Automatic extraction of behavioral patterns for elderly mobility and daily routine analysis

Chen Li

Follow this and additional works at: https://repository.hkbu.edu.hk/etd_oa

Recommended Citation
https://repository.hkbu.edu.hk/etd_oa/510

This Thesis is brought to you for free and open access by the Electronic Theses and Dissertations at HKBU Institutional Repository. It has been accepted for inclusion in Open Access Theses and Dissertations by an authorized administrator of HKBU Institutional Repository. For more information, please contact repository@hkbu.edu.hk.
DATE: June 8, 2018

STUDENT'S NAME: LI Chen

THESIS TITLE: Automatic Extraction of Behavioral Patterns for Elderly Mobility and Daily Routine Analysis

This is to certify that the above student's thesis has been examined by the following panel members and has received full approval for acceptance in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Chairman: Prof. Chiu Sung Nok
Professor, Department of Mathematics, HKBU
(Designated by Dean of Faculty of Science)

Internal Members: Prof. Xu Jianliang
Professor, Department of Computer Science, HKBU
(Designated by Head of Department of Computer Science)

Dr. Chen Li
Assistant Professor, Department of Computer Science, HKBU

External Members: Prof. Poupart Pascal
Professor
David R. Cheriton School of Computer Science
University of Waterloo
Canada

Prof. Lam Wai
Professor
Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong

Proxy: Dr. Tam Hon Wah
Associate Professor, Department of Computer Science, HKBU

In-attendance: Dr. Cheung Kwok Wai
Associate Professor, Department of Computer Science, HKBU

Issued by Graduate School, HKBU
Automatic Extraction of Behavioral Patterns for Elderly Mobility and Daily Routine Analysis

LI Chen

A thesis submitted in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Principal Supervisor: Dr. William Kwok-Wai CHEUNG

Hong Kong Baptist University

June 2018
DECLARATION

I hereby declare that this thesis represents my own work which has been done after registration for the degree of PhD at Hong Kong Baptist University, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications.

I have read the University's current research ethics guidelines, and accept responsibility for the conduct of the procedures in accordance with the University's Committee on the Use of Human & Animal Subjects in Teaching and Research (HASC). I have attempted to identify all the risks related to this research that may arise in conducting this research, obtained the relevant ethical and/or safety approval (where applicable), and acknowledged my obligations and the rights of the participants.

Signature: [Signature]

June 2018
Abstract

The elderly living in smart homes can have their daily movement recorded and analyzed. Given the fact that different elders can have their own living habits, a methodology that can automatically identify their daily activities and discover their daily routines will be useful for better elderly care and support. In this thesis research, we focus on developing data mining algorithms for automatic detection of behavioral patterns from the trajectory data of an individual for activity identification, daily routine discovery, and activity prediction.

The key challenges for the human activity analysis include the need to consider longer-range dependency of the sensor triggering events for activity modeling and to capture the spatio-temporal variations of the behavioral patterns exhibited by human. We propose to represent the trajectory data using a behavior-aware flow graph which is a probabilistic finite state automaton with its nodes and edges attributed with some local behavior-aware features. Subflows can then be extracted from the flow graph using the kernel $k$-means as the underlying behavioral patterns for activity identification. Given the identified activities, we propose a novel nominal matrix factorization method under a Bayesian framework with Lasso to extract highly interpretable daily routines. To better take care of the variations of activity durations within each daily routine, we further extend the Bayesian framework with a Markov jump process as the prior to incorporate the shift-invariant property into the model.

For empirical evaluation, the proposed methodologies have been compared with a number of existing activity identification and daily routine discovery methods based
on both synthetic and publicly available real smart home data sets with promising results obtained. In the thesis, we also illustrate how the proposed unsupervised methodology could be used to support exploratory behavior analysis for elderly care.

