Human Mobility Modeling and Predictive Analysis

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2016

Danaipat Sodkomkham

Abstract

In this research, we aim to advance in machine learning techniques that can help us gain a better understanding of human mobility. Human mobility has been studied broadly in the past few decades. Ranging from a higher level perspective, such as travel patterns, visitation patterns and predictive modeling, to a lower level, e.g. path finding mechanism, tasks prioritization and routing. A study conducted in this research has shown samples of tasks dependent mobility patterns inside an office environment and its potential predictability. The predictive models for both short-term prediction and long term prediction have also been developed and successfully tested on a real dataset. The results have given us more understanding of dynamicity and how the participants would utilize the space. Therefore there are potentials for spatially-related applications, such as users-based power management system and notifications of suspicious behaviors, to be built based on the proposed methods.

Even though, the proposed probabilistic model for long term human mobility prediction was initially designed for discrete representation of positions (e.g. locations associated with sensor IDs), the approach can also be extended to support continuous representation (e.g. xy coordinates). The probability density and the likelihood of future visitations at any given point (x, y) of interest can be estimated using a non-parametric method called kernel density estimation or KDE. However, standard KDE is a costly operation and cannot scale well with the unbounded size of data streaming from sensors or mobility tracking system. Besides, existing online KDEs cannot handle multi-dimensional data streams very efficiently. Therefore, we proposed a kernel density compression algorithm that was designed for multivariate (and univariate) data streams for efficient density evaluation that can scale well to streaming sensor data. Since continuous tracking system was not implemented yet, we tested the proposed method with the problem of real time decoding of neural encoding/decoding. The developed online kernel density compression algorithms not only enables real-time encoding and decoding of neural activities but can also be generalized to estimate probability density function of any multivariates data streams without any modification. More specifically, the proposed online KDE technique can be applied to implement an online probabilistic model for long-term human mobility prediction and can be used to visualize dynamicity of space utilization in real-time to help us understand how the participants utilize different

areas of the facility during the day.

Acknowledgement

I would like to express my very great appreciation to Professor Masayuki Numao for his valuable and constructive suggestions throughout this research work. This thesis would have not been possible without his guidance and support in various topics. I would also like to express my very great appreciation to Professor Satoshi Kurihara, Koichi Moriyama, Ken-ichi Fukui and Roberto Legaspi for their help in offering me the resources in running the research.

I would also like to express my sincere gratitude to Dr. Fabian Kloosterman for his patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to thank all Master's and Ph.D. students under Professor Numao's supervision, for their helps during the project.

More importantly, this work was partly supported by JSPS Strategic Young Researcher Overseas Visits Program for Accelerating Brain Circulation, National Institute of Mental Health Grant MH-061976 and Office of Naval Research MURI N00014-10-1-0936 Grant to M. A. Wilson.

Finally, I wish to thank my parents and friends for their support and encouragement throughout my study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Problem Statement

In the past decade, machine learning tools designed for understanding and predicting human mobility have been developed and included in many real-world applications since a variety of options for mobility tracking systems have been introduced and publicly adopted. To name a few, commercial applications such as taxi services are actively relying on the understanding of trajectory patterns and reliable predictions to offer quality service to the customers[1]. In another context, PUCK architecture [16] was introduced to provide a smart reminder in the smart environment that automatically recognizes habitual activities, and reminds the habitants when they forgot something important. This could be helpful for some participants who have dementia, or mind cognitive impairment. Google Now[2] uses user's mobility information on a smartphone to provide better, and more specific search results. It can automatically understand and predict where the user is likely to be in future, and then provide some useful information about the trip, such as traffic condition, estimated arrival time, and weather information at the predicted destination. Moreover, the ability to predict future locations of people is also an important element in transportation planning [43, 39], bandwidth provisioning in wireless local area network [70], and targeted advertisement dissemination [25].

In our application, we utilized an actual office environment with various built-in sensors and actuators to enable ubiquitous computing technology to control different settings in the environment. Our prototype smart office environment was initially de-

[1]Taxi Service Tajectory Prediction Challenge @ ECML PKDD 15 https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i

[2]http://www.google.com/landing/now/

signed to create a working environment that can learn multi-users' behavioral activities and intelligently react to these activities smartly. Our objective in developing this smart environment is to simplify mundane repetitive tasks and to improve people's lives. For example, a smart office that could predict future occupancy of a meeting room and automatically prepare the electronic facilities in the room prepared before the meeting. A smart office could also be programmed to predict needs from daily activities; for example, to ensure that hot coffee was ready to be served at a particular time. These applications require the ability to foresee individual's future whereabouts and mobility. This is referred to the open problem of *long-term human mobility prediction*. In contrary, predicting next steps of a moving subject is another class of problem known as *short-term human mobility prediction*. In this research, we addressed both problems in multi-users environment using low-profile, ambient simple sensors, such as IR proximity sensors and magnetic sensors.

Many researches in the past had been relying on accurate sources of mobility traces, such as GPS [59, 43], human tracking cameras [78], RFID tagging [2, 31] and mobile phone data [22, 69]. But not many attentions had been given to tracking methods that are less accurate but are less intrusive to the subjects privacy, such as human mobility tracking by detecting changes in WiFi signal strength [7, 36, 45], simple infrared proximity sensors and other kinds of simple motion detection sensors. Although, using mobility data from accurate sources enables researchers to focus more on the development of new modeling and prediction technique without much concern about reliability and noise in the training and test data. On the other hand, human mobility traces from low accuracy sensing techniques are much less reliable and full of noisy samples. Preprocessing and feature extraction techniques, such as patterns mining, must be applied to extract necessary information and remove non-patterns samples from the dataset.

Especially in this work, we installed IR proximity sensors and magnetic sensors in different area of the experimental space to detect motions and interaction from the users with environment. By specification, the IR senors trigger when there is object presented in front of the sensor within 20-150cm range and the magnetic sensors trigger when there are changes in the sensed magnetic force. Since different sensors associate with different locations that they were installed, trajectories of the users can be represented by sequences of sensor firing events. Thus, prediction of future position of a traveling subject can be formulated as a classification problem, where target classes are different

locations associated with different sensor IDs.

False detection of motions were very common in this setting due to electrical interference from the power source. IR sensors, in particular, can also fail to detect movements when the users were too far or too close from the sensors. Other than technical issues, difficulties when analyzing this dataset also came from the test environment where there were multiple users. Because these sensors cannot identify and distinguish different users, we could not separate any two movement sequences from two different users by simply looking at the raw stream of sensor triggers.

In previous research [15], we have shown that these problems can be solved efficiently using frequent sequential patterns mining algorithms. Overlapping trajectories can be separated accurately if both trajectory patterns occurred separately frequently enough. In addition to spatial frequency, we added the temporal aspect to the frequent sequential patterns mining algorithm called PrefixSpan [50] to create a trajectory mining algorithm that can search for both spatially and temporally frequent patterns. Although, we could not assess quality of the algorithms from the mined patterns directly, we trained a machine leaning algorithm to predict future location of a moving subject based on given trajectory patterns and used their prediction accuracy as a proxy measure to evaluate quality of the mined trajectory patterns. The results showed that the proposed temporal sequential patterns mining (TSPM) algorithm was able to achieved higher accuracy in the prediction task compared to PrefixSpan. That is, trajectory patterns mined with temporal features resulted in more informative patterns and more suitable for prediction task than the standard sequential pattern mining techniques, such as PrefixSpan.

A question yet unanswered was how other factors, such as pattern length, support, and confidence settings of the patterns, could affect prediction accuracy. Therefore, in Chapter 3, we present an adaptation of the theoretical analysis of maximum predictability [70] to discover how we could increase predictability from the preprocessing step.

As mentioned earlier, the problem of human mobility can be divided into two subclasses: short-term and long-term prediction. Short-term prediction problems mainly focus on predicting next moves or destination of a moving subject, given prefix trajectory. In this work we proposed a novel design for short-term prediction in Section 4.2. The proposed technique is able to accept varying trajectory length and is able to outperform our initial design that was based on a decision tree classifier [68]. Without

a doubt, recent movements along the prefix trajectory are highly predictive to where the user was heading toward. In contrast, recent movements hardly involve in subject's whereabout in further future (e.g. next day or next week). For this reason, prediction techniques that work well predicting short-term mobility would perform poorly when used to solve long-term prediction problem [59].

Therefore, in Section 4.1, we addressed the problem of long-term mobility prediction differently from existing techniques that were designed for short-term prediction [59, 43]. Instead of predicting based on trajectory patterns, we proposed a model that models repetitive patterns in human movements. The proposed aperiodic/periodic model for long-term human mobility prediction is not only limited to discrete representation of locations (i.e. location name, sensor ID), but also capable of modeling repetitive visitation patterns where each visitation is represented with continuous values (i.e. xy-coordinates). The probability density and the likelihood of repeating visitation at any (x, y) position of interest can be estimated using a kernel density estimation technique.

However, kernel density estimation is a costly operation and cannot be implemented to execute efficiently for online learning, where the model needs continuous updates from stream of sensor data. Furthermore, existing techniques that optimized KDE for streaming data are not efficient when handling multivariate stream (e.g. 2D streams of xy-coordinates). Therefore, in Chapter 5, we propose a kernel density compression algorithm for online KDE that can handle multi-dimensional data streams efficiently with accuracy. To demonstrate the speed and accuracy of the proposed kernel density compression algorithm in a real-time applications, the algorithm was tested with the simulation of neural decoding, in which the density model needs to update to new sample of multi-dimensional neural signal every 1ms.

The developed online kernel density compression algorithms not only enables real-time decoding of neural activities but can also be generalized to estimate probability density function of any multivariates data streams without any modification. More specifically, the proposed online KDE technique can be applied to implement an online probabilistic model for long-term human mobility prediction and can be used to visualize dynamicity of space utilization in real-time to help us understand how participants utilize different areas of the facility during the day.

In summary, we proposed an improvement of short-term human mobility predictor (discrete classes of location IDs) and a novel probabilistic model for long-term human

Table 1.1: Outline of the thesis

	Discrete representation	Continuous representation
Short-term prediction	addressed in Section 4.2	not addressed
Long-term prediction	addressed in Section 4.1	can be implemented efficiently with the technique described in Chapter 5

mobility prediction that works with both discrete and continuous representation of positions. The contributions of this thesis can be summarized as shown in Table 1.1.

Chapter 2

Literature Review

In this chapter, we discuss some interesting researches that are related to our work. That includes human mobility tracking techniques, human mobility modeling, predictive techniques and existing technologies for density estimation for streaming data.

2.1 Human Mobility Tracking

Regarding various choices of tracking technologies, it seems logical to select ones that provide most accurate results. However, there is an apparent trade-off between accuracy and conspicuousness of the tracking technique. Even though, high accuracy methods such as camera-based localization, GPS and similar tracking systems are capable of providing precise tracking of each individual subjects, they can cause uneasiness to the subjects. For example, by using colored pictures from cameras with object detection technologies and semi-supervised classification algorithm, Yu et al. [78] were able to create a system that recognizes people and their positions. Moreover, they were directly able to map each individual's movement to a floor map. From this example, it is evident that there needs to be a balance between rich mobility information and discomfort caused by the cameras. Apart from the camera techniques [53, 5] discussed earlier, mobile phone data [22, 69], GPS [59, 43], and RFID tagging [2, 31] requires people to carry (or put on) a tracking device while in the environment, which is not feasible in real-world implementations.

On the other hand, simple sensors, such as infrared proximity sensors and magnetic sensors, are small enough to blend-in to the environment and keep their small profiles while seamlessly observing human mobility. However, this tracking method has not been used in many human mobility researches in the past because the data collected by such

simple sensors is less informative and more prone to noise than high precision sensing technologies. Besides, the amount of useful information after removing all the noisy signals is yet questionable. That being said, the capability of the mobility predictive model built using simple sensors' data might be limited.

2.2 Limits of the predictability in Human Mobility

To quantify quality of collected datasets, Song et al. introduced an analysis of the predictability of human mobility [69]. The analysis explored the limitation in predictability of an individual's movements. Note that they did not take the quality of the prediction techniques into consideration.

By employing Fano's inequality [17, 46], the analysis by Snog et al. assessed whether the upper limit of the probability of a moving person's destination could be correctly predicted given the most recent trajectory and the past collective mobility data.

Despite variations in individual's daily behavior, Song's [70] analysis over a large population monitored by mobile phone data shows 93% potential predictability in an individual's mobility. In other words, predicting individual's movements can be effectively achieved when historical trajectory data is available. However, the analysis neither discuss which features of trajectory pattens contributed most to the prediction, nor how the predictability would change if the patterns were from multiple users. In this study, we extend the analysis of human predictability to cover (1) trajectory patterns' features importances and (2) the predictability analysis of collective human mobility data, both short-term and long-term predictions.

2.3 Human Mobility Modeling and Prediction

Human mobility has been modeled in different ways depending on the prediction technique used in the model. Each approach has different merits and inferiorities. In the past, a considerable number of literatures, especially in traffic prediction [43, 69] and smart environment researches [2], mainly focused on spatial characteristics of movements wherein individual's trajectories are tracked and frequent trajectory patterns are extracted and analyzed for prediction. The trajectories-based predictive models usually work well in the short-term human mobility prediction problem in which only instant future locations of a moving person (or vehicle) are interested. As an example, in [43],

the WhereNext framework implemented a short-term location prediction using the temporal annotated sequences that provide movement patterns with rich spatio-temporal contexts to the predictive model. Hence, it was able to predict the next movement of a moving user more accurately than using movement traces alone. The predictive model in WhereNext was implemented by creating a tree data structure that contains all frequent trajectory patterns. Next, a matching function must be defined to match these patterns with a newly observed movement; then, what comes after the similar movement pattern founded on the predictive tree will become the predicted next location. In contrast, temporal patterns-based models [66, 55, 60] are more promising for long-term prediction in which future whereabouts of a user even for several hours or days ahead are predicted.

2.4 Online Kernel Density Estimation

Density estimation are fundamental in a wide range of scientific applications including machine learning. Probabilistic machine learning models oftentimes require a knowledge of underlying samples distribution in order to work. Besides, it is a common tool for data analytic tasks in which the probability density can be visualized with heat-maps or contour plots for visual presentation and analytic purposes.

Density estimation is a costly operation. The time complexity for a traditional kernel density estimation (KDE) is the order of $\mathcal{O}(MN)$, where M is the number of evaluation points and N is the number of samples (also equals to the number of kernels when implemented without any tweaks). That is, density estimation becomes slower as the size of the dataset increases. For this reason, traditional KDE cannot be efficiently used with streaming data, where new samples are continuously observed. Although state-of-the-art dual-tree based KDE [24], which is known to be the fastest and most accurate algorithm, can scale the computation to $\mathcal{O}(N)$, it requires $\mathcal{O}(N \log N)$ time for construction (building tree structures). However, the dual-tree approach was not designed as an incremental algorithm. Namely, it is inefficient in an online setting, where the tree-based data structure requires reconstruction every time a new observation becomes available. So are the recently proposed distributed and parallel KDE [81] that was designed specially for very large datasets. The distributed KDE employed sub-sampling techniques to reduce size of KDE model and speedup the density estimation. However, the algorithm proposed in [81] was not designed to be incremental, namely a new data

point cannot be directly inserted to the built model without rebuilding the entire model. Because each update takes $\mathcal{O}(N)$, the sub-sampling KDE was obviously more suitable for batch processing than online processing, where N could grow unboundedly.

Reducing the number of computations required to compute a point estimate of the density has been a main interest in optimizing KDE. A number of improvements have been considered to reduce the number of kernel components. For example, a straightforward binning method has been introduced in [61, 30], where the density is estimated from a sum of densities estimated from each bin weighted by the number of samples placed in each bin. In [64], a method that estimates the density with the convolution theorem using Fourier transformation, in which binning of data is also required, has been proposed. However, this requires the bin size for each dimension to be specified; therefore, such approaches quickly become inefficient for high-dimensional multivariate data streams.

As an alternative to binning, in [1, 32, 38, 37, 79, 77], samples were clustered and reduced by replacing them with cluster centroids. The main problem with the cluster-based approaches is the computational burden of the optimization required for data partitioning and the solution's dependency on the initial condition. In addition, cluster-based methods, including a self organizing map-based KDE [12], often update their models with a small batch of buffered data (often a few hundreds). Hence, the models are not truly up-to-dated to each new sample, because they have to wait until the buffers get filled so that the models can call clustering procedure to update their density estimators with the new samples.

Some methods have been adapted to handle streaming data by reducing the data condensation overhead so that the model can keep pace with the input stream. The M-kernel merging technique [82] is an online density estimator that can handle a univariate data stream by limiting the number of kernel components and substituting redundant components with a representative component (kernel merging). The model invokes a merging routine whenever its buffer is full. Then, downhill simplex optimization is employed to find the best way to replace two components with a representative component that would minimize the merging cost, which is the absolute error (L_1-norm) between the estimation and the underlying density. Since its kernel merging strategy is based on numerical optimization, the time that the algorithm requires to update the model to a new sample can be high. As a solution, Heinz [28] has introduced an M-kernel based

online KDE with a constant time pairwise merging technique to solve the problem of high update cost of the M-kernel algorithm. However, the M-kernel and other M-kernel-based approaches [28, 29, 6] cannot be generalized to support multivariate data streams without losing speed because the algorithms rely on sortedness which is more complicated to achieved for multivariate data. The lack of total ordering in multidimensional data forces the algorithms to use nearest neighbors search, which would incur additional computational complexity [29]. Therefore, in Chapter 5, we propose a fast kernel compression technique for multivariates kernel density estimation that can be efficiently for data streams.

Chapter 3

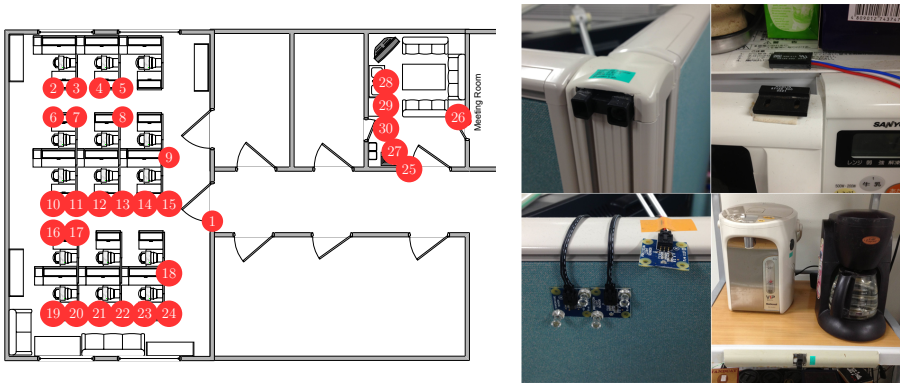
Human Mobility Modeling

In this chapter, we discuss how human mobility can be modeled from noisy sensors data. Additionally, we investigate the limits of predictability for a mobility dataset derived from data collected by the sensors.

