L. Barabsi, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, Feb. 2010.

[34] J. Krumm, "Ubiquitous advertising: The killer application for the 21st century," *IEEE Pervasive Computing Magazine*, vol. 10, no. 1, pp. 66–73, Jan. 2011.

[35] C. Ellis, J. Scott, M. Hazas, and J. Krumm, "EarlyOff: Using house cooling rates to save energy," in *Proc. ACM BuildSys*, New York, NY, USA, Nov. 2012, pp. 39–41.

[36] J. Scott, A. B. Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar, "PreHeat: Controlling home heating using occupancy prediction," in *Proc. ACM UbiComp*, Beijing, China, Sep. 2011, pp. 281–290.

[37] S. Isaacman, R. Becker, R. Cceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky, "Identifying important places in people's lives from cellular network data," in *Proc. Pervasive Computing*, Berlin, Heidelberg, Germany, Jun. 2011, pp. 133–151.

[38] R. E. Silverman, *Tracking Sensors Invade the Workplace*. [Online] Available: $http://online.wsj.com/article/SB10001424127887324034804578344303429080678.html$, 2013.

[39] D. B. Rubin, "Inference and missing data," *Biometrika*, vol. 63, no. 3, pp. 581–592, Dec. 1976.

[40] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 37, no. 6, pp. 1067–1080, Nov. 2007.

[41] M. Azizyan, I. Constandache, and R. R. Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proc. ACM MobiCom*, New York, NY, USA, Sep. 2009, pp. 261–272.

[42] J. Park, B. Charrow, D. Curtis, J. Battat, E. Minkov, J. Hicks, S. Teller, and J. Ledlie, "Growing an organic indoor location system," in *Proc. ACM MobiSys*, New York, NY, USA, Jun. 2010, pp. 271–284.

[43] A. Varshavsky, E. D. Lara, J. Hightower, A. LaMarca, and V. Otsason, "GSM indoor localization," *IEEE PerCom*, vol. 3, no. 6, pp. 698–720, Dec. 2007.

[44] D. H. Kim, Y. Kim, D. Estrin, and M. B. Srivastava, "Sensloc: Sensing everyday places and paths using less energy," in *Proc. ACM SenSys*, New York, NY, USA, Nov. 2010, pp. 43–56.

[45] Y. Jiang, X. Pan, K. Li, Q. Lv, R. Dick, M. Hannigan, and L. Shang, "ARIEL: Automatic Wi-Fi based room fingerprinting for indoor localization," in *Proc. Ubicomp*, New York, NY, USA, Sep. 2012, pp. 441–450.

[46] C. Wu, Z. Yang, Y. Liu, and W. Xi, "WILL: Wireless indoor localization without site survey," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 64 – 72.

[47] Y. Jiang, K. Li, L. Tian, R. Piedrahita, X. Yun, O. Mansata, Q. Lv, R. P. Dick, M. Hannigan, and L. Shang, "MAQS: A personalized mobile sensing system for indoor air quality monitoring," in *Proc. Ubicomp*, New York, NY, USA, Sep. 2011, pp. 271–280.

[48] A. M. Hossain, Y. Jin, W. S. Soh, and H. N. Van, "SSD: A robust RF location fingerprint addressing mobile devices' heterogeneity," *IEEE Transaction on Mobile Computing*, vol. 12, no. 1, pp. 65–77, Jan. 2013.

[49] K. Fukunaga and L. D. Hostetler, "The estimation of the gradient of a density function, with applications in pattern recognition," *IEEE Transaction of Information Theory*, vol. 21, no. 1, pp. 32–40, Jan. 1975.

[50] Y. Sheng, K. Tan, G. Chen, D. Kotz, and A. Campbell, "Detecting 802.11 MAC layer spoofing using received signal strength," in *Proc. IEEE INFOCOM*, Phoenix, AZ, USA, 2008, pp. 1768 – 1776.