**Keywords:** Human activity identification, daily routine discovery, nominal matrix factorization, Bayesian inference, Markov jump process.
Acknowledgements

When I first came to Hong Kong Baptist University as a new graduate student, I knew little about research. During my undergraduate years, reading textbooks and studying hard are all I was expected to do. But now I was expected not only to understand the existing ideas, but also to come up with new ideas. The problem is how to achieve that. Thanks to my thesis advisor William Kwok-Wai CHEUNG who works closely with me on research and supports me on both academic and non-academic matters during my PhD years. He kindly showed me the ways of doing research, patiently went through my papers, and encouraged me. I have benefited immensely from his ideas and his feedback.

I also would like to thank Joseph Kee-Yin NG for providing support and guidance on my research projects. My thanks also go to Ji-Ming LIU, Byron Koon-Kau CHOI, Li CHEN for help, comments and advices.

I also would like to thank all my dear friends for providing support and friendship in all these years. Without you, the world is a wilderness. I also would like to thank the colleagues of Computer Science Department for their help during various stages of my studies.

I especially want to thank my mom Xiao LIANG, my dad Junping LI and my wife Fei HUANG for their unconditional love and support. Without your endless support, it was impossible to finish my studies abroad.
# Table of Contents

Declaration ............................ i

Abstract ................................. ii

Acknowledgements ........................ iv

Table of Contents ........................ v

List of Tables ............................ viii

List of Figures ........................... ix

Chapter 1  Introduction .................. 1

1.1 Smart home and data mining for assisted living ...................... 1

1.2 Unsupervised learning for elderly mobility and daily routine analysis . 2

1.3 Thesis organization ............................. 6

Chapter 2  Related work .................. 7

2.1 Process mining ............................ 8

2.2 Activity modeling ........................... 9

2.3 Daily routine discovery .......................... 10

2.4 Markov jump process .......................... 11

2.5 CASAS smart home test bed .......................... 11

Chapter 3  Elderly mobility analysis based on behavior-aware flow graph modeling .......................... 13
# 3.1 Introduction .................................................. 13

# 3.2 Learning flow graph via a state-merging approach ............... 14

## 3.2.1 Inferring the behavior-aware mobility flow graph .......... 15

## 3.2.2 Behavior-aware features as state attributes .................. 16

## 3.2.3 Detecting subflows as activities using the weighted kernel $k$-

   means ......................................................... 24

# 3.3 Experiments .................................................. 25

## 3.3.1 Real indoor trajectory data sets ........................... 26

## 3.3.2 Performance on activity identification ..................... 27

## 3.3.3 Periodic patterns of indoor movement behaviors .......... 33

# 3.4 Summary ..................................................... 38

---

# Chapter 4 Daily routine pattern discovery via matrix factorization 41

## 4.1 Introduction .................................................. 41

## 4.2 Bayesian nominal matrix factorization for mining daily routine patterns 42

### 4.2.1 Probabilistic nominal matrix factorization .................. 43

### 4.2.2 Label embedding ......................................... 44

### 4.2.3 An extended hierarchical model .......................... 45

### 4.2.4 Bayesian Lasso .......................................... 46

### 4.2.5 Inference .................................................. 48

### 4.2.6 Folding up the $d$ dimensions ............................ 49

## 4.3 Experimental results ......................................... 49

### 4.3.1 Evaluation of daily activity routine discovery ............. 49

### 4.3.2 Applicability to other data sets ........................ 58

## 4.4 Summary ..................................................... 67

---

# Chapter 5 Discovering shift-invariant daily routine patterns using

   Markov Jump Process ........................................... 68

## 5.1 Introduction .................................................. 68

## 5.2 Markov jump process (MJP) ................................... 69
## List of Tables

3.1 Statistics of real trajectory data sets. .............................................. 27
3.2 Resulting entropies of real trajectory data sets. ................................. 28
3.3 The portion of activity labels in each identified subflow extracted
   from dataset 1. .................................................................................... 31
3.4 The portion of activity labels in each identified frequent pattern ex-
   tracted from dataset 1. ........................................................................ 31
3.5 The portion of activity labels in each identified subflow extracted
   from dataset 2. .................................................................................... 32

4.1 Mean MAE(±SD) of smart home data. ................................................. 55
4.2 Mean MAE(±SD) of subsets of dataset 1. ............................................. 57
4.3 Mean MAE(±SD) of the lenses data set. .............................................. 67