3.1 Collective Human Mobility Data

Our experimental smart environment was a functioning working space that included individual cubicle work stations, recreational space, and meeting areas. Twenty graduate students and faculty regularly work in this space. The floor plan is depicted in Figure 3.1(a). Individuals may have different duties, different class schedules, and different daily routines, which results in different directional and temporal mobility patterns. We installed two types of sensors in the environment to monitor activities and movements. Infrared distance sensors were primarily used to detect movement at each specific location. Magnetic sensors were attached to the hinge of the refrigerator and the oven to detect their use (Figure 3.1(b)). All sensors were connected through our laboratory's network, and they continuously fed live streams of mobility data to a database. By employing these ambient sensors, participants did not need to be equipped with an intrusive tracking device during the experimental period and could move normally without being overtly aware that they were being monitored. We observed visitations and mobility in the experimental environment 24 hours a day for 3 months (precisely 92 days) during the autumn semester. The experimental space was divided into 30 locations of interest ($N = 30$); sensors were concealed at each located to record movements. We modeled human mobility with two representations for different purposes as follows.

Temporal sequences of repeated visitations. Collective mobility at each location



(a) Floor plan

(b) Infrared and magnetic sensors used to observe mobility in the experimental space

Figure 3.1: (a) Infrared and magnetic sensors used in the experiment (b) Placement of the sensors

is represented by the temporal sequence of repetitive visitations visited by unknown people during the observation. The state of visitation at a particular time is denoted by a binary value: 1 for *visited*, 0 for *not visited*. For instance, a sequence v_x represents mobility at location x from 00:00 to 23:59, with the sample rate μ of one sample per hour.

$$v_x = \big(\langle t'_0, 0\rangle, \langle t'_1, 1\rangle, \langle t'_2, 0\rangle, \ldots, \langle t'_{23}, 1\rangle\big),$$

where t'_i represents the observed *time frame* from $t_0 + i\mu$ to $t_0 + (i+1)\mu$, and t_0 is the start time, i.e., $t_0 = 00{:}00$ and $t'_0 = [00:00,\ 01:00)$.

Trajectories. By increasing the sensors' sample rate μ to one sample per 200 ms, we were able to record every visitation. Then, from a temporal sequence of visitations, $(\langle x_0, t_0\rangle, \langle x_1, t_1\rangle, \ldots, \langle x_{w-1}, t_{w-1}\rangle)$, we linearly searched for each transition point in the sequence where the transition time $t_{i+1} - t_i > 30$ s to create smaller sequences that represent trajectories.

Despite the unobtrusiveness and simplicity of the ambient sensing method, a considerable amount of the obtained data was noisy. To handle noises (such as false triggered events, sensors blocked by obstacles, and simultaneous trajectories from different people) and extract movement trajectories from the collective mobility

dataset efficiently, we applied the sequential pattern mining algorithm PrefixSpan [50] to extract only sub-trajectories of length n that appeared in a set of all observed trajectories, $\mathcal{T}$, more frequently than a certain minimum number of occurrences, $support_{min}$, during the observation.

3.2 Limits of Predictability

We evaluated the predictability over the collective mobility dataset using the methodologies introduced by Song et al. [69]. By employing Fano's inequality [17, 46], we assessed whether the upper limit of the probability of a moving person's destination could be correctly predicted given the most recent trajectory and the past collective mobility data.

Let T_i' denote a movement trajectory, and D_i be a destination of T_i' from the observations, $\mathcal{T} = (\langle T_0', D_0\rangle, \langle T_1', D_1\rangle, \ldots, \langle T_m', D_m\rangle)$. Given a predictor: $f : T_i' \mapsto D_i'$ that works well to predict a future location D_i of a moving individual based on recent length n movement trajectory T_i' and a set of length n trajectories $\mathcal{T}_n$, in which $\mathcal{T}_n \subseteq \mathcal{T}$, let e denote the event of failed prediction, i.e., $f(T_i') \neq D_i$, and let $P(e)$ be its probability. According to Fano's inequality, the lower bound on the error probability $P(e)$ can be found in the following inequality.

$$H(D|T') \leq H(e) + P(e)\log(N-1). \tag{3.2.1}$$

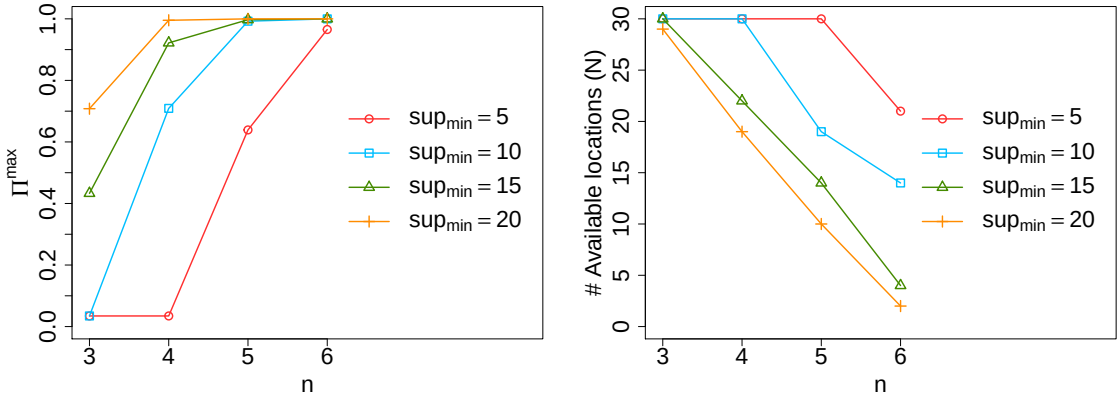
Thus, the probability of predicting correctly, denoted by Π, is $1 - P(e)$. Namely,

$$H(D|T') \leq H(e) + (1-\Pi)\log(N-1), \tag{3.2.2}$$

where the destination D can take up to N possible locations and $H(e)$ is the corresponding binary entropy as follows:

$$\begin{aligned} H(e) &= -P(e)\log(P(e)) - (1-P(e))\log(1-P(e)) \\ &= -(1-\Pi)\log(1-\Pi) - \Pi\log(\Pi). \end{aligned} \tag{3.2.3}$$

The conditional entropy $H(D|T')$ appeared in Eq. (3.2.2) quantifies the amount of information needed to predict the destination D, given recent trajectory T'. Given the probability $P(T')$ of the set of past trajectories $\mathcal{T}_n$ containing T' and the joint probability



(a) The probability Π^{max} as functions of trajectory length n

(b) Relation between the parameter $support_{min}$, trajectory length n, and the number of predictable destinations N left after noise removal

Figure 3.2: Predictability of collective human mobility in smart environment: Π^{max} is the upper bound of the probability that a particular predictive algorithm is able to predict a person's location correctly using only the collective dataset.

$P(T', d)$, the conditional entropy $H(D|T')$ is defined by

$$H(D|T') = \sum_{d \in D, T' \in T} P(T', d) \log \frac{P(T')}{P(T', d)}. \tag{3.2.4}$$

Then, we calculated the entropy $H(D|T')$ separately for each length n trajectories, i.e., $\mathcal{T}_n \subseteq \mathcal{T}$, and analyzed the maximum potential predictability (denoted by Π^{max}) or the probability of predicting the destination of a person correctly given the collective mobility dataset by solving for the Π^{max}, where $\Pi \leq \Pi^{max}$ in Eq. (3.2.5), according to Eqs. (3.2.2), (3.2.3) and (3.2.4)

$$H(D|T') = -(1-\Pi^{max}) \log(1-\Pi^{max}) - \Pi^{max} \log(\Pi^{max}) + (1-\Pi^{max}) \log(N-1). \tag{3.2.5}$$

Figure 3.2(a) shows Π^{max} as functions of n, where n denotes length of the considered trajectories. It is not surprising that n increases predictability; a longer trajectory provides the predictor with more evidence, which helps narrow the search space. The $support_{min}$ also shows potential to eliminate unusual trajectories in the dataset, and gives significantly higher potential predictability. However, there is a trade-off between the degree of predictability and the number of predictable locations, as shown in Figure 3.2(b). A high threshold of minimum support ($support_{min}$) results in fewer numbers of locations N available to the predictive algorithm.

To summarize, despite the fact that the collective human mobility cumulative move-

ments and behaviors from different people and seems diverge to the experimenter in the first place, accurate prediction of the location of a person is achievable with acceptably high probability. However, the analysis does not provide any clue about potential predictability for a long-term prediction configuration when the inference of a person's mobility cannot rely on recent movement patterns and frequent historical trajectories. Moreover, predictive techniques that work well for short-term human mobility predictions cannot be extended to long-term predictions effectively [59, 55]. Therefore, in the next section, we studied the possibility to employ the periodicity in human behavior to foresee their mobility in far future instead of directly modeling trajectories.

3.3 Periodicity in Collective Human Mobility

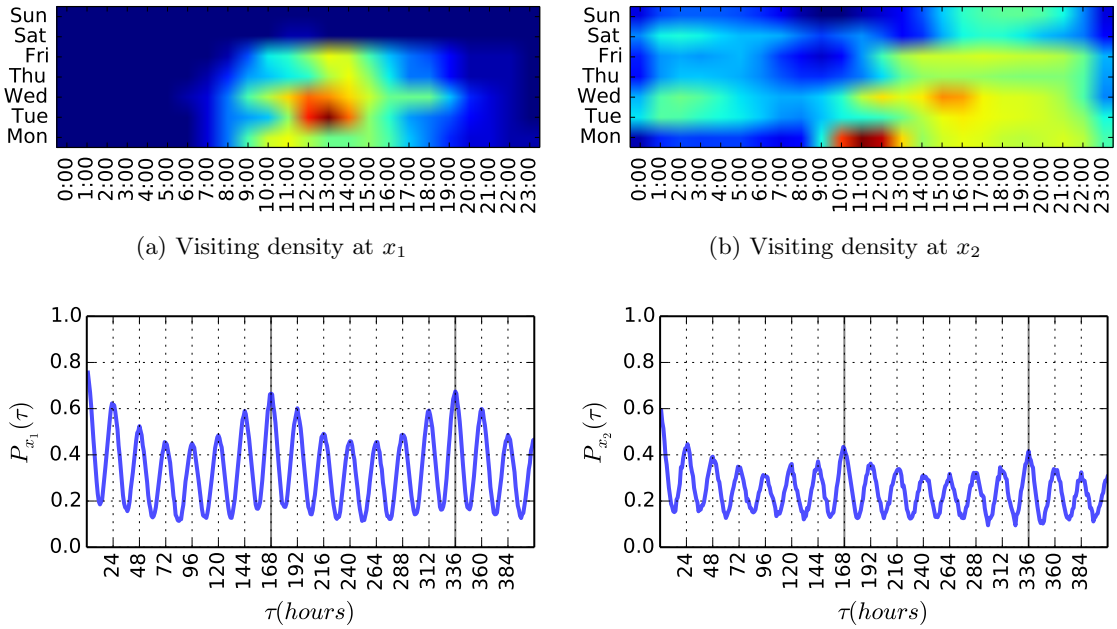
Even without the use of data mining tools, it is evident that most of human activities are periodic to some extent. If a certain action or movement pattern is repeated regularly with a particular interval τ, and if this behavior is consistent over time, it is certainly predictable with the time period τ. In addition, the probability of predicting the correct location of an individual in the future depends on the tendency of such mobility patterns recurring at intervals. Therefore, we define *periodicity probability* to quantify this property formally.

Definition 3.3.0.1. *Let $P_x(\tau)$ denotes the periodicity, which is the probability of a particular event x reoccurring regularly with the constant time interval τ, where τ is a positive integer. Given the temporal sequence, as described in Section 3.1, of events from t'_0 to t'_m in which the location x was visited, the periodicity probability $P_x(\tau)$ is defined by*

$$P_x(\tau) = P\big(v_x(t'_{i+\tau}) = 1 | v_x(t'_i) = 1\big),\ t'_i \in \{t'_0, t'_1, \ldots, t'_{m-1}\}, \tag{3.3.1}$$

where $v_x(t'_i)$ indicates the state of the visitation at x during the time frame t'_i.

At the location x_1, apparent daily periodic behavior is revealed in the density plot presented in Figure 3.3(a), where dense areas are concentrated and aligned on a certain *time of the day*. In contrast, the density plot of x_2 in presented Figure 3.3(b) shows slightly weaker daily periodicity. The loosely dense areas that are distributed broadly over time suggest a low degree of certainty of the repetition. However, weekly periodic behavior is evident in x_2, as distinct shades appeared on each the *day of the week* row



(a) Visiting density at x_1

(b) Visiting density at x_2

(c) The periodicity $P_{x_1}(\tau)$

(d) The periodicity $P_{x_2}(\tau)$

Figure 3.3: In (a) and (b), the density of visitations at location x_1 and x_2 related to *time of the day* and *day of the week* are depicted, respectively. Busy times and days, in which a high number of visitations occurred within the same period of time, are shown in dark red. Dark blue indicates the opposite. The periodicity $P_{x_1}(\tau)$ and $P_{x_2}(\tau)$ of the corresponding locations x_1 and x_2 are shown as a function of time period τ, in (c) and (d), respectively.

indicating unique behavioral patterns on each day.

To find significant periodicity in the collective human mobility, we searched for τ that maximizes the periodicity probability at each location separately. Figures 3.3(c) and 3.3(d) show two sample locations where periodic behavior can be observed. The small peaks in these plots reveal relatively high probability that these particular locations were visited regularly with the time period τ, when τ is in multiples of 24 hours. Moreover, the maximum predictability probabilities are found at multiples of 168 hours. Without doubt, this clearly indicates that *daily* and *weekly* behaviors exist in the collective mobility data. Using a more algorithmic method to find significant period τ, the Fourier analysis also suggested that $\tau = 24$ hours and 168 hours correspond to two of the most significant frequencies of $\approx 4.167 \times 10^{-2}$ Hz and $\approx 5.925 \times 10^{-2}$ Hz, respectively.

In the next section, we analyze the possibility of the collective human mobility being predictable with periodic behavioral patterns.

3.4 Predictability of the Periodic Model

Intuitively, the periodicity $P_x(\tau)$ can be assumed to have estimated the precision of a periodic-based predictive model, which is based on a strong assumption of periodically repeated visitations. Hence, the periodicity $P_x(\tau)$ can be considered as a measurement for the predictability of the periodic model. In addition, we want to provide another predictability analysis applying an academic concept from information theory to the periodic model.

First, we assign *periodic entropy* to the history data of repetitive visitations at each location to determine the amount of information needed to foresee future visits given historical records of repetitive visitations. At each location x, the periodic entropy is computed as follows.

Definition 3.4.0.1. *Given the collective mobility data, the entropy S_x^τ that quantifies the degree of uncertainty of the periodicity $P_x(\tau)$ in the dataset is as follows:*

$$S_x^\tau = \sum_{\nu \in \{0,1\}} P(v_x) H(v_x(t'_{i+\tau} | v_x(t'_i) = \nu)),\ t'_i \in \{t'_0, \ldots, t'_{m-1}\}, \tag{3.4.1}$$

where $P(v_x)$ is the probability of a location x being visited, and the conditional entropy $H(v_x(t'_{i+\tau})|v_x(t'_i) = \nu)$ is

$$H(v_x(t'_{i+\tau})|v_x(t'_i) = \nu) = \sum_{\varphi \in \{0,1\}} P(\varphi|\nu) \log\left(\frac{1}{P(\varphi|\nu)}\right), \tag{3.4.2}$$

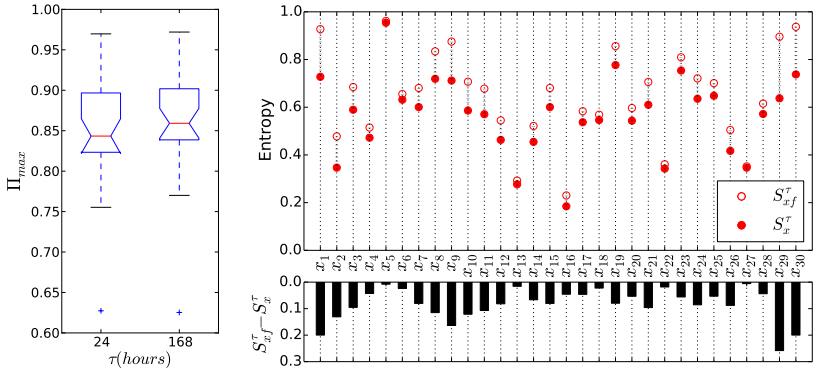
where $P(\varphi|\nu)$ stands for $P(v_x(t'_{i+\tau}) = \varphi | v_x(t'_i) = \nu)$.

In addition, let S_{xf}^τ be the entropy of future visitations; i.e.,

$$S_{xf}^\tau = -\sum_{\varphi \in \{0,1\}} P(v_x(t'_{i+\tau}) = \varphi) \log(P(v_x(t'_{i+\tau}) = \varphi)),\ t'_i \in \{t'_0, \ldots, t'_{m-1}\} \tag{3.4.3}$$

Next, we determine the predictability for each location x of the periodic model with the probability $\Pi_{x,\tau}$ defined as follows.

Definition 3.4.0.2. *Let $\Pi_{x,\tau}$ be the probability that the periodic model predicts times of future visitations at x correctly by always predict visits at all times that are $k\tau$ apart from the last visit for $k = 1, 2, \ldots$. Thus the associated entropy $H(\Pi_{x,\tau})$ of the predictability*



(a) Predictability of periodic models.

(b) Differences between the periodic entropy and the entropy of future visits at each location.

Figure 3.4: Predictability Π_{τ}^{max} and corresponding periodic entropy

$\Pi_{x,\tau}$ *is as follows:*

$$H(\Pi_{x,\tau}) = -\Pi_{x,\tau}\log_2(\Pi_{x,\tau}) - (1-\Pi_{x,\tau})\log_2(1-\Pi_{x,\tau}). \tag{3.4.4}$$

The maximum predictability $\Pi_{x,\tau}^{max}$ can be determined using Fano's inequality in accordance with Eq. (3.2.2).

$$S_x^{\tau} \leq H(\Pi_{x,\tau}) + (1-\Pi_{x,\tau})\log_2(N-1). \tag{3.4.5}$$

Because $\Pi_{x,\tau} \leq \Pi_{x,\tau}^{max}$ and $N = 2$ prevents this boundary to the binary classification, the following correction is required.

$$\begin{aligned} S_x^{\tau} &\leq H(\Pi_{x,\tau}) + (1-\Pi_{x,\tau})\log_2(N-1) \leq H(\Pi_{x,\tau}) + (1-\Pi_{x,\tau})\log_2(N) \\ &= -\Pi_{x,\tau}^{max}\log_2(\Pi_{x,\tau}^{max}) - (1-\Pi_{x,\tau}^{max})\log_2(1-\Pi_{x,\tau}^{max}) + (1-\Pi_{x,\tau})\log_2(N). \end{aligned} \tag{3.4.6}$$

After solving for $\Pi_{x,\tau}^{max}$ in Eq. (3.4.6), the predictability $\Pi_{x,\tau}^{max}$ determines the upper limit of the probability of predicting future visits of people at location x in the far future given an appropriate periodic model (with the time period τ). We evaluated S^{τ} and Π_{τ}^{max} separately for each location, and the associated distribution of Π_{τ}^{max} is shown in Figure 3.4(a). Both distributions of the predictability Π_{τ}^{max} indicate the average predictability over all locations is approximately greater than 80%, in both daily and weekly models. The average predictability of the weekly model is slightly higher and has lower variance than the daily model. It is reasonable to conclude that the weekly model fits the collective mobility data better than the daily periodic model.