[51] H. V. Poor and O. Hadjiliadis, *Quickest detection*. Cambridge University Press, UK, 2009.

[52] L. N. Kanal, *Perceptron*. Encyclopedia of Computer Science, 2003.

[53] Cisco. (2010) Cisco visual networking index: Global mobile data traffic forecast update. [Online]. Available: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[54] S. R. A. K. I. Trestian and A. Nucci, "Taming user-generated content in mobile networks via drop zones," in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 2840–2848.

[55] Y. Wang and X. Liu. (2012) Update: User-profile-driven adaptive transfer. [Online]. Available: http://www.cs.ucdavis.edu/~liu/preprint/Update.pdf

[56] C. A. H. M. C. Gonzalez and A. L. Barabasi, "Understanding individual human mobility patterns," vol. 453, pp. 779–782, Jun. 2008.

[57] N. B. C. Song, Z. Qu and A.-L. Barabsi, "Limits of predictability in human mobility," vol. 327, pp. 018–1021, Feb. 2010.

[58] A. S. R. G. M. H. K. M.-R. Ra, J. Paek and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proc. MobiSys*, San Francisco, CA, Jun. 2010, pp. 255–270.

[59] R. M. A. Balasubramanian and A. Venkataramani, "Augmenting mobile 3G using wifi," in *Proc. the 8th international conference on Mobile systems, applications, and services (MobiSys)*, New York, NY, Jun. 2010, pp. 209–222.

[60] S. Mangold and S. Kyriazakos, "Applying pattern recognition techniques based on hidden markov models for vehicular position location in cellular networks," in *Proc. IEEE Vehicular Technology Conference*, Houston, Texas, Sep. 1999, pp. 780–784.

[61] D. Ashbrook and T. Starner, "Using gps to learn significant locations and predict movement across multiple users."

[62] A. Bhattacharya and S. K. Das, "Lezi-update: An information-theoretic framework for personal mobility tracking in pcs networks."

[63] Z. O. N. Ding, "Variational nonparametric bayesian hidden markov model," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, TX, Mar. 2010, pp. 2098 – 2101.

[64] R. Z. Z. Han and V. H. Poor, "Repeated auctions with bayesian nonparametric learning for spectrum access in cognitive radio networks."

[65] H. V. P. T. B. W. Saad, Z. Han and J. B. Song, "A cooperative bayesian nonparametric framework for primary user activity monitoring in cognitive radio networks."

[66] H. Abu-Ghazaleh and A. Alfa, "Application of mobility prediction in wireless networks using markov renewal theory."

[67] D. Katsaros and Y. Manolopoulos, "Prediction in wireless networks by markov chains."

[68] J. K. Le and J. C. Hou, "Modeling steady-state and transient behaviors of user mobility: Formulation, analysis, and application," in *Proc. the 7th ACM international symposium on Mobile ad hoc networking and computing*, Florence, Italy, May 2006, pp. 85–96.

[69] N. Eagle and A. Pentland, "Eigenbehaviors: identifying structure in routine."

[70] W. C. Peng and M. S. Chen, "Mining user moving patterns for personal data allocation in a mobile computing system," in *Proc. the 29th Int. Conf. Parallel Processing*, Toronto, Canada, Aug. 2000, pp. 573–580.

[71] O. H. P. J. W. Lee and K. H. Ryu, "Temporal moving pattern mining for location-based service."

[72] J. R. F. Calabrese and C. Ratti, "Eigenplaces: Analyzing cities using the space-time structure of the mobile phone network."

[73] Y. Bengio, "Learning deep architectures for ai."

[74] D. R. I. Arel and T. Karnowski, "Deep machine learning - a new frontier in artificial intelligence research."

[75] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," vol. 313, pp. 504 – 507, Jul. 2006.

[76] G. E. H. D. H. Ackley and T. J. . Sejnowski, "A learning algorithm for boltzmann machines."

[77] S. O. G. E. Hinton and Y. Teh, "A fast learning algorithm for deep belief nets."