5.1 Error rate(±SD) on activity prediction based on the smart home data
   sets. ......................................................................................................... 79
List of Figures

1.1 Representing trajectories of two different activities represented using
   i) frequent sequential patterns (left) and ii) our proposed method
   (right). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
1.2 The flow chart of the proposed methodology. . . . . . . . . . . . . . . . 6

2.1 A sample of sensor triggering events. . . . . . . . . . . . . . . . . . . 9
2.2 Part of a spaghetti-like process model inferred from. . . . . . . . . . . 9

3.1 A sensor triggering sequence involving sensors $a$, $b$ and $c$. . . . . . . . 17
3.2 Three PTAs for three segments of sensor triggering events. . . . . . . . 18
3.3 Frequencies of the unigrams in PTA 2 and PTA 3. . . . . . . . . . . . . 18
3.4 Frequencies of the bigrams in PTA 2. . . . . . . . . . . . . . . . . . . 18
3.5 Frequencies of the bigrams in PTA 3. . . . . . . . . . . . . . . . . . . 18
3.6 The data structure of a PTA. . . . . . . . . . . . . . . . . . . . . . . 19
3.7 Visualization of the FBLM feature sets at different states (upper:
   coarse grained; lower: fine grained). . . . . . . . . . . . . . . . . . . 21
3.8 Visualization of the RFBLM feature set at different states (upper:
   obtained from two overlapped segments $\hat{X}_1$ and $\hat{X}_2$; lower: combined
   ones). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
3.9 A subflow discovered from a smart home data set. . . . . . . . . . . . . 26
3.10 Sensitivity of subflow detection quality on different number of subflows. 28
3.11 Size of the resulting PDFA given different threshold values for state
   merging. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
3.12 Sensitivity of subflow detection quality on the threshold values. . . . 29
3.13 Performance comparison on subflow detection quality given different n-gram statistics adopted in RFBLM. 30
3.14 Similarity among activities of dataset 1 computed based on the KL divergence of the corresponding ground-truth labels. (Cells in deep blue color indicate pairs of similar activities.) 33
3.15 Similarity among activities of dataset 2 computed based on the KL divergence of the corresponding ground-truth labels. (Cells in deep blue color indicate pairs of similar activities.) 34
3.16 The coverage of the PDFA trained based on training data of different days. 35
3.17 Entropy / Coverage values of the PDFA trained from one month’s data (row) and tested on data of subsequent months (column). 35
3.18 Similarity matrices of the subflows obtained from 2010-12 and 2011-01. 36
3.19 Similarity matrices of the subflows obtained from 2011-01 and 2011-02. 36
3.20 Similarity matrices of the subflows obtained from 2011-02 and 2011-03. 37
3.21 Similarity matrices of the subflows obtained from 2011-03 and 2011-04. 37
3.22 Entropy / Coverage values of the PDFA trained from one week’s data (row) and tested on data of subsequent weeks (column) 38
3.23 Similarity matrices of the subflows obtained from week1 and week1. 39
3.24 Similarity matrices of the subflows obtained from week1 and week2. 39
3.25 Similarity matrices of the subflows obtained from week1 and week3. 40
3.26 Similarity matrices of the subflows obtained from week1 and week4. 40
4.1 The proposed Probabilistic Nominal Matrix Factorization for daily routine discovery. 44
4.2 The graphical model for factorization (daily routine discovery). 45
4.3 Weighted adjacency matrix of the labels for the synthetic data set. 52
4.4 Visualization of label embedding of synthetic data. 53
4.5 Basis vector reconstruction MAE of different algorithms at different noise levels based on SD3. 53
4.6 Data reconstruction MAE of different algorithms at different noise levels based on SD3. ................................................................. 54
4.7 Performance of PNoMF on dataset 1 with different numbers of basis vectors. ................................................................. 55
4.8 Performance of PNoMF on dataset 2 with different numbers of basis vectors. ................................................................. 56
4.9 Daily routines \( (U) \) of DEC 2010 ................................................................. 58
4.10 Coefficient matrix \( (V) \) of DEC 2010 ................................................................. 58
4.11 Daily routines \( (U) \) of JAN 2011 ................................................................. 58
4.12 Coefficient matrix \( (V) \) of JAN 2011 ................................................................. 59
4.13 Daily routines \( (U) \) of FEB 2011 ................................................................. 59
4.14 Coefficient matrix \( (V) \) of FEB 2011 ................................................................. 59
4.15 Daily routines \( (U) \) of MAR 2011 ................................................................. 60
4.16 Coefficient matrix \( (V) \) of MAR 2011 ................................................................. 60
4.17 Daily routines \( (U) \) of APR 2011 ................................................................. 60
4.18 Coefficient matrix \( (V) \) of APR 2011 ................................................................. 61
4.19 Daily routines \( (U) \) of dataset 2 ................................................................. 61
4.20 Coefficient matrix \( (V) \) of dataset 2 ................................................................. 61
4.21 Hamming distance matrix of the 435 representatives. ...................... 62
4.22 Data reconstruction MAE. ................................................................. 63
4.23 Distributions of voting records of two parties in 16 political issues. ... 64
4.24 Voting patterns \( U \) obtained by TMF. ................................................................. 65
4.25 Voting patterns \( U \) obtained by PNoMF. ................................................................. 66