Figure 3.4(b) shows differences between the periodic entropy, S_x^τ, and the entropy of future visitations, S_{xf}^τ, at each location x when $\tau = 24$ hours. Note that as S_x^τ is closer to zero and farther from S_{xf}^τ, future visitations are more likely to periodically depend on previous visitations. For instance, locations x_5, x_6, x_{17}, x_{18}, and x_{28}[1] were not periodic, while the locations $x_1, x_2, x_8, x_9, x_{29}$ and x_{30} appeared to be more periodic than others. Therefore, the periodicity-based predictive model alone would not work in all locations; hence, we have developed the integrated aperiodic and periodic model for long-term human mobility prediction.

3.5 Conclusion

Human mobility and activities monitoring technologies have been improved notably in the past decade. Many advance non-intrusive techniques, such as WiFi signal strength-based techniques [18, 19, 51] and laser-based tracking [14], have been developed and made available for human behavior researches. Similarly, we aim at unobtrusiveness and simplicity and opt for the ambient sensors, in which tracking sensitivity is relatively higher than the other methods. In addition, the hierarchical probabilistic model and statistical method proposed in [47] are also capable of applying to non-intrusive sensing data and, yet, are able to recognize multi persons' behaviors. The limitation that these techniques are facing, however, is that they cannot distinguish individual identities. Consequently it is impossible to create an individual predictive model for each individual's mobility pattern. That is, despite advance acquisition techniques, the obtained data are still mixed up and inseparable among multi users.

In conclusion, the answer to the question about limitations of the predictability of collective human mobility from simple ambient sensor data collected in a smart environment has been found. The dataset collected from such ambient systems appeared to be noisy and did not contain much useful information for prediction tasks at first. After sufficient preprocessing with sequential patterns mining, our analyses have shown the potential predictability of the dataset in both short-term prediction setting and long-term prediction setting. Moreover, we have also discovered the key factors that greatly affect the accuracy, which give us some insight on how and where to be careful with when preprocessing the dataset.

For short-term prediction, the length and the support of the trajectory pattens are

[1] Placement of each point of interest x_i can be found in Figure 3.1(a)

the key factors that most influence the predictability of the dataset. In other words, trajectory patterns with more evidences to support are less likely to be observed from actual moving subject rather than from noise or false alarm sensor triggers. And longer patterns contain more more information about past mobility patterns (i.e., paths and speeds) compared to shorter patterns.

For long-term prediction, it is straightforward that the likelihood of repetitive patterns, either periodic or aperiodic, is the most important feature that determines the limit of its predictability.

Chapter 4

Human Mobility Prediction

4.1 Long-term Human Mobility Prediction

The proposed Aperiodic and Periodic model (APP) for long-term human mobility prediction [65] combines two paradigms. The first approach (periodic approach) employs the periodic property in human mobility to foresee future visits. The second approach (aperiodic approach) does not rely on the periodicity; rather, it presumes that mobility patterns are similar to a day in the past that has similar features. APP model uses one of the two approaches to predict human mobility at a certain location x depending on the periodicity probability $P_x(\tau)$ at that specific location. If $P_x(\tau)$ is more than the user-specific threshold P^{τ}_{min}, then APP uses the periodic approach. Otherwise, it switches to the aperiodic approach.

4.1.1 The Periodic Approach

APP model's periodic predictive approach was designed to foresee times of future visitations at each location in the smart space. To predict future locations of multiple people, the predictions are computed independently for each location. Then all the results are combined to provide a set of locations that are likely to be visited at the specific time in the far future.

The fundamental idea behind the prediction is based on the assumption of periodicity. If the visitations at x recur regularly and repeatedly with constant time interval τ and if this periodic behavior appears consistently over time, then the probability $P^{\tau}_x(t'_f)$ that the future visitation will occur within the time frame t'_f, when the last visit

happened at t'_{m-1} can be computed as

$$P_x^\tau(t') = P(v_x(t'_{m-1-(k)\tau+\delta}) = 1 | v_x(t'_{m-1-(k+1)\tau+\delta}) = 1),\ k = 1, 2, \ldots, \lfloor m/\tau \rfloor \quad (4.1.1)$$

where $\delta = \tau - (f - m + 1) \bmod \tau$.

This simple, yet accurate, predictive approach works well only at locations, where individuals' mobility has apparent periodicity. Otherwise, the periodic approach obtains poor predictions because mobility in those particular locations is not governed by periodic behavior. To address this problem, we proposed the optional aperiodic approach, which is independent of the periodic behavior.

4.1.2 The Aperiodic Approach

In the second predictive technique implemented in APP method, we extract significant patterns of repetitive visitations that occurred on different days at each location. Next the days that have a similar visiting behavior pattern are clustered, resulting in groups of *similar days*. Then, we extract contextual features from each group of similar days. Due to the limited dataset available, in this project only two features of interest were considered: (1) day of the week, (2) whether or not it was a holiday. Note that unlimited additional features, such as temperature, traffic, weather conditions, or meeting schedule, that might relate to the mobility pattern can be used to more comprehensively characterize the day.

The intuition that supports this predictive approach is derived from the weekly model presented in Section 4.1.1, i.e., human mobility patterns on the same *day of the week* are likely similar. In addition, human activities on national holidays are apparently different from normal workdays; therefore, we need an additional bit to explicitly specify this property. Hence, the mobility pattern of *a day* can be individually modeled by the visitations at each location. Recall the temporal sequence v_x described in Section 3.1; mobility at a certain location x can be represented by a vector:

$$\begin{aligned} d_x =& [v_x, day_{week}, holiday] \\ =& [v_{t'_0}, \ldots, v_{t'_{23}}, Sun, Mon, \ldots, Sat, Hol]. \end{aligned} \quad (4.1.2)$$

The day vector d_x consists of 32 bits. The first 24 bits model visitations at location x during a specific time frame of a day, which is divided into hours. The next 7 bits

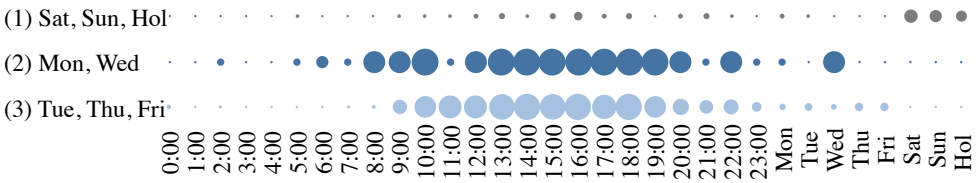


Figure 4.1: Three cluster centroids that represent three mobility patterns.

indicate day of week, and the last bit indicates a holiday.

Similarity between two day vectors is generally measured by the Hamming distance [26]. Then, the k-means clustering algorithm [42] is applied to a set of day vectors to find clusters of similar days. Note that the clustering considers only the first 24 bits, the visitation part of day vectors. The parameter k of the algorithm directly implies the number of different mobility patterns that occurred on different days. The centroid of each cluster now represents a common mobility pattern that provides predictions (probability) of visitations on particular future days that have similar features. A concrete example of similar day clusters from the real dataset is shown in Figure 4.1. The cluster centroids in Figure 4.1 clearly show three different visiting patterns at that particular location. Cluster (1) contains a set of days in the past history when visitations rarely happened, and the majority of this set are Saturdays, Sundays, and days specified as holidays. On the other hand, clusters (2) and (3) contain more active days. The days in cluster (2), primarily Mondays and Wednesdays, have very low visitations records from 11.00–12.00 and 21.00–22.00; moreover, the visitations seem to occur earlier than on the days in cluster (3). Interestingly, this follows from the fact that we have scheduled meetings in the experimental space every Monday and Wednesday, which causes the mobility pattern on these days to be different from other days.

sectionEvaluating Prediction Performance In this section, we report on an evaluation of the prediction performance of the proposed long-term human mobility predictor on a physical collective human mobility dataset accumulated inside the working environment. As described in Section 3.1, the dataset contains 92 days of mutual movements of all participants. Data were collected consecutively 24 hours a day, 7 days a week from approximately 20 participants using infrared sensors and magnetic sensors. These sensors were installed at 30 locations over the experimental space to detect activities and mobility at each area (Figure 3.1(a)). Movements and activities were not scripted beforehand; all actions occurred spontaneously or deliberately in relation to each indi-

Table 4.1: Human Mobility Dataset

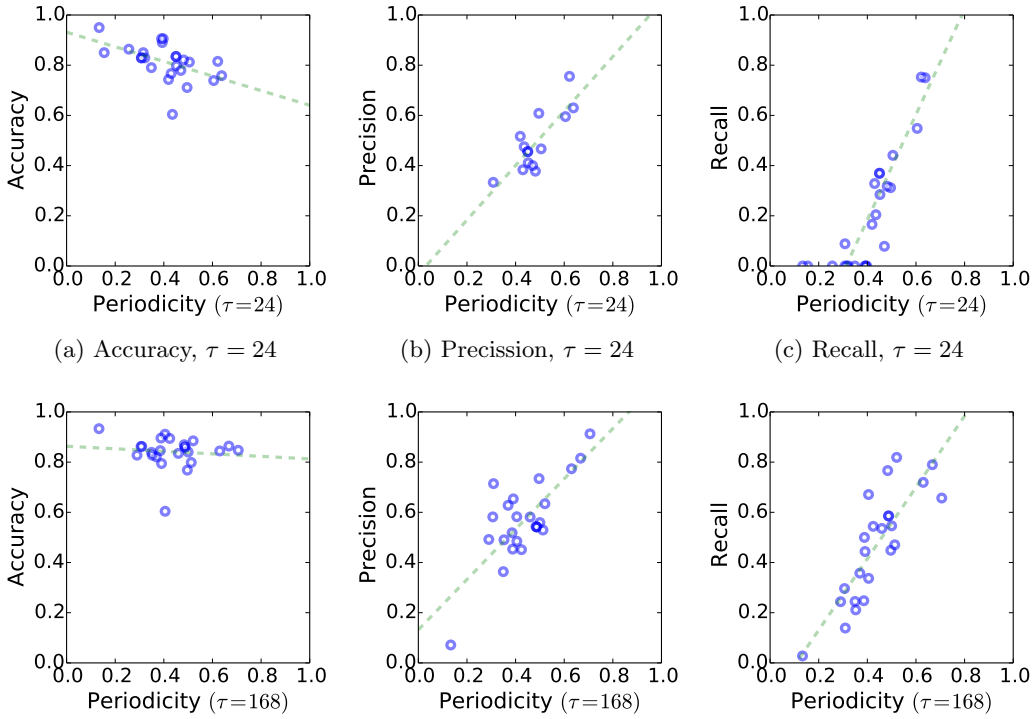
Dataset	Training	Test
Sample rate	hourly	
Number of participants	≤ 20	
Observed locations	30	
Size of data	62 days (1,488 hours)	30 days (720 hours)

vidual's routine, work schedule, and needs at that instant.

First, we evaluated the periodic approach for long-term human mobility prediction. Two months of collective mobility data was used to build the predictive model and the remaining 30 days of mobility data were used to test the model. Details of the dataset are summarized in Table 4.1.

4.1.3 Periodicity and Prediction Performance

We determined relations between the periodicity probability ($P_x(\tau = 24)$ and $P_x(\tau = 168)$) and the prediction accuracy precision and recall rate at each location separately. Dashed lines drawn in Figure 4.2 estimated predictive performance as a linear function of the periodicity at each location of interest. Figures 4.2(a) and 4.2(d) exhibit a decreasing trend of prediction accuracy with increasing periodicity probability; however, the periodic predictor returns higher precision and recall rates as the dataset has higher probability of such movements being periodically repeated. Nevertheless, the measurement of prediction accuracy is meaningless to us because the datasets, which contain historical visitations records for each location, have negative skew. In other words, a naive predictor could achieve at least a 60% chance of correctly predicting visitations (either *"visited"* or *"not visited"*) at a specific time frame in the future by always guessing *"not visited"*. Figures 4.2(b) and 4.2(e) show a direct relationship between the precision rate and periodicity probability. Likewise, the recall rates in Figures 4.2(c) and 4.2(f) show that the datasets with higher periodicity are more predictable than others. Moreover, when the periodicity probabilities are lower than 0.4, the daily periodic approach (Figure 4.2(c)) clearly returns poor results, i.e., those visitations were not daily periodic and were hardly covered by the weekly periodic model. These results confirm our hypothesis that the periodic approach alone is not effective for predictions with low periodicity probability.



(a) Accuracy, $\tau = 24$ (b) Precission, $\tau = 24$ (c) Recall, $\tau = 24$

(d) Accuracy, $\tau = 168$ (e) Precission, $\tau = 168$ (f) Recall, $\tau = 168$

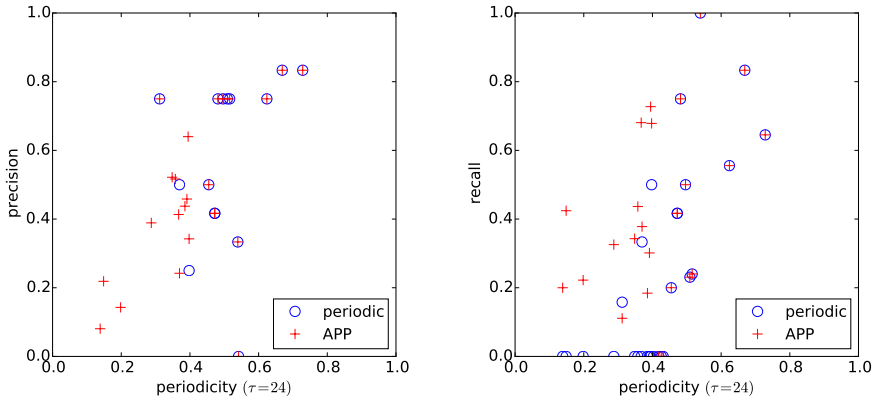
Figure 4.2: Periodicity and prediction performance

4.1.4 Prediction Performance of the Aperiodic Approach

The aperiodic part of APP is implemented with the similar-day predictive approach described in 4.1.2. In previous experiments, the periodic approach underperformed on the datasets where mobility was not really periodic. This was particularly the case for the daily periodic model (see Figures 4.2(a), 4.2(b), and 4.2(c)) when most locations in the experimental space had periodicity probability lower than 0.4. Hence, in this experiment, the aperiodic part of APP is activated when periodicity is lower than the minimum threshold of $P^{\tau}_{min} = 0.4$.

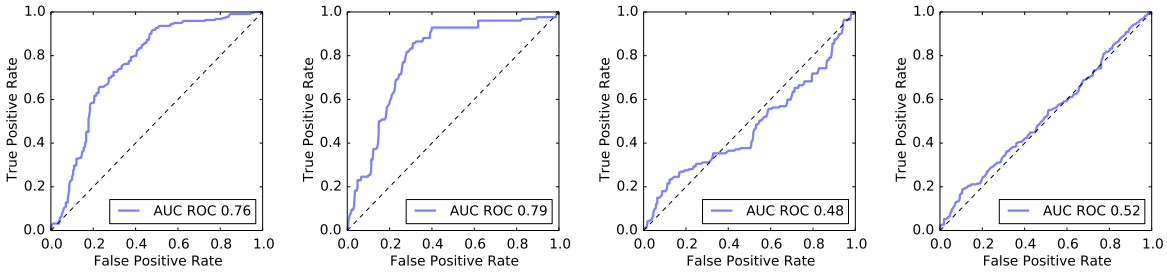
Figure 4.3 reveals the benefit of including the aperiodic component in the APP predictive model. In Figure 4.3(a), the precision rates of the APP model after the implementation of the similar-day approach for low-periodicity data are noticeably improved compared to the periodic approach alone. The precision plots of the periodic approach on the left (periodicity ≤ 0.4) were mostly omitted from analysis because the periodic predictor never predicted *"visited"* for those locations, resulting in undefined precision rates.

APP also improves the recall rates, as shown in Figure 4.3(b). It is interesting



(a) Periodicity and Precision (b) Periodicity and Recall

Figure 4.3: Prediction performance of the similar-day approach

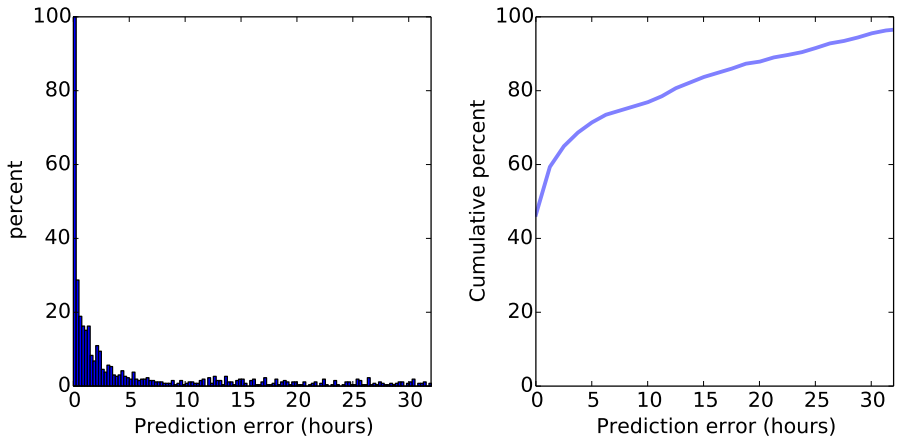


(a) AUROC x_1 (b) AUROC x_{19} (c) AUROC x_5 (d) AUROC x_{28}

Figure 4.4: Area under the ROC curve

that the recall rates used to achieve close to 0.0 with the periodic approach increase to 0.6 when the APP predictive technique is employed. Samples of prediction evaluations of the two most accurate and the two least accurate predicted locations are shown in Figure 4.4. In the most accurate case, APP was able to obtain the area under the receiver operating characteristic (ROC) of 0.79, where the sensitivity of the decision threshold was varied (Figure 4.4(b)). Nonetheless, the worst predictor scores occurred for a ROC of 0.48 (Figure 4.4(c)).

In this final evaluation, we measured how close the predicted visitations are to the actual visits. The prediction error is simply the distance between the predicted timestamp of an expected future visit and the timestamp of the closest actual future visit. The results are summarized in Figure 4.5. Impressively, 60% of the estimation errors are less than 2.5 hours (with mean = 7.5 hours, and median = 1.5 hours), considering that the prediction was made a month in advance.



(a) Histogram of prediction error

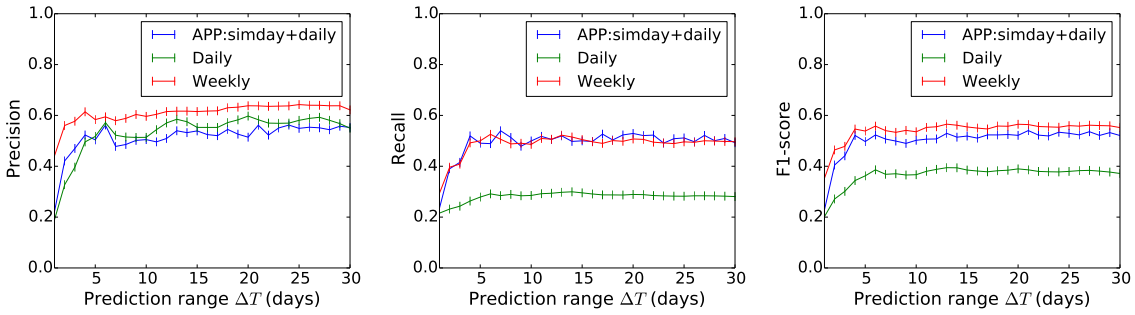
(b) CDF of prediction error

Figure 4.5: Distribution of prediction error

In summary, the aperiodic part in our proposed long-term human mobility predictive technique improves prediction performance, particularly when the periodicity probability is too low to infer future visitations. However, the similar-day approach in the aperiodic part is not sufficiently effective to improve the predictive technique that employs the weekly periodic approach ($\tau = 168$). The reason for this is that the *day of week* feature resulted from cluster analysis in the similar-day approach corresponded to the weekly periodic model, and *holiday* is not a significant feature since there were few holidays during the three months when the dataset was collected. Implementation of the similar-day approach (aperiodic part) and the weekly periodic approach did not achieve noticeable improvement compared with implementation of only the weekly periodic predictive approach.