5.1 Visualization of an MJP path. ................................................................. 72
5.2 Comparison of different methodologies on basis vector reconstruction MAE at different shift levels. ................................................................. 80
5.3 Comparison of different methodologies on matrix reconstruction MAE at different shift levels. ................................................................. 80
5.4 Comparison of different methodologies on activity prediction error
  rate at different shift levels. ........................................ 81
5.5 Comparison of different methodologies on basis vector reconstruction
  MAE at different noise levels. ....................................... 81
5.6 Comparison of different methodologies on matrix reconstruction MAE
  at different noise levels. ........................................... 82
5.7 Comparison of different methodologies on activity prediction error
  rate at different noise levels. ....................................... 82
5.8 Visualization of activities in data set 2. .......................... 84
5.9 Visualization of generator matrix of MJP corresponding to daily rou-
  tine #1. .............................................................. 85
5.10 Visualization of instances of daily routine #1. ...................... 86
5.11 Visualization of generator matrix of MJP corresponding daily rou-
  tine #2. .............................................................. 87
5.12 Visualization of instances of daily routine #2. ...................... 88
5.13 Visualization of generator matrix of MJP corresponding daily routine
  #3. .............................................................. 89
5.14 Visualization of instances of daily routine #3. ...................... 90
5.15 Visualization of generator matrix of MJP corresponding daily routine
  #4. .............................................................. 91
5.16 Visualization of instances of daily routine #4. ...................... 92
Chapter 1

Introduction

The advent of ubiquitous computing and sensor technologies have enabled new opportunities for human activity analysis. Related applications include daily activity pattern detection in city [34, 71], activity recognition [40], assisted living [74, 57, 48], etc. In particular, for assisted living, the elderly living in a smart home equipped with sensors can have their daily indoor movement logged and analyzed.

1.1 Smart home and data mining for assisted living

In the literature, there are a number of smart home projects reported. HomeMesh [27] aims to design a living spaces which can support the capabilities of elderly in order to enhance their quality of life. CASAS [65] monitors the consistency and completeness of a set of daily activity from dementia patients. IMMED [51] uses a wearable camera to monitor the loss of cognitive capabilities of dementia patients via assessing some instrumented activities of daily living. These projects are mostly for assisted living where the elderly living in a smart home equipped with sensors can have their daily indoor movement logged and analysed for better elderly care. Analyzing the logs can help detect potential abnormal behaviors which could be caused by unfavorable health situations [74, 57]. For related studies, daily routines
are often extracted to provide decision support for better elderly care [48]. In the literature, different computational tools have been developed for the analysis. In particular, a number of supervised learning methods have been adopted and found effective for activity analysis and recognition [4, 13]. However, there are limitations when putting them into practice. Manually creating labeled data for the training is time-consuming. Also, analyzing human activities is often of exploratory nature, where the activity labels are simply unknown in advance. Thus, unsupervised learning methods have been explored for automatic activity identification [54, 67].