4.1.5 Long-term Prediction Performance

In this section, we report the results of testing the robustness of APP over a long-term. (Details of the dataset are summarized in Table 4.1.) Prediction performance for each day was summarized and plotted across a prediction range of 30 days. The results presented in Figure 4.6 show steady prediction performances even when predicting for 30 days in the future. The F_1 score, which is the harmonic mean of the precision and the recall rate (Figure 4.6(c)), summarizes the prediction performance of the three proposed predictive techniques as follows. First, the collective mobility dataset that initially appears to be random contains sufficient information to enable accurate predictions even



(a) Precision (b) Recall (c) F_1 score

Figure 4.6: Long-term prediction performance

in the far future. Activities and corresponding mobility in the dataset are likely to be periodic on a weekly basis; hence, the weekly periodic predictive approach alone can achieve an average F_1 score of 0.55. On the other hand, the daily model performs relatively poorly (average F_1 score of 0.37) with this dataset because of the low periodicity probability on a daily basis. However, after implementing the similar-day approach together with the daily predictive model, the integrated model can achieve an average F_1 score of 0.52.

4.2 Short-term Mobility Prediction

In our earlier attempts in modeling and predicting human mobility using the dataset from the same set of sensors used in this project [68], we have succeeded in predicting future whereabouts of a moving participant with the average precision of 90% using a decision tree-based classifier trained on sequential patterns of users' trajectories. However, the result from our recent analysis [66] (details were summarized in Chapter 3) on the theoretical limit of the predictability of human mobility given the dataset from the same selection of sensors has revealed the potential of achieving a near perfect prediction. That being said, despite the possibility of missing and other failures that could happened and affect the quality of sensor reading, we have shown that, with essential preprocessing step, the sensor data contains sufficient information to train a machine learning model and get highly accurate predictive model.

Our previous predictive technique [68] relied on the decision tree learning, which has a few advantages over other classification approaches for exploratory analysis. Model of a decision tree has a structure that is easy to understand, even to people with little

background in machine learning, and features importance can be analyzed directly from the model for broader understanding of mobility patterns.

However, decision tree learning has a couple of downsides which can greatly affect the accuracy of the prediction. Firstly, decision tree-based learnings are related to the problem in which two (or more) variates explain the same thing. A decision tree will extend its branch on the best variable, whereas other approach will consider them both. Secondly, in a decision tree, all variates are assumed to interact. That is all variates are forced to interact with every variables further up in the tree. The assumed interactions that two variates behave dependently can greatly degrade the predictive accuracy. Thirdly, poor resolution with continuous variables.

The proposed method is trajectory pattern-based with the information of transition times incorporated into the model to take into account moving speed which has the potential to identify users' identity and intentions.

4.2.1 Extracting Trajectories from Sensor Readings

Representation of a trajectory used by this algorithm is similar to the sequential representation described in Section 3.1. Regardless of different types of sensors, we want each sensor to report a binary state of its vicinity being visited by a user (or users). Then these states of visitations triggered from different sensors are sorted by their time of visits resulting in a sequence of visited location IDs (sensor IDs) and time stamps that represent movements. We cut the continuous sequence of sensor readings at the intervals with longer duration than 30 seconds resulting in shorter sequences in which each sequence represent individual trajectory. Specifically, a trajectory is represented with a sequence of visitation tuples. Each visitation tuple contains 1) visited location ID s_i and 2) visited time t_i. For instance, the movement demonstrated from area A to area D in Fig. 4.7 can be represented with the following sequence of length-4.

$$tr_{ABCD} = \langle (\boldsymbol{x_a}, t_1), (\boldsymbol{x_b}, t_2), (\boldsymbol{x_c}, t_3), (\boldsymbol{x_d}, t_4) \rangle \tag{4.2.1}$$

While movement speed can be determined simply by transition times between each hop along the path.

$$ti_{ABCD} = \langle ti_{ab}, ti_{bc}, ti_{cd} \rangle \tag{4.2.2}$$

As stated earlier in the introduction, we have to handle the problem of inaccu-

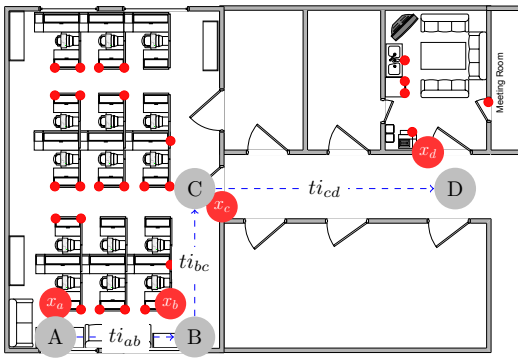


Figure 4.7: Placement of each sensor is indicated with read circle. The trace shows an example trajectory of a person traveling inside the environment from area A to area D. Each transition time ti_{ij} implies moving speed.

rately generated trajectories from noises and collateral movements. Similar to previous works [68, 65], we applied the temporal sequential pattern mining algorithm to extract spatial-temporally frequent movement patterns from the original dataset, i.e. frequent in term of both speed and direction. The resulting trajectory patterns are expected to be far less noisy and more predictable, according to the analysis in [66].

4.2.2 Trajectory Patterns-based Human Mobilility Predictive Model

Future location of a moving user is dependent on the most recent visits history, with respect to moving speed and chosen route. Trajectory patterns-based models have been studied in a number of ways, such as Markovian model [70] and Spatio-temporal sequential pattens-based model [59]. In this work, we opted for the trajectory patterns-based method, which is a standard sequential patterns-based predictive model which is formally defined as the following steps:

1. Given a set of trajectories $\Gamma = \{tr_1, \ldots, tr_N \mid \forall |tr_i| \geq 2\}$, let $\tilde{\Gamma}$ be the mined set of frequent sub-trajectories that occur at least $supp_{min}$[1] times in Γ with the minimal transition probability of $conf_{min}$[2]. Specifically, we used the PrefixSpan algorithm [27] for frequent trajectories mining.

2. Let the transition probability $P_{tr}(n \mid m)$ be the probability of finding that the next visitation is at n given that m is the most recent trajectory of the user as

[1]The minimum support threshold
[2]The minimum confidence threshold

follows

$$p^{tr}(n \mid m) = conf(m \to n) = \frac{supp(m \to n)}{supp(m)}, \tag{4.2.3}$$

where $supp(m \to n)$ denotes the number of occurrence that the trajectory m were followed by a visitation at n and $supp(m)$ is the number of occurrence of m in Γ. Note that $supp(\tilde{tr}_i) \geq supp_{min} > 0$ and $conf(\tilde{tr}_i) \geq conf_{min} \geq 0,\ \forall \tilde{tr}_i \in \tilde{\Gamma}$.

3. For each length-l frequent trajectory $\tilde{tr}_i \in \tilde{\Gamma}$, define $\tilde{tr}_i^{[0]}$ as the first tuple of the trajectory $\tilde{tr}_i$ and let $\tilde{tr}_i^{[-1]}$ be the first tuple from the end of the trajectory $\tilde{tr}_i$. $\tilde{tr}^{[-2]}$ is the second element from the end, and so forth. Assign a notation $\tilde{tr}_i^{[a:b]}$ to denote the sub-trajectory of $\tilde{tr}_i$ from the ath tuple to the $(b-1)$th tuple (inclusive), where $a < b$. Apparently, $\tilde{tr}_i^{[0:l]}$ equals to $\tilde{tr}_i$ itself. Further, to make the notation less cluttered, we would like to abbreviate $\tilde{tr}_i^{[0:b]}$ to just $\tilde{tr}_i^{[:b]}$; likewise, $\tilde{tr}_i^{[a:l]}$ is equal to $\tilde{tr}_i^{[a:]}$. For instance, if $\tilde{tr}_i = \langle (x_a, t_1), (x_b, t_2), (x_c, t_3), (x_d, t_4) \rangle$ then the destination is $\tilde{tr}_i^{[-1]} = (x_d, t_4)$ and its corresponding prefix is $\tilde{tr}_i^{[:-1]} = \langle (x_a, t_1), (x_b, t_2), (x_c, t_3) \rangle$, while $\tilde{tr}_i^{[1:]} = \langle (x_b, t_2), (x_c, t_3), (x_d, t_4) \rangle$.

   Given a length-l test trajectory tr', the prediction step works as follows:

   (a) Create a candidate set C containing all the frequent trajectories in which their prefixes are identical to tr', i.e. $C = \{\tilde{tr}_i \mid \forall \tilde{tr}_i \in \tilde{\Gamma} \wedge \tilde{tr}_i^{[:-1]} = tr'\}$. The equality in this sense concerns only the spatial context of the trajectories. Two trajectories are considered equal if every visitations at each step are similar, regardless of the difference of the visited times or the time spent at each visit.

   (b) If the set C is empty and $|tr'| > 1$, repeat step 3a with a shorter tr', that is the new $tr' \leftarrow tr'^{[1:]}$, which results in a shorter test trajectory that disregards the fist visit of the trajectory.

   (c) In the case that step 3a and 3b are repeated until the test trajectory cannot be trimmed anymore i.e. $|tr'| = 1$, the model returns an empty set, which indicates that the model cannot recognize the query trajectory and cannot provide a proper prediction. In other words, the probabilities of all possible outcomes are, evenly, zeros.

   (d) Otherwise, in the case that C is not empty, the model returns a distinct set, which contains the predicted locations and the confidence of each prediction.

Algorithm 4.2.2.1 Trajectory Patterns-based Prediction
(transition time concerned)

procedure PREDICT_PROBA_NNB($tr, ti, \tilde{\Gamma}$)
  $C \leftarrow \varnothing$
  while C is $\varnothing \wedge |tr'| \geq 1$ **do**
    $C \leftarrow \{\tilde{tr}_i \mid \forall \tilde{tr}_i \in \tilde{\Gamma} \wedge \tilde{tr}_i^{[:-1]} = tr'\}$
    $tr' \leftarrow tr'^{[1:]}$
  end while
  if $C \neq \varnothing$ **then**
    $P^{NNB} \leftarrow \{(\tilde{y}, p^{NNB}(\tilde{y}_i \mid ti)) \mid \forall \tilde{y} \in$ Possible outcomes$\}$ ▷ equation ((4.2.5))
    return $\text{argmax}_{\tilde{y} \in \text{Possible outcomes}} P^{NNB}$
  else
    return $\varnothing$
  end if
end procedure

Although the introduced trajectory patterns-based model is able to provide the prediction probability properly, the temporal context of a trajectory has not been utilized in the prediction process yet. Thus we introduce a transition time concerned version of trajectory patterns-based model wherein the prediction probability of each potential location is computed based on the expected transition time between the last visit and the destination. That is rather than only asking: *"Where is the user expected to be next, given his most recent movement trajectory tr'"*, we would add *"Where is the user expected to be ti after?"*.

We keep the modification simple. In step 3d, when C is not empty, final transition times

$$ti_{\tilde{tr}_i} = |\tilde{tr}_i^{[-1]}.t - \tilde{tr}_i^{[-2]}.t| \tag{4.2.4}$$

of each candidate trajectory $\tilde{tr}_i \in C$ are computed, where $\tilde{tr}^{[k]}.t$ accesses the timestamp property of a tuple $\tilde{tr}^{[k]}$. Then the prediction probability of each possible outcome $\tilde{y}$ is estimated based on soft nearest neighbor classification [60]. That is

$$p^{NNB}(\tilde{y} \mid ti) = \frac{\sum_{\tilde{tr}_i \in C \wedge \tilde{tr}_i^{[-1]} = \tilde{y}} \exp(-(ti - ti_{\tilde{tr}_i})^2)}{\sum_{\tilde{tr}_i \in C \wedge \tilde{tr}_i^{[-1]} \neq \tilde{y}} \exp(-(ti - ti_{\tilde{tr}_j})^2)}, \tag{4.2.5}$$

where tr' is the most recent trajectory of a moving user and ti is the given time interval between tr' and the next visitation. The prediction is then selected from the most likely position as follows:

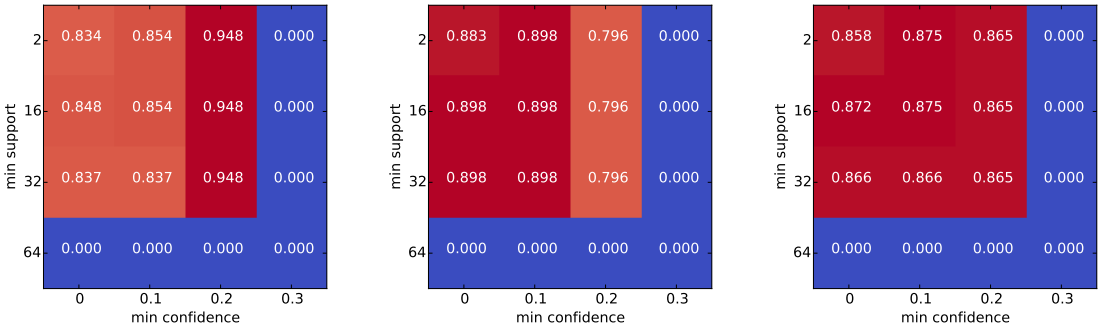
$$\text{Predicted location} = \text{argmax}_{\tilde{y}}\ p^{NNB}(\tilde{y} \mid ti) \tag{4.2.6}$$

The complete procedure of the transition time concerned trajectory patterns-based model is elaborated in Algorithm 4.2.2.1.

4.2.3 Performance Evaluation using Real-world Mobility dataset from Smart Environment

In this section, we compared the proposed trajectory patterns-based transition time concerned model with our prior attamps at predicting human mobility [15] (denoted as DT). The predictive technique used in [15] was based on a decision tree classifier. The decision tree-based human mobility predictor takes $n-1$ previous visited steps, $n-2$ transition times between each step and the expected time until the last step and the predicting location as features for length-n trajectories. To train the model, we divide training data by length and train the decision tree separately for each length-i trajectory, where $i = 2, \ldots, 4$. Both decision tree-based approach and the proposed method rely on the training trajectories to be provided in good quality. To extract real trajectories from noises and other sensor events other than human mobility, we employed the temporal sequential patterns mining algorithm [15] with the minimum support of 2 and the minimum confidence of 10% to the raw sensor data to connect sensor events, remove spurious sequences and extract only a set of trajectories that meet the requirement stated above.

We tested two competing human mobility predictive techniques and evaluated their predictive performance using 10-fold cross validation. Referred to the dataset of mobility tracked from 30 interested locations mentioned in Section 3.1, sequence of sensor firing events were split into set of sequences with the 30 seconds threshold, i.e. events that are over 30 seconds apart are considers unconnected events). Then the set of events sequences equally divided into 10 partitions, where 9 of them will be used to train the predictive model and the remaining partition is kept for testing. The processes is repeated 10 times, with different testing partitions. Note that the test set can also contain noises, such as false positive triggering and interleaving sequences due to two or more users walked at the same time, because we partitioned raw data into train and test sets without any preprocessing. Therefore, to cleanup testing datasets, we applied temporal sequential patterns mining algorithm [67] to extract frequent temporal sequential patterns that appeared at least five times with the minimum confidence of 35% within the test set; namely, $supp_{min} = 5$ and $conf_{min} = 0.35$. The test dataset



(a) precision (b) recall (c) F_1 score

Figure 4.8: Effects of minimum support and confidence parameter settings on prediction accuracy

contained 3,771 trajectories of varying lengths and only 19 locations from 30 locations of interest appeared as destinations of the testing trajectories.

Both decision tree-based predictor and the trajectory patterns-based transition time concerned predictor rely heavily on the quality of the input trajectory patterns. That is the higher the confidence, the more precision score the predictors could obtain. In contrary, the lower the confidence and the lower the support, the more recall score the predictors could obtain. Therefore, the right balance between precision and sensitivity must be fine tuned to suit each applications. For instance, in applications where false positive are not as severe as false negatives, we would want to lower the minimum confidence and support parameters to have our predictors trained on more generalized patterns. Whereas, if false positives cost more than false negatives, we would want out predictors only learn from highly confident patterns. As shown in Figure 4.8, prediction accuracy of an interested location x was measured with the precision, recall and F_1 scores. From Figure 4.8(a), the precision increased as we incremented the minimum support and confidence parameters, until the predictor failed since the minimum support and/or the minimum confidence were too high such that the mined patterns did not contain any trajectory patterns towards x. In contrary, Figure 4.8(b) shows that as the minimum support and confidence parameters were decreasing, recall rate also decreased because of the mined patterns contained less and less patterns that can be used to predict x.

Since we do not have specific preference of whether to emphasis on the precision or sensitivity, we chose to maximize the harmonic mean of both, namely the F_1 score.

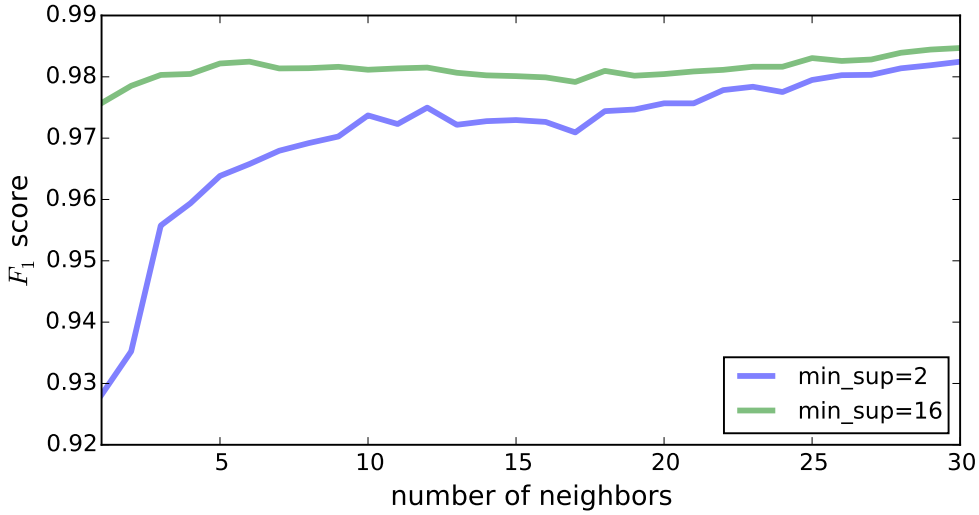


Figure 4.9: Number of nearest neighbors k and prediction accuracy

From Figure 4.8(c), we picked the minimum support parameter of 16 and the minimum confidence parameter of 0.1 to extract trajectory patterns that will be used to train our predictors.