1.2 Unsupervised learning for elderly mobility and daily routine analysis

In this thesis, we focus on elderly mobility and daily routine analysis using unsupervised learning methods in a smart home setting. It means that we do not assume any prior knowledge on the activities and the daily routines to be identified. The only assumption we make is that a person when performing an activity at home (e.g., preparing a meal) should move around according to some regularity of his own. We call this kind of regularity a behavioral pattern. Our first goal is to detect behavioral patterns from the observed sensor triggering events to identify the underlying activities. Furthermore, how such behavioral patterns (activities) appear within a day should follow some routines (e.g., routines for ordinary days and routines for weekends). Our second goal is to discover such daily routines from the data.

The activity identification task (Goal 1) is challenging as different activities of the same person are performed in the same smart home. The trajectory segments, even though corresponding to different activities, are highly likely to have some short patterns of sensor triggering sharing among them. In order to differentiate the trajectory segments and detect behavioral patterns which are more activity specific, one needs to characterize each sensor triggering event by considering also the triggering of its nearby sensors over a longer time window before and after its triggering
(or in other words a longer range). Also, a person seldom moves in exactly the same way to perform the same activity, and spatio-temporal variations of the behavioral pattern are unavoidable (e.g., one may occasionally stay at different positions in the kitchen for occasionally longer or shorter time). All these make the conventional frequent pattern based method which represents a particular activity as a cluster of similar discrete frequent patterns not directly applicable. As an illustration, Figure 1.1 shows a few sequences of sensor triggering events of two different activities. Representations obtained based on the frequent sequential pattern method are shown on the lower left. It is obvious that the representations of the two activities are hard to be distinguished. The key problem is that the ordering information of the appearance of the sequential patterns is discarded.

To alleviate the aforementioned challenges, we propose to infer from the indoor trajectory data (time-stamped sensor triggering events of an individual) a behavior-aware mobility flow graph. We represent the mobility flow graph as a probabilistic finite state automaton (PDFA) where its nodes are attributed with features characterizing some local movement features. Behavioral patterns are represented as subflows in the flow graph. Activities are identified by detecting subflows embedded in the flow graph. Our conjecture is that an activity can be characterized by a set of states (local movement patterns) where, once reached one of them, there is a high chance of staying within and transiting among the states for a while before leaving. To detect the subflows, we adopt a weighted kernel $k$-means algorithm. As illustrated in Figure 1.1, using our proposed method, the flow graphs inferred for the two activities can retain the ordering information and obviously end up with two distinct enough representations.

For daily routine discovery (Goal 2), it is also non-trivial even though most of the people simply apply matrix factorization techniques. On one hand, it carries the errors from the preceding activity identification task. Also, the assumption that the basis vectors extracted by existing matrix factorization methods are corresponding to some interpretable daily routines may not be valid. Thirdly, different parts of
Figure 1.1: Representing trajectories of two different activities represented using i) frequent sequential patterns (left) and ii) our proposed method (right).

A daily routine may not appear at exactly the same time each day. So, in general, a more robust factorization technique is needed. And the problem will be further complicated if interleaving activities are considered and the environment is dynamic (say with moving objects or other individuals).

In this thesis, we first propose a probabilistic nominal matrix factorization method. In particular, after detecting the subflows as activities, we tag each sensor triggering event in the trajectory data with an activity label as detected. A matrix with each column being a sequence of daily activity labels ordered by the time within a day is first formed, can be factorized into basis factors as daily routines. While it is common to decompose each sequence of labels (nominal data) into multiple sequences of binary data (one for each label) as in [14] to convert the matrix to binary, we factorize the nominal data matrix directly. On one hand, we avoid multiplying the size of the data. Also, we want to capture the correlation of the activities, which is hard to be done after converting it into multiple binary matrices. In the literature, most of the matrix factorization techniques deal with matrices with continuous real values (with few exceptions [47, 55]). To perform nominal matrix factorization directly, we assume that the similarity of detected subflows can be estimated. We embed the discrete labels (or nominal values) onto a d-dimensional continuous space accordingly. Then, a hierarchical probabilistic model is proposed for the factorization, which is
important as the activity labels identified for the sensor triggering events are noisy. A Bayesian Lasso is introduced to ensure the sparsity of the coefficient matrix, and thus the interpretability of the discovered basis factors (i.e. daily routines).