Other than the support and confidence parameters, the proposed trajectory patterns-based transition time concerned model has an additional parameter k which is used to control the number of nearest patterns that will be considered to make a prediction. Smaller k limits the model to consider only a few nearest patterns, whereas Larger k allows the model to consider more candidate patterns. In general, it is important to find the right k so that the model would not be too general (underfitting) or too specific to the training patterns (overfitting). This is true for our method only when the input patterns are mined with low minimum support and confidence settings, where infrequent patterns that possibly contain noise signals might still be included in the training patterns. We can see from Figure 4.9, in which the training patterns were mined from the TSPMalgorithm with the minimum support setting of 2 and no minimum confidence limit, that the accuracy of the model (as measured by the F_1 score) was heavily relied on the k parameter. Since the training patterns contained more infrequent patterns, it needed more number of candidates to make good prediction. In contrast, when the minimum support and confidence parameters were set properly, the parameter k didn't affect the accuracy of the model as much.

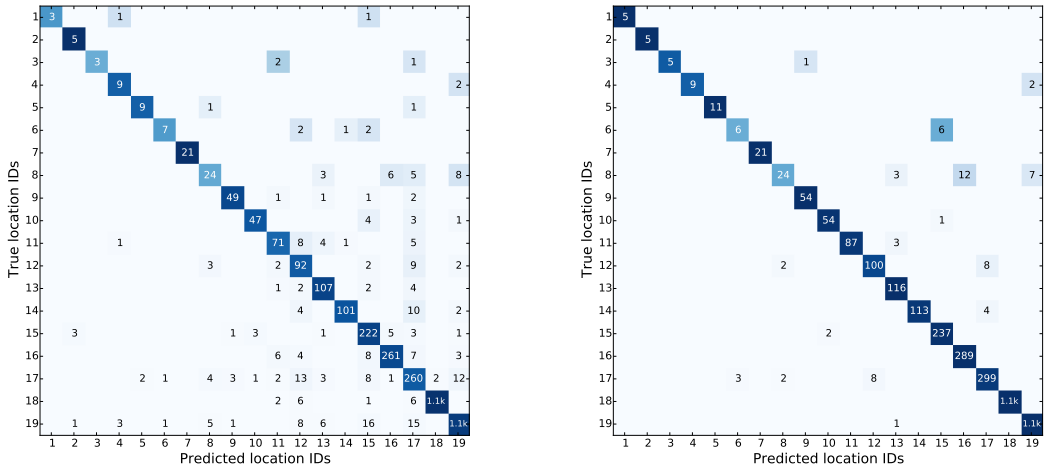
Next, we set the minimum support to 16 and the minimum confidence to 0.1. Prediction performance of the proposed method and a competition were evaluated using the

Loc IDs	# of visits	Precisions		Recalls		F_1 scores	
		DT	TBTT	DT	TBTT	DT	TBTT
1	5	**1.0000**	**1.0000**	0.6000	**1.0000**	0.7500	**1.0000**
2	5	0.5556	**1.0000**	**1.0000**	**1.0000**	0.7143	**1.0000**
3	6	**1.0000**	**1.0000**	0.5000	**0.8333**	0.6667	**0.9091**
4	11	0.6429	**1.0000**	**0.8182**	**0.8182**	0.7200	**0.9000**
5	11	0.8182	**1.0000**	0.8182	**1.0000**	0.8182	**1.0000**
6	12	**0.7778**	0.6667	**0.5833**	0.5000	**0.6667**	0.5714
7	21	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
8	46	0.6486	**0.8571**	**0.5217**	**0.5217**	0.5783	**0.6486**
9	54	0.9074	**0.9818**	0.9074	**1.0000**	0.9074	**0.9908**
10	55	0.9216	**0.9643**	0.8545	**0.9818**	0.8868	**0.9730**
11	90	0.8161	**1.0000**	0.7889	**0.9667**	0.8023	**0.9831**
12	110	0.6619	**0.9259**	0.8364	**0.9091**	0.7390	**0.9174**
13	116	0.8560	**0.9431**	0.9224	**1.0000**	0.8880	**0.9707**
14	117	0.9806	**1.0000**	0.8632	**0.9658**	0.9182	**0.9826**
15	239	0.8315	**0.9713**	0.9289	**0.9916**	0.8775	**0.9814**
16	289	0.9560	**0.9601**	0.9031	**1.0000**	0.9288	**0.9797**
17	312	0.7855	**0.9614**	0.8333	**0.9583**	0.8087	**0.9599**
18	1130	0.9982	**1.0000**	0.9867	**1.0000**	0.9924	**1.0000**
19	1142	0.9722	**0.9922**	0.9510	**0.9991**	0.9615	**0.9956**
Avg		0.8489	**0.9592**	0.8220	**0.9182**	0.8223	**0.9349**
Weighted Avg		0.9304	**0.9821**	0.9260	**0.9828**	0.9269	**0.9818**

Table 4.2: predictive accuracy per class (locations)

precision, recall and F_1 scores. The results in details displayed in Table 4.2 clearly indicate that the proposed trajectory patterns-based model has outperformed the decision tree-based model at every classes (locations). The proposed method even obtained perfect prediction of the location ID 1, 2, 5 and 18, whereas the decision tree approach only achieved a moderate 75%-99% of F_1 score. The proposed method not only performed well on large classes, such as Loc IDs 15-23, with over a hundred visitations during the experiment, but also performed well on locations where the occupants rarely visited, e.g. Loc IDs 1-3. Whereas, the decision tree approach experienced more prediction errors at these locations, especially Loc IDs 2, 4 and 8.

The errors from misclassification from both methods can be seen more clearly in confusion matrices in Figure 4.10. The confusion matrix from the results of the decision tree-based predictor (Figure 4.10(a)) clearly shows more erroneous predictions compared to the results to the proposed method in Figure 4.10(b), which, from the results, seems more promising in delivering location-based services to the occupants than its competing predictive technique.



(a) tree-based model

(b) trajectory patterns-based transition time concerned model

Figure 4.10: Accuracy of (a) tree-based human mobility model and (b) trajectory patterns-based transition time concerned human mobility model visualized with confusion matrices

4.2.4 Conclusion

We have proposed a promising framework for human mobility predicting that works well even when presented with noisy data from simple motion detection sensors. Other than the essential preprocessing step of trajectory patterns mining that we have used in the past researches [15, 67] (see [66] for the analysis), in this work we presented a novel trajectory predictive technique that can model trajectory patterns from multiple occupants and predict future position of a moving subject with very high accuracy. The proposed transition time concerned trajectory pattern-based model has achieved over 98.21% of precision and the F_1 score of 98.18% prediction on a real world human mobility data. The proposed predictive model outperforms our previously proposed predictive model which has achieved only 95.92% of precision and 93.39% of F_1 scores on the same performance evaluation test. In addition, this result has closed the gap between the empirical performance and the theoretical limit of the predictability of that we have estimated in the previous research [66]. The proposed predictive technique has introduces a great opportunities to smart environment researchers to develop successful location-based applications with less complicated sensor system that leads to lower cost of production and development.

Chapter 5

Kernel Density Compression for online KDE

5.1 Kernel Density Estimation

KDE is a nonparametric method for approximating an unknown probability density function $p(\boldsymbol{x})$ from observations. The term kernel refers to any probability density function that is placed over each observation to quantify the likelihood of a small region around each observation containing $\boldsymbol{x}$. Thus, the estimated density is the weighted summation of contributions that each observation makes to the distribution.

5.1.1 Traditional Kernel Density Estimation

By definition, any kernel function $K(u) : \mathbb{R} \mapsto \mathbb{R}_{\geq 0}$ must be symmetric, i.e. $\forall_{u \in \mathbb{R}} : K(-u) = K(u)$, and satisfy the following condition:

$$\int_{-\infty}^{\infty} K(u)du = 1. \tag{5.1.1}$$

Given a set of observations $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$, $\forall \boldsymbol{X_i} \in \mathbb{R}^d$, the estimated probability density function $p(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^d$ is as follows:

$$p(\boldsymbol{x}) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{\boldsymbol{X}_i - \boldsymbol{x}}{h}\right), \tag{5.1.2}$$

where h is a parameter used to define the width of each kernel. This parameter is often referred to as *bandwidth*. For instance, a commonly used Gaussian kernel has the following density function:

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{h^d\sqrt{(2\pi)^d}}\exp\left(-\frac{1}{2}\frac{(\boldsymbol{X_i}-\boldsymbol{x})^T(\boldsymbol{X_i}-\boldsymbol{x})}{h^2}\right). \tag{5.1.3}$$

In general cases, the bandwidth parameter can be represented by a vector $\boldsymbol{h} = [h_1, h_2, \ldots, h_d]$ of d dimensions, in which each element individually represents the bandwidth parameter of each dimension. Thus, the density estimation function in Eq.(5.1.3) can be generalized as follows:

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\prod_{j=1}^{d}h_j\sqrt{(2\pi)^d}}\exp\left(-\frac{1}{2}\left(\frac{\boldsymbol{X_i}-\boldsymbol{x}}{\boldsymbol{h}}\right)^T\left(\frac{\boldsymbol{X_i}-\boldsymbol{x}}{\boldsymbol{h}}\right)\right). \tag{5.1.4}$$

5.1.2 Problem of KDE with online data

Kernel density estimation is a costly operation. The computation in Eq.(5.1.2) involves N iterations over all observations. Thus, a simple looping density estimation of M query points has a time complexity of $\mathcal{O}(MN)$ or $\mathcal{O}(N^2)$, where $M \approx N$. Although state-of-the-art dual-tree based KDE [24], which is known to be the fastest and most accurate algorithm, can scale the computation to $\mathcal{O}(N)$, it requires $\mathcal{O}(N \log N)$ time for construction (building tree structures). However, the dual-tree approach was not designed as an incremental algorithm. Namely, it is inefficient in an online setting, where the tree-based data structure requires reconstruction every time a new observation becomes available. So are the recently proposed distributed and parallel KDE [81] that was designed specially for very large datasets. The distributed KDE employed sub-sampling techniques to reduce size of KDE model and speedup the density estimation. However, the algorithm proposed in [81] was not designed to be incremental, namely a new data point cannot be directly inserted to the built model without rebuilding the entire model. Because each update takes $\mathcal{O}(N)$, the sub-sampling KDE was obviously more suitable for batch processing than online processing, where N could grow unboundedly.

Reducing the number of computations required to compute a point estimate of the density has been a main interest in optimizing KDE. A number of improvements have been considered to reduce the number of kernel components. For example, a straightforward binning method has been introduced in [61, 30], where the density is estimated from a sum of densities estimated from each bin weighted by the number of samples placed in each bin. In [64], a method that estimates the density with the convolution theorem using Fourier transformation, in which binning of data is also required, has been proposed. However, this requires the bin size for each dimension to be specified;

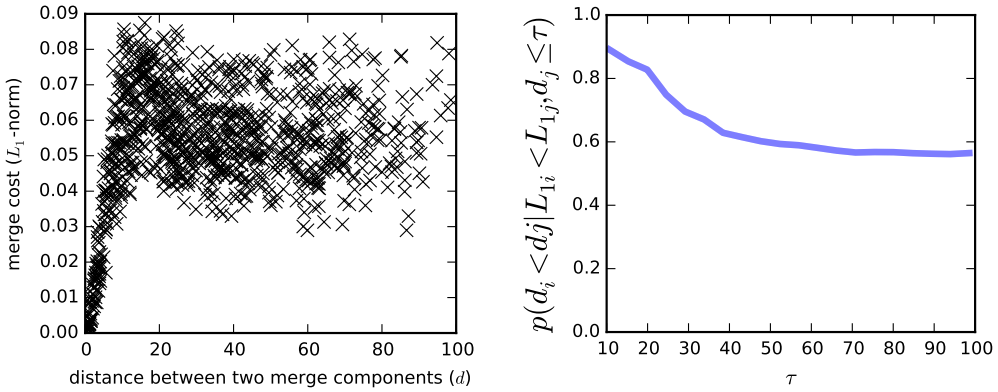
therefore, such approaches quickly become inefficient for high-dimensional multivariate data streams.

As an alternative to binning, in [1, 32, 38, 37, 79, 77], samples were clustered and reduced by replacing them with cluster centroids. The main problem with the cluster-based approaches is the computational burden of the optimization required for data partitioning and the solution's dependency on the initial condition. In addition, cluster-based methods, including a self organizing map-based KDE [12], often update their models with a small batch of buffered data (often a few hundreds). Hence, the models are not truly up-to-dated to each new sample, because they have to wait until the buffers get filled so that the models can call clustering procedure to update their density estimators with the new samples.

Some methods have been adapted to handle streaming data by reducing the data condensation overhead so that the model can keep pace with the input stream. The M-kernel merging technique [82] is an online density estimator that can handle a univariate data stream by limiting the number of kernel components and substituting redundant components with a representative component (kernel merging). The model invokes a merging routine whenever its buffer is full. Then, downhill simplex optimization is employed to find the best way to replace two components with a representative component that would minimize the merging cost, which is the absolute error (L_1-norm) between the estimation and the underlying density. Since its kernel merging strategy is based on numerical optimization, the time that the algorithm requires to update the model to a new sample can be high. As a solution, Heinz [28] has introduced an M-kernel based online KDE with a constant time pairwise merging technique to solve the problem of high update cost of the M-kernel algorithm. However, the M-kernel and other M-kernel-based approaches [28, 29, 6] cannot be generalized to support multivariate data streams without losing speed because the algorithms rely on sortedness which is more complicated to achieved for multivariate data. The lack of total ordering in multidimensional data forces the algorithms to use nearest neighbors search, which would incur additional computational complexity [29].

5.1.3 Idea of the proposed method

In this work, we addressed the problem of finding an optimal merging pair from a different perspective. Rather than optimizing for the optimal merging components pair



(a) Distance and merging cost

(b) Probability of finding a pair of Gaussians i having smaller distance than the other j given that merging pair i would cost less than pair j and merging only occurs when the distance is less than the distance threshold τ

Figure 5.1: Relationship between distance and merging cost

that is expected to minimize the overall merging cost, we skip such optimization and search for the best merging pair by their distance to reduce the time required to process each new sample. In particular, a newly arriving component will be merged with the nearest kernel component only if the nearest kernel exists within a specified distance threshold τ. Otherwise, the new component will be inserted into the model without merging. The parameter τ is used to limit the scope for neighbor search and can be used to control the compression ratio. Additionally, validity of the M-kernel based approaches is limited to one-dimensional data only. Generalization of the M-kernel to higher dimensional data would impact the compression overhead. In other words, the time required for the query of the optimal merging pair would be quadratic in stead of linear to the buffer size. On the contrary, the time required by our approach to search for the nearest component is always linear.

A simulation wherein 1,000 pairs of 2D multivariate Gaussian components in which different weights, widths and centres were randomly assigned shows a result that supports our approach. The result shown in Figure 5.1(a) indicates a clear relationship between the resulting merging cost and the distance between the two merging components; smaller distances, to some extent, tend to give smaller errors. Note that when the distances are greater than 10, this presumption is less likely to be true. Thus, we introduce a distance threshold parameter τ that limits the greatest distance allowed for merging to avoid the high probability of merging non-cost-optimized pairs at greater

distances. Furthermore, the necessity of τ is made clear in Figure 5.1(b). Given two arbitrary pairs of Gaussians i and j, the probability of finding a pair of Gaussians i having smaller distance than j's distance, given that merging pair i costs less than pair j and the merging only happens when the distance is less than the threshold τ, is a function of the distance threshold τ. The plot shows that the probabilities are high when the limits of the distance between two components allowed for merging are small. Namely, employing a smaller distance threshold τ yields a higher chance of finding optimal pairs on the basis of the shortest distance. However, note that it is difficult to find nearby neighbour kernels to merge with when the search radius (distance threshold τ) is small, which results in a less compressed model. Thus, the trade-off between the additional error caused by merging non-optimal pairs and the compactness of the resulting model can be managed by the τ parameter.

5.2 Proposed online kernel density estimation

5.2.1 Kernel compression algorithm

In an online setting, where observations are made sequentially and data samples arrive in order, the compression algorithm is required to process and compress new samples faster than the rate of the input stream so that the model is always updated and ready for the density estimation at any time. The process for handling new observation (training) stream is illustrated by a flowchart in Figure 5.2. Start off with an empty model G, as a new data sample $\boldsymbol{x_i}$ arrives, a Gaussian kernel is assigned with $\mathbf{x}_i$ as the centre with a predefined diagonal covariance matrix $\boldsymbol{\Sigma}$. Note that $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{h}^2)$, where $\boldsymbol{h^2} = [h_1^2, h_2^2, \ldots, h_d^2]$. Each new component is weighted uniformly with a weight coefficient w of 1. The algorithm then looks for an opportunity to merge this new component with an existing component in the mixture model in order to maintain the number of components. The algorithm performs a search for the most proper component to merge the new sample with in order to minimize compression error. Search for the proper component is done by a nearest neighbour search that considers the Mahalanobis distance between two components. In other words, it attempts to find a Gaussian component $\hat{g}$ in a set of Gaussian mixture models G that minimizes the Mahalanobis distance $D_{\hat{g}}(\boldsymbol{x_i})$ as follows:

$$\hat{g} = \underset{g_j \in G}{\text{argmin}}\, D_{g_j}(\boldsymbol{x_i}), \tag{5.2.1}$$

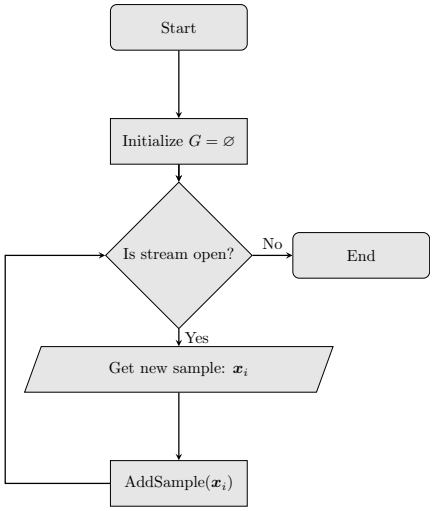


Figure 5.2: Training data stream handler

```
1: function AddSample(x_i)
2:     g' ← N(1, x_i, Σ)
3:     if G is empty then
4:         G.insert(g')
5:     else
6:         ĝ ← argmin_{g_j ∈ G} D_{g_j}(x_i)
7:         if D_ĝ(x_i) ≤ distance_threshold
   then
8:             G.remove(ĝ)
9:             g'' ←Merge(g', ĝ)
10:            G.insert(g'')
11:        else
12:            G.insert(g')
13:        end if
14:    end if
15: end function
```

Algorithm 5.1: Adding new sample

where

$$g_j = \mathcal{N}(w_j, \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) \tag{5.2.2}$$

and

$$D_{g_j}(\boldsymbol{x_i}) = \sqrt{(\boldsymbol{x_i} - \boldsymbol{\mu_j})^T \boldsymbol{\Sigma_j}^{-1} (\boldsymbol{x_i} - \boldsymbol{\mu_j})}. \tag{5.2.3}$$

The Mahalanobis distance $D_{\hat{g}}$ is then used to determine whether the newly arrived data sample should be merged to the nearest component. Only the nearest component with a specified *distance threshold* is merged. Otherwise, a new Gaussian kernel is assigned where the new data sample is with a predefined covariance matrix $\boldsymbol{\Sigma}$, thereby resulting in a new component. The outline of the algorithm for adding a new sample is given in Algorithm 5.1. We further discuss how the kernel merging routine (line 9) is handled in the next subsection.

5.2.2 Gaussian kernel merging

Goal of kernel merging is to substitute a mixture of two Gaussian components with a single Gaussian component that approximates the mixture best. The M-kernel merging [82], an alternative solution for Gaussian kernel merging that we have introduced in Section 5.1, approximated mean and bandwidth of the merged component using downhill simplex optimization method on the function of accuracy lost in the kernel merging. However, the optimization becomes less efficient when applying the M-kernel technique to higher dimensional data. Therefore, rather than optimizing for the mean vector and

the covariance matrix that would minimize the accuracy lost from kernel merging, we opt for the moment-preserving merge, which the approximation can be done in almost constant time ($\mathcal{O}(1)$) given that the number of dimensions are small.

Full covariance match

A mixture of two Gaussian components $\mathcal{N}(w_1, \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(w_2, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ can be approximated by a single representative Gaussian component by matching the zeroth, first, and second moments to the mixture. Namely, weight w, mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ of the resulting component $\mathcal{N}(w, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be approximated by the moment matching method as follow.

$$w = w_1 + w_2. \tag{5.2.4}$$

$$\boldsymbol{\mu} = w^{-1}(w_1\boldsymbol{\mu}_1 + w_2\boldsymbol{\mu}_2). \tag{5.2.5}$$

$$\begin{aligned}\boldsymbol{\Sigma} &= w^{-1}\left(w_1(\boldsymbol{\Sigma}_1 + (\boldsymbol{\mu}_1 - \boldsymbol{\mu})(\boldsymbol{\mu}_1 - \boldsymbol{\mu})^T) + w_2(\boldsymbol{\Sigma}_2 + (\boldsymbol{\mu}_2 - \boldsymbol{\mu})(\boldsymbol{\mu}_2 - \boldsymbol{\mu})^T)\right) \\ &= w^{-1}\left(w_1\boldsymbol{\Sigma}_1 + w_2\boldsymbol{\Sigma}_2 + w_1\boldsymbol{\mu}_1\boldsymbol{\mu}_1^T + w_2\boldsymbol{\mu}_2\boldsymbol{\mu}_2^T\right) - \boldsymbol{\mu}\boldsymbol{\mu}^T \\ &= w^{-1}\sum_{i=1}^{2} w_i(\boldsymbol{\Sigma}_i + \boldsymbol{\mu}_i\boldsymbol{\mu}_i^T) - \boldsymbol{\mu}\boldsymbol{\mu}^T. \end{aligned} \tag{5.2.6}$$

Although the moment-preserving merge described above was proven to give a merged component that has a minimal Kullback-Leiber discrimination from the mixture (see Theorem 3.2 in [54]), the resulting covariance matrix is not always diagonal in which many computational optimizations can take advantage of its mathematical properties (see Section 5.2.3 for further details). Therefore, we propose another kernel merging method for Gaussian components called the *Bandwidth match*, wherein the merge happens on the bandwidth vector of a Gaussian kernel and the merged covariance matrix is reconstructed from the merged bandwidth. To put it differently, the merge only considers diagonal elements of the covariance matrices. Hence, the resulting covariance matrix is guaranteed to be a diagonal matrix.

Bandwidth match

Suppose we are given a d-dimensional Gaussian kernel with the bandwidth $\boldsymbol{h} = [h_1, h_2, \ldots, h_d]$. By definition, the corresponding $d \times d$ covariance matrix $\boldsymbol{\Sigma}$ of the Gaussian kernel can

be constructed as follows.

$$\boldsymbol{\Sigma} \equiv \begin{bmatrix} h_1^2 & 0 & \dots & 0 \\ 0 & h_2^2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h_d^2 \end{bmatrix}. \tag{5.2.7}$$

In other words,

$$\begin{aligned} \boldsymbol{\Sigma} &\equiv \text{diag}([h_1^2, h_2^2, \dots, h_d^2]) \\ &= \text{diag}(\boldsymbol{h}^2). \end{aligned} \tag{5.2.8}$$

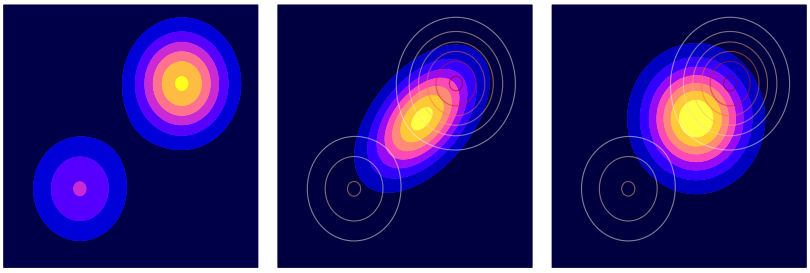
Referring to the moment-preserving merge, we modified Eq.(5.2.6) to only match diagonal elements of the covariance matrix, ignoring all other elements as follow.

$$\begin{aligned} \boldsymbol{h}^2 &= w^{-1}\left(w_1(\boldsymbol{h}_1^2 + (\boldsymbol{\mu}_1 - \boldsymbol{\mu})^2) + w_2(\boldsymbol{h}_2^2 + (\boldsymbol{\mu}_2 - \boldsymbol{\mu})^2)\right) \\ &= w^{-1}\left(w_1\boldsymbol{h}_1^2 + w_2\boldsymbol{h}_2^2 + w_1\boldsymbol{\mu}_1^2 + w_2\boldsymbol{\mu}_2^2\right) - \boldsymbol{\mu}^2 \\ &= w^{-1}\sum_{i=1}^{2} w_i(\boldsymbol{h}_i^2 + \boldsymbol{\mu}_i^2) - \boldsymbol{\mu}^2. \end{aligned} \tag{5.2.9}$$

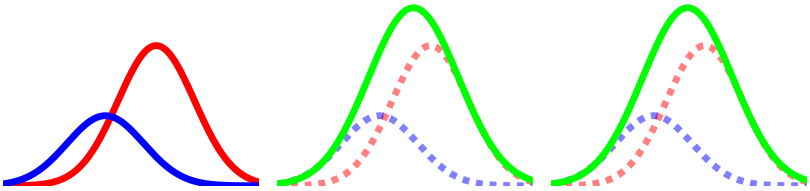
If needed, the resulting bandwidth $\boldsymbol{h}^2$ can also always be used to reconstruct a diagonal covariance matrix $\boldsymbol{\Sigma}$ using Eq.(5.2.8).

It is worth noting that, in contrast to the full covariance match, the proposed bandwidth match method does not require matrix multiplication, which takes $\Theta(d^2)$ to compute for each merge. Instead, $\boldsymbol{h}^2$ only requires $\Theta(d)$ time complexity to compute. Hence , kernel merging of high-dimensional data could perform significantly faster with the bandwidth match method. Kernel merging can be done even faster without the second-moment matching. In the next merging method, we proposed an experimental idea of keeping the bandwidth parameter constant through out the entire stream.

Differences among the two merging methods for one-dimensional (1D) and two-dimensional (2D) kernels are visualized in Figure 5.3.



(a) before (b) full covariance (c) bandwidth



(d) before (e) full covariance (f) bandwidth

Figure 5.3: Kernel merging methods

5.2.3 Efficient Density Evaluation

Suppose we are given a mixture of Gaussian components, probability density function of the mixture containing C components can be expressed as follows:

$$p(\boldsymbol{x}) = \sum_{i=1}^{C} w_i \phi_i(\boldsymbol{x}), \tag{5.2.10}$$

where

$$\phi_i(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}_i|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right). \tag{5.2.11}$$

An interesting characteristic of the bandwidth match merging method is the covariance matrices of all components remain diagonal after compression. Therefore, we can rewrite Eq.(5.2.11) to

$$\phi_i(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d \prod_{j=1}^{d} h_{i,j}^2}} \exp\left(-\frac{1}{2}\sum_{j=1}^{d} \frac{(x_j - \mu_{i,j})^2}{h_{i,j}^2}\right), \tag{5.2.12}$$

where we can set a *cut-off threshold* to skip the density evaluation of certain components $\boldsymbol{\mu}_i$ when they are too distant, which assumes that the evaluating point $\boldsymbol{x}$ is less likely to be drawn from such distributions. Thus, the very low density contributions from distant components are omitted to obtain a computational speedup. The cut-off optimization

is implemented while looping the sum through d dimensions in the exponent part of Eq.(5.2.12). That is, if the normalized sum of the squared distance at any point reaches the threshold, then the evaluation of $\phi_i(x)$ is stopped, the result of $\phi_i(x)$ is dismissed and the procedure moves on to the next component $\phi_{i+1}(x)$. Note that the cut-off can occur earlier if the dimensions are sorted such that dimensions with higher variance are considered first. This optimization significantly reduces evaluation time, particularly for high-dimensional data.

5.3 Simulations and applications to experimental data

To gain a better understanding of how neural ensembles communicate and process information, neural decoding algorithms are used to extract information encoded in their spiking activity. Bayesian decoding is one of the most used neural population decoding approaches to extract information from the ensemble spiking activity of rat hippocampal neurons. Recently it has been shown how Bayesian decoding can be implemented without the intermediate step of sorting spike waveforms into groups of single units. Here we extend the approach in order to make it suitable for online encoding/decoding scenarios that require real-time decoding such as brain-machine interfaces. We propose an online algorithm for fast Bayesian decoding that reduces the time required for decoding neural populations, resulting in a real-time capable decoding framework. More specifically, we improve the speed of the probability density estimation step, which is the most essential and the most expensive computation of the spike-sorting-less decoding process, by developing a kernel compression algorithm. In contrary to existing online kernel compression techniques, rather than optimizing for the minimum estimation error caused by kernels compression, the proposed method compresses kernels on the basis of the distance between the merging component and its most similar neighbor. Thus, without costly optimization, the proposed method has very low compression latency with a small and manageable estimation error. In addition, the proposed bandwidth matching method for Gaussian kernels merging has an interesting mathematical property whereby optimization in the estimation of the probability density function can be performed efficiently, resulting in a faster decoding speed. We successfully applied the proposed kernel compression algorithm to the Bayesian decoding framework to reconstruct positions of a freely moving rat from hippocampal unsorted spikes, with significant improvements in the decoding speed and acceptable decoding error.

Neural encoders and decoders are commonly used by neuroscientists to study the relation between behavioural stimulus and neural responses. Statistical inferences have played an important role in many encoding/decoding frameworks, e.g. [10, 80, 49, 71, 73, 52]. Generally, the encoding model captures necessary properties from the recorded neural activities and construct a model that maps to the observed behaviours or stimuli. The decoding model then employs the constructed relation to infer behaviours or stimuli based on the observed neural activity. For example, neural signals recorded from the action potentials (spikes) of pyramidal neurons in the CA1 region of the hippocampus contain information that is correlated to spatial behaviours of an animal [48]. These cells are also known as place cells because spiking activities of certain place cells become more active when an animal is in a certain location [11]. In other words, the temporal patterns of spikes from different place cells are spatially tuned to different locations.

Most of existing neural encoders/decoders require sorted spikes to operate [4, 9, 10, 21, 20, 44, 57, 58, 62, 72, 76, 80] (see [8] for a review). That is spiking activity of each single neuron has to be isolated from others and separated from background electrical noise before being handed over to the encoding/decoding model. This prerequisite step is called "spike sorting". Many works have been contributed to the developing of fast and reliable spike sorting algorithms [41]. However, a study has shown that classification errors of assigning spikes to incorrect unit have various impact to information capacity of the resulting sorted spikes [23]. In addition, objective of spike sorting to isolate and identify the cell that originated each spike is rather different from the goal of neural decoding which is to minimize the decoding error. Unclassified spikes during the sorting in attempt to minimize sorting errors could still convey information that is necessary to the encoder/decoder.

To avoid the possibility of information loss and accumulation of errors from spike sorting, Bayesian encoding/decoding framework proposed in [35] has introduced a method to create a direct mapping between spike waveform features and the covariates of interest without a prerequisite step of spike sorting. The name "Bayesian" comes from the adoption of a statistical inference that utilizes Bayes' theorem. More specifically, the decoding is obtained by the maximum posterior probability in Eq.(5.3.1), where the covariates are spatial behaviors of the animal, e.g. positions or head directions.

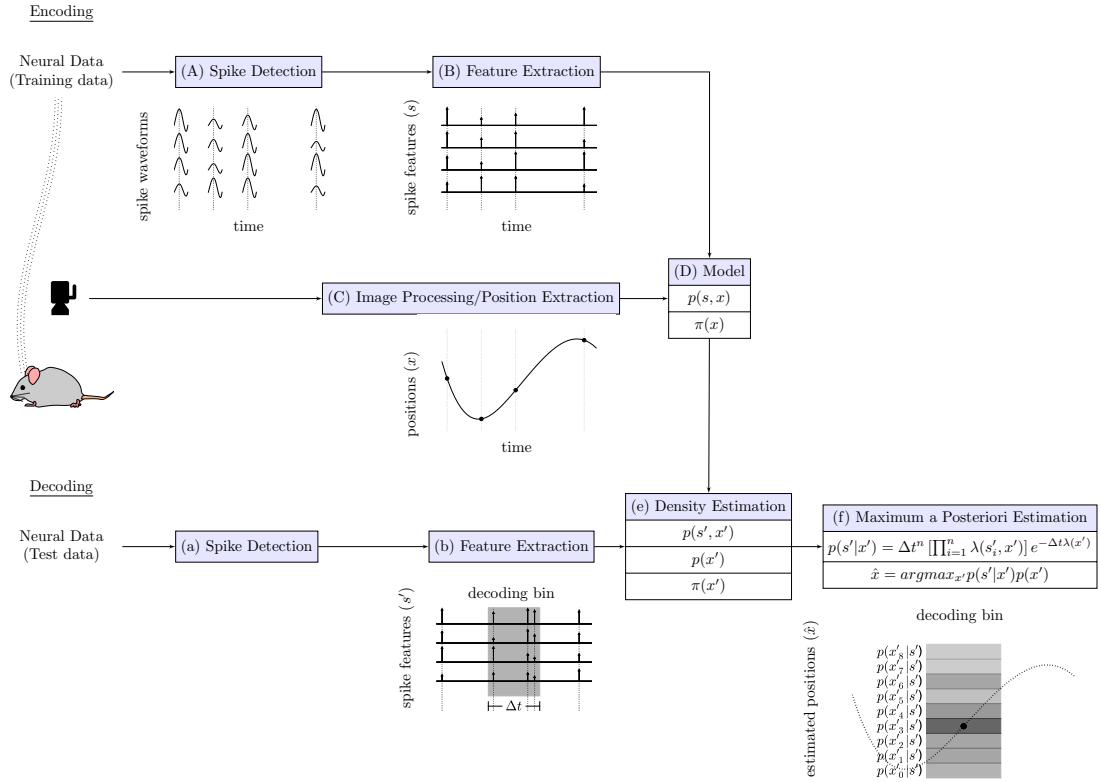


Figure 5.4: Bayesian decoding using unsorted spikes in the rat hippocampus

$$p(covariates|spikes) = p(x|s) = \frac{p(s|x)p(x)}{p(s)}$$
$$\propto p(s|x)p(x) \tag{5.3.1}$$

Outline of the Bayesian encoding/decoding framework [35] is illustrated in Figure 5.4. The first stage (A) detects and extracts spike waveforms from extracellularly recorded multiunit activity from CA1 region of a free moving rat. Next (B), waveform features, such as amplitudes, are extracted. At the same time (C), positions of the animal is tracked using a video camera and forwarded together with the waveform features to the next stage (D), where the probability models $p(s, x)$ and $\pi(x)$ are modeled.

During the decoding phase (e), a sequence of spikes is partitioned into bins. For each decoding bin, the posterior probability is computed and the behaviour is decoded. The likelihood $p(s|x)$ of the stimulus x given a set of spike features s models the relation between spiking patterns (modulation of spike amplitudes and firing rates) and

behaviours by assuming spatiotemporal Poisson statistics as follows:

$$p(s|x) = \Delta t^n \left[\prod_{i=1}^{n} \lambda(s_i, x) \right] e^{-\Delta t \lambda(x)}, \tag{5.3.2}$$

where the decoding bin containing n spikes has a size of Δt time interval. Rate parameter $\lambda(s_i, x)$ which is the fraction of the occurrences of certain spike features s_i coinciding with certain stimulus x divided by the total time stimulus x ($occupancy(x)$) is presented as follows:

$$\lambda(s_i, x) = \frac{spikecount(s_i, x)}{occupancy(x)} = \frac{N}{T} \frac{p(s_i, x)}{\pi(x)} = \mu \frac{p(s_i, x)}{\pi(x)}, \tag{5.3.3}$$

where N is the total number of spikes and T is the total time from all the decoding bins. $p(s_i, x)$ is the joint probability distribution of finding spike features s_i that occur simultaneously when the animal experiences stimulus x. The probability distribution $\pi(x)$ is the probability of finding the animal experiencing stimulus x. Similarly,

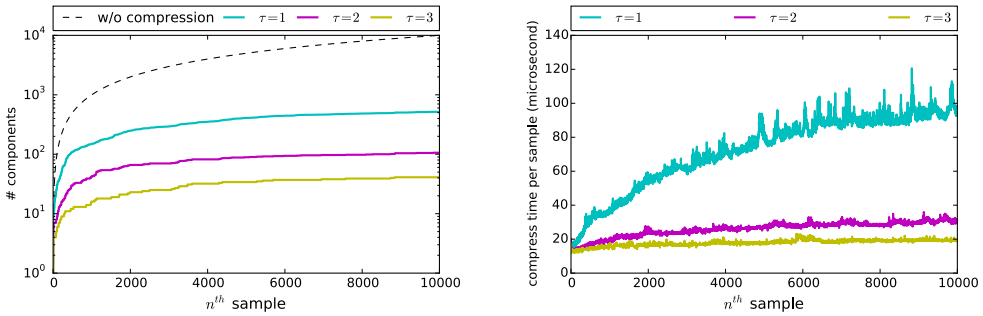
$$\lambda(x) = \frac{spikecount(x)}{occupancy(x)} = \frac{N}{T} \frac{p(x)}{\pi(x)} = \mu \frac{p(x)}{\pi(x)}, \tag{5.3.4}$$

with the exception that $spikecount(x)$ counts all the spikes that occur during the experience of stimulus x regardless of the spike features.

The most time consuming step is the computation of the probability densities $p(s_i, x)$, $p(x)$ and $\pi(x)$, which can be estimated using kernel density estimation (KDE). However, time complexity of KDE is approximately quadratic, which is extremely expensive, often requires excessive time and impractical for applications such as our goal of developing real-time Bayesian decoding framework. Moreover, tradition KDE does not scale well to the unbounded streams of data, which is the nature of all real-time applications.

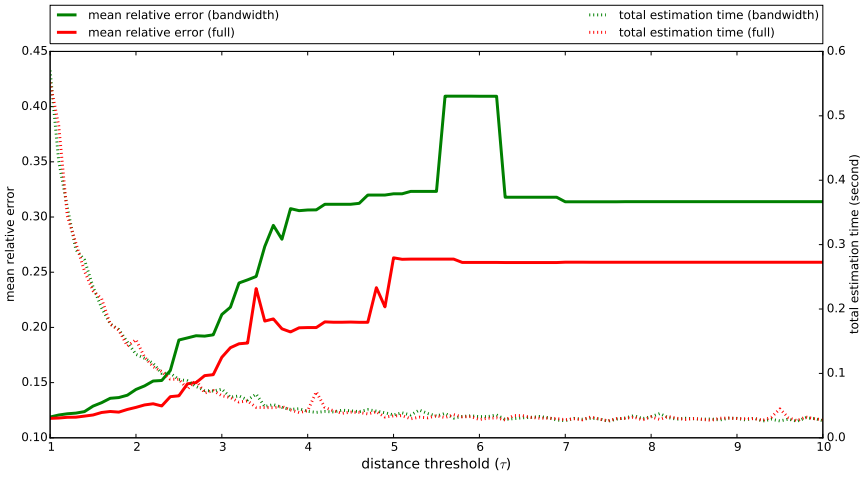
Thus we implemented the proposed kernel compression algorithm into the encoder/decoder to speedup the density estimation task in order to achieve real-time decoding. The proposed method achieves faster density estimation by replacing redundant kernel components with mixtures of merged components, resulting in a reduced number of kernel components; thus, the density evaluation time is reduced.

We have separated the experiments into three parts to cover (1) the trade-off between speed and accuracy, (2) performance evaluation on high-dimensional data streams and (3) performance evaluation on the real-time decoding of the rat hippocampus.



(a) number of components after adding the n^{th} sample

(b) time required to compress each new sample



(c) time required to compress each new sample

Figure 5.5: Performance of the kernel compression algorithm on the data stream of spike features from rat hippocampus

5.3.1 Trade-off between speed and accuracy

We tested the proposed kernel compression algorithm using data stream of neural and behavioral recoding of a free moving rat in the experiment environment described in the introduction. Each sample of the spike features stream contains 4 values (from 4 separated electrode in a tetrode), and the data stream of the animal positions contains 2 values (xy-coordinates). To give an overview image of how the distance threshold τ has an effect to the compression, we streamed the first 10,000 samples from the 4-dimensional data stream of spike features to the proposed compression algorithm. As displayed in Figure 5.5(a), the larger the distance threshold τ, the more the number of KDE components were reduced. As a result, the time required to handle each new sample was also reduced greatly, as shown in Figure 5.5(b). After the compression, we compared the estimation accuracy and the estimation speed of the compressed model

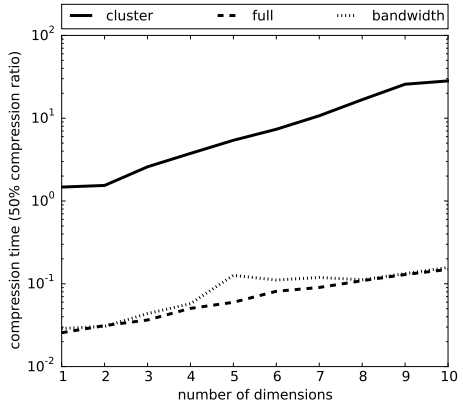
with tradition KDE model which was built from the same set of dataset. The densities estimated from the traditional KDE (without compression) were used as a ground truth. The accuracy of the compressed model was evaluated by the average of relative errors which is defined as follows:

$$\text{relative error}_i = \frac{d_i^{compressed} - d_i^{KDE}}{d_i^{KDE}} = \frac{d_i^{compressed}}{d_i^{KDE}} - 1, \tag{5.3.5}$$

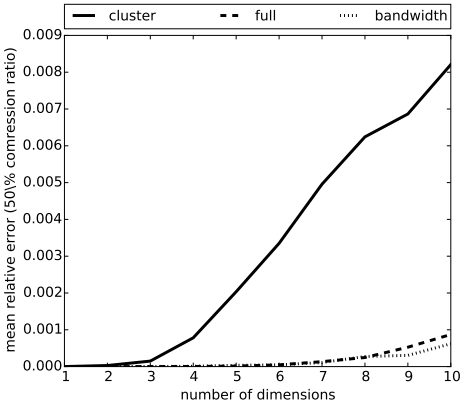
where $d_i^{compressed}$ is the density at the evaluation point i estimated from the compressed KDE and d_i^{KDE} is the true density at i estimated from the uncompressed KDE. The averaged errors were computed from the density estimations of other 10,000 data points drawn from the spike features stream. Accuracy and speed of the density estimation of the proposed method with two different kernel merging methods are shown in Figure 5.5(c). From the result, as we increased the distance threshold parameter τ, the times required to estimate all 10,000 evaluation points were reduced quickly. At the same time, the accuracy loss from kernels merging started to increase as we raised τ higher until the errors were almost stable when τ were high enough to merge every new sample, resulting in a compressed KDE with only one component. In addition, from the result we can observe that it was a very good trade-off between speed and accuracy. From the experiment, traditional KDE took 2.5 seconds to finish density estimations of 10,000 evaluation points. At $\tau = 2$, the proposed methods only took 0.15 second to finish the same task. That is we speeded up the density estimation by almost 17 folds, whereas the average estimation errors were raised only by 15% from the full covariance match and bandwidth match methods.

5.3.2 Performance evaluation on high-dimensional data streams

To emphasize on the advantages of the proposed bandwidth match method over other kernel compression techniques on high dimensional data streams, we compared compression speeds of the proposed methods and the cluster-based *Online Discriminative Kernel Density Estimator with Gaussian kernels* [37] on a synthesized dataset of high dimensional data streams of 1,000 uniformly random numbers. In this simulation, all compression algorithms were set to compress 1,000 original samples to about 500 samples (50% compression ratio). For the cluster-based online KDE, an existing cluster is updated every time 100 new sample is added (buffer size = 100). Visualization in Figure 5.6(a) and Figure 5.6(b) clearly shows that our kernel merging approaches out-



(a) the average time required to compress each high-dimensional sample



(b) the average time required to compress each high-dimensional sample, focusing on the cluster-based approach [38] and the proposed kernel compression with full covariance match

Figure 5.6: Performance comparison on high dimensional data streams

performed the cluster-based online KDE in both speed and accuracy tests. From the result, it is apparent that the number of dimensions has significant impact on the performance of the cluster-based approach, while our choices of kernel merging techniques suffered less from high dimensionality, especially for the proposed bandwidth matching method.

5.3.3 Performance evaluation on the real-time decoding of the rat hippocampus

In this final experiment, we integrated the proposed kernel compression algorithm into the Bayesian decoding framework [35]. The dataset that will be used in this experiment was prerecorded from the experiment similar to the setup illustrated in Figure 5.4. Spiking activity from the hippocampus was recorded using 9 tetrodes[1]. Positions of the animal in xy-coordinates were tracked using a video camera. To simulate real-time encoding/decoding environment, we set a non-overlapping sliding window to read in small batches from data streams of both spike features and the animal's positions at a time. Size of the sliding window was set to 250 milliseconds, which is the recommended bin size of for decoding unsorted spikes of rat hippocampus [35]. Real-time decoding framework implemented in this experiment was designed to alternate between the decoding and encoding steps. Referring to a block diagram presented in Figure 5.7, when the

[1]Tetrode is a bundle of 4 electrodes

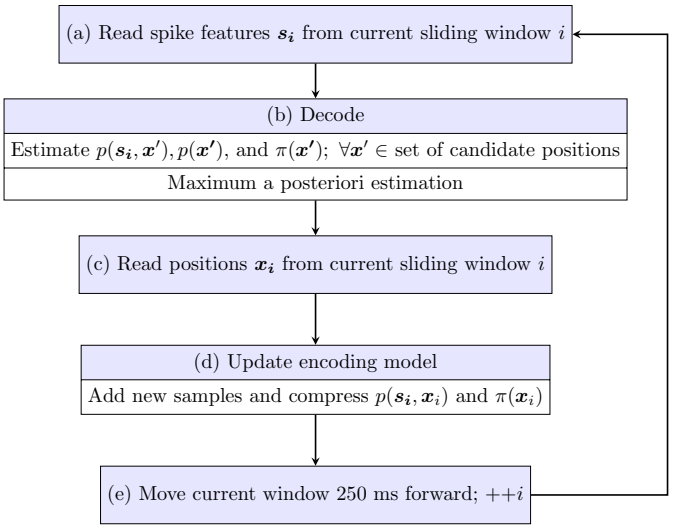
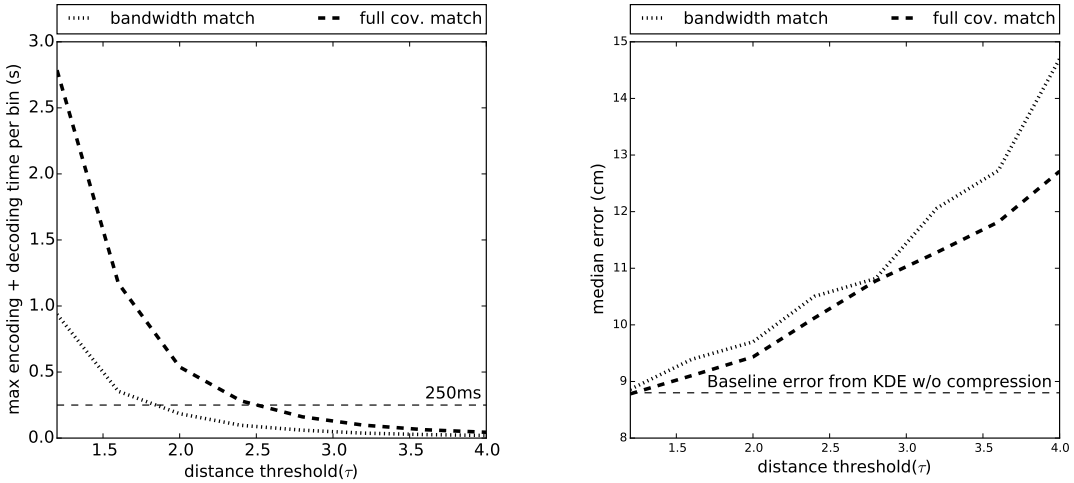


Figure 5.7: Block diagram of the experimental real-time encoding/decoding framework

sliding window moves on to the next batch of the streams (a), the decoding step (b) is invoked first to decode information from the newly observed neural signals then the encoding step (d) joins to update the encoding model with the newly observed data. The the sliding window is proceeded (e) to process next batch of data from the streams.

Accuracy of the decoding is evaluated by the euclidean distance between the observed position and the position estimated from neural data. To find the right amount of compression that would speed up the encoding/decoding to real-time and would not incur much accuracy loss, we varied the merging distance threshold (τ) to find the right parameters for each kernel merging methods. Because the number of spikes in the sliding window may vary from time to time, the amount of time required to process each window can also vary. We measured the max amount of time required to decode and encode each window and visualized the longest time needed by each algorithm as a function of τ. The results are shown in Figure 5.8(a). For the encoding/decoding framework to be able to process data streams in real-time, the time required to process each batch of neural and behavioral data in a sliding window has to be shorter than the streaming rate, which is 250 millisecond per batch. According to the results displayed in Figure 5.8(a), the encoding/decoding model that implements full covariance matrix match would need to set the distance threshold above 2.8, whereas the proposed bandwidth match can afford to have lower distance threshold τ, which resulted in lower decoding errors as shown in Figure 5.8(b). Despite the fact that the bandwidth matching method only matches diagonal elements of the covariance matrix and discard the rest, we can expected the



(a) Relation between the amount of time required to decode and encode each window and the merging dis-merging distance threshold (τ)

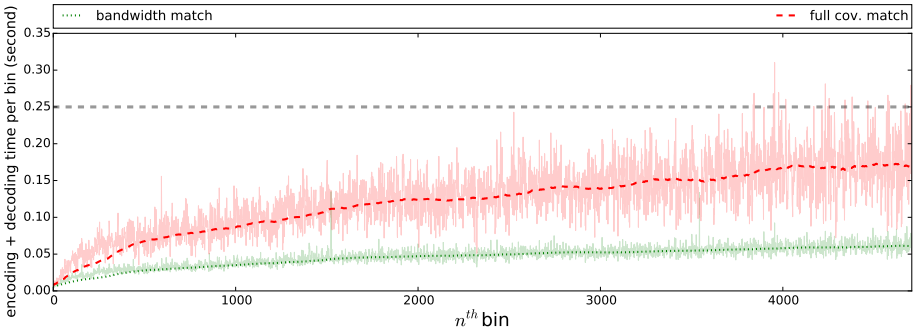
(b) Relation between the decoding error (cm) and the merging distance threshold (τ)

Figure 5.8: Performance comparison between the full covariance match and the proposed bandwidth match kernel merging approaches

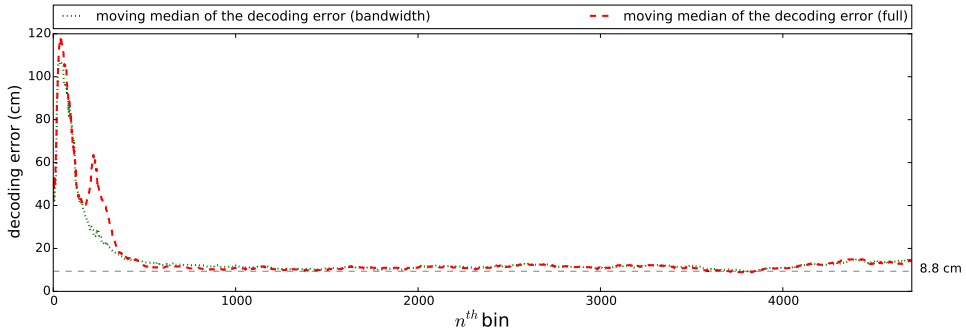
bandwidth matching method to produce more error. Interestingly, it was as accurate as its competition until $\tau = 2.0$, yet always faster.

ext, we set the distance threshold $\tau = 3.0$ for both of the kernel merging methods. Results in Figure 5.9(a) show progressions of the time required to process each window as the encoding models processed more data points over time. It can be seen that the performance in terms of speed was steady. Namely, the model can scale to large data stream efficiently. Decoding accuracy from both kernel compression methods visualized by the moving median of the decoding errors (cm) are shown in Figure 5.9(b). From the result, it can be seen that the decoding errors decreased quickly and became stable when both decoders had sufficient training data for the decoding at about the 500^{th} bin. The decoding model that was equipped with the full covariance matrix matching methods obtained the median decoding error of 11.03 cm, compared to a slightly larger median error of 11.21 cm from the bandwidth matching method. However, the bandwidth matching method was much faster and truly capable of real-time decoding. In contrast to the full covariance match, the decoding model could not process all bins within the time limit of 250 milliseconds per encoding/decoding bin.

In summary, we were able to speedup the Bayesian encoding/decoding framework by at least 300 times, which is fast enough to run the encoder/decoder on real-time



(a) Progression of the encoding/decoding speed over time



(b) Decoding accuracy of the decoder that implements full covariance matching method in details

Figure 5.9: Performance comparison between the full covariance match and the proposed bandwidth match kernel merging approaches on the decoding of rat positions from unsorted spikes

data. The decoding accuracy loss from kernel compressions were relatively small and still manageable.

Notes on the developing and testing environment

All the core algorithms that were used for the experiments in this paper were implemented in C programming language. The codes for all experiments and performance evaluations were written in Python. The computing server used in all experiments runs on Intel Xeon 12-Core 2.7 GHz $\times$ 2 CPUs.

5.4 Conclusions

In this work, we have improved the decoding speed of the Bayesian framework for encoding and decoding of unsorted spikes from the rat hippocampus. Real challenges of this work were to design a kernel compression algorithm that could enable KDE to handle high-dimensional data streamed at high speed efficiently. Thus we proposed a fast kernel compression technique that not only can reduce size of a density estimator

with very low accuracy loss, but also works well with high-dimensional data. More specifically, the proposed bandwidth match for kernel compression has been shown by the experiments presented in this paper to be very efficient especially when compressing high-dimensional data. Results from the real-time neural decoding experiment also confirmed the potentials of the bandwidth matching method.

Chapter 6

Conclusions

6.1 Summary of Contributions

In this research, we advanced the techniques in human mobility modeling and prediction that are practical to implemented in real-world applications without privacy issues and provide highly accurate result.

Starting with a careful choice of human mobility sensing and tracking method that less obtrudes the users compared to other methods, such human tracking using cameras and GPS tracking devices. In this research, we used ambient simple sensors, such as IR proximity sensors, light sensors, and magnetic sensors, that can be installed easily and can blend-in very well to the environment. Although, it can be expected that the data that these simple sensors collect are very noisy and seem inappropriate to use them to train a machine learning model directly without any careful preprocessing.

In Chapter 3, not only we discussed how human mobility can be model efficiently for different prediction tasks (short-term and long-term prediction), but we also analyzed to find theoretical limits of the predictability of the collected human mobility data and the factors that affected most to the predictability of the data. The results have confirmed that even though the simple sensors' human mobility data were very noisy, sufficient information can be extracted and can be used to train a classification model to give acceptably high accuracy for both short-term and long-term predictions

Next, we proposed the Aperiodic and Periodic models for long-term human mobility prediction that can accurately predict visitations of the uses at any locations of interest for days ahead of time with acceptably high accuracy compared to an exiting model that is based on nonlinear component analysis [60]. The proposed predictor aims at modeling repetitive patterns of users' visitations using both periodic model (constant

time interval) and aperiodic model (days with similar visitations patten are discovered using a clustering technique). With the proposed method, we could achieved stable F_1 score of 55% from predicting future visitations at 30 locations of interest for 30 days ahead.

In addition, we also proposed a new design for short-term human mobility predictor that can improve the prediction accuracy from about 96% of F_1 score from a tree-based model to 98% of F_1 score with the proposed technique that implements nearest neighbors classification to incorporate transition times between each step into the model.

Finally, the study was focused on the improvement of the kernel density estimation technique, which is useful in many probabilistic machine learning model and statistical applications. One specific application that we are addressing in this work is to have a KDE technique that can handle multi-dimensional data stream efficiently and accurately enough to encode/decode neural signals from a moving rat. Therefore, we designed and proposed a fast kernel compression technique that can efficiently compress and remove redundancy in the samples and results in a more compact representation of the distribution than tradition KDE, which cannot be scaled to efficiently to streaming data. The proposed techniques successfully enable real-time encoding/decoding of rat hippocampal spikes with moderately (and controllable) accuracy trade-off.

In summary, we addressed a variety of technical problems that are related to mobility study, from human mobility tracking, modeling and prediction to a tool for neuroscientists to study how the brain performs navigational tasks. Not only location-based smart home applications and real-time brain machine interface will benefit from the analyses and predictive techniques proposed in this work, but also more general applications, such as human social study and link prediction [74] and real-time population density visualization, also can be built based on the techniques presented in this research.

6.2 Recommendation for Future Works

Although, we have shown that a collection of simple motion detection sensors can be used to track mobilities from multiple participants in an indoor environment effectively. The collected mobility data contain sufficient information to train a machine learning model to predict next steps of a traveling user (short-term) or future visitations at any location of interest at any point in time in the future (long-term). Another question yet to be studied further afterwards is the predictability analysis and predictive accuracy

evaluation that factor in size of the environment, number of sensors, placements and distribution of the sensors, and the number of participants.

Additionally, there are recent studies that have been trying to investigate the correlations between social relationships and human dynamics. The studies [13, 74] have discovered that there are correlations between individuals' movements and social interactions (e.g. phone calls, messages). Moreover, Cho et al. [13] have also succeeded in improving prediction performance by integrating social contexts with mobility model. Whereas Wang et al. [74] have utilized the similarity between two individuals' movements to foresee their future interactions.

From here, we have seen that more aspects of human dynamics have helped researchers constructed human mobility models that employ broader understanding of human mobility. Ensemble methods, such as stacked generalization [75], can be implemented combine predictions from different machine learning models that are trained to predict human mobility from different aspects. Stacked generalization technique has been successfully used to improve prediction accuracy in many classification tasks [63, 3, 33]. The intuition behind stacked generalization is straightforward. Base models that are trained from different aspects of human mobility can be seen as different feature extractors, thus stacking is basically having another layer of classification model learn from these features to improve their accuracy.

In addition to the proposed fast kernel compression for online KDE, even thought the proposed kernel compression algorithm has an advantage over the some existing online KDE algorithms, such as the M-kernel merging algorithm [82], and most of cluster-based approaches [1, 32, 38, 37, 79] when dealing with multi-dimensional data streams, the proposed methods has not taken into account the situation where data stream is not stationary. Evolving density is a common problem that can impact performance of our kernel compression algorithms in many ways. Firstly, kernel merging would be less likely to occur, size of the model could expand uncontrollably if the distribution keeps drifting away its over time with out repeating the same spot. Secondly, the fixed distance threshold parameter (τ) might get outdated when the underlying distribution expanded or condensed as the distribution evolved. The first problem can be handled easily using decaying weights similarly to [28, 29], in which old components would weighted less and can be removed after certain period of time. The more challenging problem is the second problem. Detecting changes [34, 40, 56] and adapting KDE model to changes in

streaming data are another challenging topics that could significantly improve accuracy of the density estimation of non-stationary data streams, which will be our next main focus in future works.

Bibliography

[1] Gregory A Babich and Octavia I Camps. Weighted parzen windows for pattern classification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(5):567–570, 1996.

[2] M.H. Baeg, Jae-Han Park, Jaehan Koh, Kyung-Wook Park, and Moon-Hong Baeg. Building a smart home environment for service robots based on RFID and sensor networks. In *International Conference on Control, Automation and Systems*, (ICCAS '07), pages 1078–1082, 2007.

[3] Xinlong Bao, Lawrence Bergman, and Rich Thompson. Stacking recommendation engines with additional meta-features. In *Proceedings of the third ACM conference on Recommender systems*, pages 109–116. ACM, 2009.

[4] Riccardo Barbieri, Loren M Frank, David P Nguyen, Michael C Quirk, Victor Solo, Matthew A Wilson, and Emery N Brown. Dynamic analyses of information encoding in neural ensembles. *Neural Computation*, 16(2):277–307, 2004.

[5] Csaba Beleznai, D. Schreiber, and M. Rauter. Pedestrian detection using GPU-accelerated multiple cue computation. In *Computer Vision and Pattern Recognition Workshops*, (CVPRW '11), pages 58–65, 2011.

[6] Arnold P Boedihardjo, Chang-Tien Lu, and Feng Chen. Fast adaptive kernel density estimator for data streams. *Knowledge and Information Systems*, 42(2):285–317, 2015.

[7] Philipp Bolliger, Kurt Partridge, Maurice Chu, and Marc Langheinrich. Improving location fingerprinting through motion detection and asynchronous interval labeling. In *Location and Context Awareness*, pages 37–51. Springer, 2009.

[8] AE Brockwell, Robert E Kass, and AB Schwartz. Statistical signal processing and the motor cortex. *Proceedings of the IEEE*, 95(5):881–898, 2007.

[9] Anthony E Brockwell, Alex L Rojas, and RE Kass. Recursive bayesian decoding of motor cortical signals by particle filtering. *Journal of Neurophysiology*, 91(4):1899–1907, 2004.

[10] Emery N Brown, Loren M Frank, Dengda Tang, Michael C Quirk, and Matthew A Wilson. A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *The Journal of Neuroscience*, 18(18):7411–7425, 1998.

[11] J Bures, AA Fenton, Yu Kaminsky, and L Zinyuk. Place cells and place navigation. *Proceedings of the National Academy of Sciences*, 94(1):343–350, 1997.

[12] Yuan Cao, Haibo He, and Hong Man. Somke: Kernel density estimation over data streams by sequences of self-organizing maps. *Neural Networks and Learning Systems, IEEE Transactions on*, 23(8):1254–1268, 2012.

[13] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[14] Grzegorz Cielniak, Maren Bennewitz, and Wolfram Burgard. Where is...? learning and utilizing motion patterns of persons with mobile robots. In *IJCAI*, pages 909–914, 2003.

[15] Sodkomkham Danaipat, Legaspi Roberto, Kurihara Satoshi, and Numao Masayuki. A study on activity predictive modeling for prompt and delayed services in smart space. In *Workshop on Computation: Theory and Practice (WCTP 2012) (PICT 7)*, pages 316–328, 2012.

[16] B. Das, C. Chen, N. Dasgupta, D.J. Cook, and A.M. Seelye. Automated prompting in a smart home environment. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 1045–1052. IEEE, 2010.

[17] Robert M Fano. Transmission of information: A statistical theory of communications. *American Journal of Physics*, 29:793–794, 1961.

[18] Brian Ferris, Dieter Fox, and Neil D Lawrence. Wifi-slam using gaussian process latent variable models. In *IJCAI*, volume 7, pages 2480–2485, 2007.

[19] Brian Ferris, Dirk Haehnel, and Dieter Fox. Gaussian processes for signal strength-based location estimation. In *In Proc. of Robotics Science and Systems*. Citeseer, 2006.

[20] Apostolos P Georgopoulos, Ronald E Kettner, and Andrew B Schwartz. Primate motor cortex and free arm movements to visual targets in three-dimensional space. ii. coding of the direction of movement by a neuronal population. *The Journal of Neuroscience*, 8(8):2928–2937, 1988.

[21] Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

[22] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, 2008.

[23] Ilan N Goodman and Don H Johnson. Information theoretic bounds on neural prosthesis effectiveness: The importance of spike sorting. In *Proc. of Acoustics, Speech and Signal Processing, IEEE International Conference on*, pages 5204–5207. IEEE, 2008.

[24] Alexander G Gray and Andrew W Moore. Nonparametric density estimation: Toward computational tractability. In *Proc. of SIAM International Conference on Data Mining*, pages 203–211. SIAM, 2003.

[25] Hamed Haddadi, Pan Hui, and Ian Brown. MobiAd: Private and Scalable Mobile Advertising. In *Proceedings of the Fifth ACM International Workshop on Mobility in the Evolving Internet Architecture*, (MobiArch '10), pages 33–38, 2010.

[26] Richard W Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

[27] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and MC Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, 2001.

[28] Christoph Heinz and Bernhard Seeger. Towards kernel density estimation over streaming data. In *Proc. of International Conference on Management of Data*, pages 80–91, 2006.

[29] Christoph Heinz and Bernhard Seeger. Cluster kernels: Resource-aware kernel density estimators over streaming data. *Knowledge and Data Engineering, IEEE Transactions on*, 20(7):880–893, 2008.

[30] Lasse Holmström. The accuracy and the computational complexity of a multivariate binned kernel density estimator. *Journal of Multivariate Analysis*, 72(2):264–309, 2000.

[31] Sajid Hussain, Scott Schaffner, and Dyllon Moseychuck. Applications of Wireless Sensor Networks and RFID in a Smart Home Environment. In *Proceedings of the 2009 Seventh Annual Communication Networks and Services Research Conference*, (CNSR '09), pages 153–157, 2009.

[32] Byeungwoo Jeon and David A. Landgrebe. Fast parzen density estimation using clustering-based branch and bound. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(9):950–954, 1994.

[33] Leif Jonsson, David Broman, Kristian Sandahl, and Sigrid Eldh. Towards automated anomaly report assignment in large complex systems using stacked generalization. In *Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on*, pages 437–446. IEEE, 2012.

[34] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proc. of the Thirtieth international conference on Very large data bases*, volume 30, pages 180–191. VLDB Endowment, 2004.

[35] Fabian Kloosterman, Stuart P Layton, Zhe Chen, and Matthew A Wilson. Bayesian decoding using unsorted spikes in the rat hippocampus. *Journal of neurophysiology*, 111(1):217–227, 2014.

[36] Ahmed E Kosba, Ahmed Saeed, and Moustafa Youssef. Rasid: A robust wlan device-free passive motion detection system. In *Pervasive computing and communications (PerCom), 2012 IEEE international conference on*, pages 180–189. IEEE, 2012.

[37] Matej Kristan and Ale Leonardis. Online discriminative kernel density estimator with gaussian kernels. *Cybernetics, IEEE Transactions on*, 44(3):355–365, 2014.

[38] Matej Kristan and Aleš Leonardis. Multivariate online kernel density estimation. In *Proc. of Computer Vision Winter Workshop*, pages 77–86, 2010.

[39] John Krumm and Eric Horvitz. Predestination: Inferring Destinations from Partial Trajectories. In *Proceedings of the 8th International Conference on Ubiquitous Computing*, (UbiComp'06), pages 243–260, 2006.

[40] Ludmila I Kuncheva. Change detection in streaming multivariate data using likelihood detectors. *Knowledge and Data Engineering, IEEE Transactions on*, 25(5):1175–1180, 2013.

[41] Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

[42] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[43] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. WhereNext: A Location Predictor on Trajectory Pattern Mining. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (KDD '09), pages 637–646, 2009.

[44] Daniel W Moran and Andrew B Schwartz. Motor cortical representation of speed and direction during reaching. *Journal of Neurophysiology*, 82(5):2676–2692, 1999.

[45] May Moussa and Moustafa Youssef. Smart cevices for smart environments: Device-free passive detection in real environments. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[46] Nicolas Navet and Shu-Heng Chen. On predictability and profitability: Would gp induced trading rules be sensitive to the observed entropy of time series? *Natural Computing in Computational Finance*, pages 197–210, 2008.

[47] Nam Nguyen, Svetha Venkatesh, and Hung Bui. Recognising behaviours of multiple people with hierarchical probabilistic model and statistical data association. In *BMVC 2006: Proceedings of the 17th British Machine Vision Conference*, pages 1239–1248. British Machine Vision Association, 2006.

[48] John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 34(1):171–175, 1971.

[49] Liam Paninski, Jonathan Pillow, and Jeremy Lewi. Statistical models for neural encoding, decoding, and optimal stimulus design. *Progress in brain research*, 165:493–507, 2007.

[50] Jian Pei, Helen Pinto, Qiming Chen, Jiawei Han, Behzad Mortazavi-Asl, Umeshwar Dayal, and Mei-Chun Hsu. PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. In *Proceedings of the 17th International Conference on Data Engineering*, (ICDE '01), pages 215–224, 2001.

[51] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 27–38. ACM, 2013.

[52] Rodrigo Quian Quiroga and Stefano Panzeri. Extracting information from neuronal populations: information theory and decoding approaches. *Nature Reviews Neuroscience*, 10(3):173–185, 2009.

[53] Stefan Roth. Discrete-continuous Optimization for Multi-target Tracking. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, (CVPR '12), pages 1926–1933, 2012.

[54] Andrew R Runnalls. Kullback-leibler approach to gaussian mixture reduction. *Aerospace and Electronic Systems, IEEE Transactions on*, 43(3):989–999, 2007.

[55] Adam Sadilek and John Krumm. Far out: Predicting long-term human mobility. In *AAAI*, 2012.

[56] Yusuke Sakamoto, Ken ichi Fukui, Joao Gama, Daniela Nicklas, Koichi Moriyama, and Masayuki Numao. Concept drift detection with clustering via statistical change

detection methods. In *Proc. of The seventh international conference on knowledge and systems engineering, KSE2015*, 2015.

[57] Emilio Salinas and LF Abbott. Vector reconstruction from firing rates. *Journal of computational neuroscience*, 1(1):89–107, 1994.

[58] Terence David Sanger. Probability density estimation for the interpretation of neural population codes. *Journal of Neurophysiology*, 76(4):2790–2793, 1996.

[59] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T. Campbell. NextPlace: A Spatio-temporal Prediction Framework for Pervasive Systems. In *Proceedings of the 9th International Conference on Pervasive Computing*, (Pervasive'11), pages 152–169, 2011.

[60] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[61] David W Scott and Simon J Sheather. Kernel density estimation with binned data. *Communications in Statistics-Theory and Methods*, 14(6):1353–1359, 1985.

[62] Shy Shoham, Liam M Paninski, Matthew R Fellows, Nicholas G Hatsopoulos, John P Donoghue, and Richard A Normann. Statistical encoding model for a primary motor cortical brain-machine interface. *Biomedical Engineering, IEEE Transactions on*, 52(7):1312–1322, 2005.

[63] Joseph Sill, Gábor Takács, Lester Mackey, and David Lin. Feature-weighted linear stacking. *arXiv preprint arXiv:0911.0460*, 2009.

[64] BW Silverman. Algorithm as 176: Kernel density estimation using the fast fourier transform. *Applied Statistics*, pages 93–99, 1982.

[65] Danaipat Sodkomkham, Roberto Legaspi, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. App: Aperiodic and periodic model for long-term human mobility prediction using ambient simple sensors. In *The Fourth International Workshop on Mining Ubiquitous and Social Environments*, page 3, 2013.

[66] Danaipat Sodkomkham, Roberto Legaspi, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. Predictability analysis of aperiodic and periodic model for long-term human mobility using ambient sensors. In *Mining, Modeling, and Recommending'Things' in Social Media*, pages 131–149. Springer, 2015.

[67] Danaipat Sodkomkham, Roberto Legaspi, Satoshi Kurihara, and Masayuki Numao. A study on next location predictive modeling using mined temporal sequential patterns as input to a decision tree.

[68] Danaipat Sodkomkham, Roberto Legaspi, Satoshi Kurihara, and Masayuki Numao. A study on activity predictive modeling for prompt and delayed services in smart space. In *Theory and Practice of Computation*, pages 266–278. Springer, 2013.

[69] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[70] L. Song, U. Deshpande, U.C. Kozat, D. Kotz, and R. Jain. Predictability of WLAN Mobility and Its Effects on Bandwidth Provisioning. In *Proceedings of the 25th IEEE International Conference on Computer Communications.*, (INFOCOM '06), pages 1–13, 2006.

[71] Eran Stark and Moshe Abeles. Predicting movement from multiunit activity. *The Journal of neuroscience*, 27(31):8387–8394, 2007.

[72] Wilson Truccolo, Uri T Eden, Matthew R Fellows, John P Donoghue, and Emery N Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of neurophysiology*, 93(2):1074–1089, 2005.

[73] Valérie Ventura. Spike train decoding without spike sorting. *Neural computation*, 20(4):923–963, 2008.

[74] Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1100–1108. ACM, 2011.

[75] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[76] W Wu, A Shaikhouni, JP Donoghue, and MJ Black. Closed-loop neural control of cursor motion using a kalman filter. In *Proc. of Engineering in Medicine and Biology Society, IEMBS'04. 26th Annual International Conference of the IEEE*, volume 2, pages 4126–4129. IEEE, 2004.

[77] Min Xu, Hisao Ishibuchi, Xin Gu, and Shitong Wang. Dm-kde: dynamical kernel density estimation by sequences of kde estimators with fixed number of components over data streams. *Frontiers of Computer Science*, 8(4):563–580, 2014.

[78] Shoou-I Yu, Yi Yang, and Alexander Hauptmann. Harry Potter's Marauder's Map: Localizing and Tracking Multiple Persons-of-Interest by Nonnegative Discretization. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, (CVPR '13), pages 3714–3720, 2013.

[79] Kai Zhang and James T Kwok. Simplifying mixture models through function approximation. In *Proc of Advances in Neural Information Processing Systems*, pages 1577–1584, 2006.

[80] Kechen Zhang, Iris Ginzburg, Bruce L McNaughton, and Terrence J Sejnowski. Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. *Journal of neurophysiology*, 79(2):1017–1044, 1998.

[81] Yan Zheng, Jeffrey Jestes, Jeff M Phillips, and Feifei Li. Quality and efficiency for kernel density estimates in large data. In *Proc. of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 433–444. ACM, 2013.

[82] Aoying Zhou, Zhiyuan Cai, Li Wei, and Weining Qian. M-kernel merging: Towards density estimation over data streams. In *Proc. of Database Systems for Advanced Applications, DASFAA 2003. Eighth International Conference on*, pages 285–292. IEEE, 2003.

List of Publications

Journal Paper

1. Danaipat Sodkomkham, Davide Ciliberti, Matthew A. Wilson, Ken-ichi Fukui, Koichi Moriyama, Masayuki Numao, and Fabian Kloosterman. "Kernel Density Compression for Real-time Bayesian Encoding/Decoding of Unsorted Hippocampal Spikes" (in press)
   URL: http://www.sciencedirect.com/science/article/pii/S0950705115003524
   doi:10.1016/j.knosys.2015.09.013

Conference and Workshop Proceedings

1. Danaipat Sodkomkham, Roberto Legaspi, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. "Predictability Analysis of Aperiodic and Periodic Model for Long-Term Human Mobility Using Ambient Sensors", Proc. the 4th International Workshops, MUSE 2013, Prague, Czech Republic, September 23, 2013, and MSM 2013, Paris, France, May 1, 2013, Revised Selected Papers

2. Roberto Legaspi, Danaipat Sodkomkham, Kazuya Maruo, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. "Time-Interval Clustering in Sequential Pattern Recognition Towards Predictive Modeling of Human Characteristics", Theory and Practice of Computation, pp. 174-186. Springer Japan, 2012.

Presentations

1. Danaipat Sodkomkham, Roberto Legaspi, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. "APP: Integrated Aperiodic and Periodic Model for Long-Term Human Mobility Prediction Using Ambient Simple Sensors", Proc.

the 4th International Workshop on Mining Ubiquitous and Social Environments, p. 3. 2013.

2. Danaipat Sodkomkham, Roberto Legaspi, Satoshi Kurihara, and Masayuki Numao. “A Study on Next Location Predictive Modelingusing Mined Temporal Sequential Patterns as input to a Decision Tree”, International Organized Session “Application Oriented Principles of Machine Learning and Data Mining”. JSAI 26 (2012): 1-5.

3. Roberto Legaspi, Danaipat Sodkomkham, Kazuya Maruo, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. “Clustering Multiple and Flexible Time Intervals in Sequential Patterns Towards Predictive Modeling of Human Gait Behavior”, International Workshop on Finding Patterns of Human Behaviors in Network and Mobility Data (NEMO) (held in conjunction with ECML-PKDD2011), Athens, Greece, September 9 2011

4. Kazuya Maruo, Danaipat Sodkomkham, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. “Mining Frequent Sequences with Flexible Time Intervals”, The 1st International Workshop of Sensor Data Mining (IWSDM2011) held in conjunction with The 8th International Conference on Networked Sensing Systems (INSS2011), Penghu, Taiwan, Taiwan, June. 12 2011

5. Kazuya Maruo, Danaipat Sodkomkham, Ken-ichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao. “Mining Frequent Sequences with Flexible Time Intervals”, The 5th International Workshop on Data-Mining and Statistical Science (DMSS2011), Osaka, March 29-30 2011