Further, we notice that people often carry out daily routines with variations in activities start time and durations. Such dynamic properties impose a fundamental challenge to all the existing daily routine discovery methods based on matrix factorization. We need methods to alleviate the inaccuracy caused by the “shifts”. We extend the nominal matrix factorization framework by modeling such variations for extracting shift-invariant daily routines. Markov jump process which is a continuous extension of discrete time Markov chain is introduced as the prior of the basis vectors to model the variations of activity durations. Daily routines are formed by observed instances with highest likelihood with respect to Markov jump process. To carry out the model inference, we adopt Gibbs sampling.

Figure 1.2 shows a flow chart summarizing the key steps of the methodology proposed in this thesis. In a nutshell, we first infer the mobility flow graph to summarize the mobility traces of an individual. A set of “behavior-aware” features that characterize the local movement patterns are proposed to guide the graph inference so as to infer a compact flow graph and at the same time to preserve the specificity of the “behavior-aware” features per state as far as possible. A weighted kernel $k$-means algorithm is then utilized to detect subflows in the flow graph for activity identification. With the labels of the identified activities marked on the trajectory data, we rearrange the data in a matrix form and propose a novel probabilistic nominal matrix factorization method and one of its extensions for discovering daily routines with high interpretability.

We evaluate the effectiveness of the proposed methodology using several publicly available smart home data sets that contain movement trajectories of an elder living in a smart home. Our experimental results show that our proposed approach can detect subflows which are more specific in terms of their correspondence to activities when compared with an existing frequent pattern clustering approach [67]. Also, we
Figure 1.2: The flow chart of the proposed methodology.

benchmark the performance of the proposed nominal matrix factorization method and its Markov jump process extension for daily routine discovery with a number of existing matrix factorization methods using both synthetic and real smart home data sets. Highly promising results in terms of the accuracy of the extracted basis vectors of discrete labels and next activity prediction are obtained.

1.3 Thesis organization

The remaining thesis is organized as follows. Chapter 2 describes the related work. Chapter 3 presents the proposed methodology for inferring the behavior-aware mobility flow graph. For daily routine pattern discovery, a novel nominal matrix factorization is proposed in Chapter 4. Chapter 5 presents a methodology of using Markov jump process to achieve shift-invariant daily routine discovery by using Markov jump process. Chapter 6 concludes the thesis with possible future extensions.
Chapter 2

Related work

In the literature, there have been quite some studies on human mobility analysis using data mining methods. For instance, macroscopic human mobility patterns of human were extracted from trajectories of mobile phones to support better urban planning and infection control [25, 72]. Algorithms for better organizing trajectory data have been developed so that important movement trends can be visualized and tracked [56]. With the recent advent of location-aware social networks, data about people’s whereabouts have become much more accessible. That further triggers new applications like anomalous trajectory pattern detection [15], discovery of interesting places [82], gathering pattern analysis [86], inferring social ties [80], discovering urban functional zones [83], among others. The aforementioned projects focus on outdoor activities. Indoor trajectory data can also be collected in a smart home setting [71, 62]. Related applications include detection of behavioral deviations [68], daily routine analysis [84], etc. In this thesis, we focus on developing data mining methodologies for indoor mobility analysis. In the following, we provide literature reviews on related topics including process mining, activity modeling, daily routine discovery and Markov jump process.
2.1 Process mining

Process mining is a field highly related to this work. It aims to discover, monitor, and improve real processes by extracting flow related knowledge from event logs readily available in today’s information systems [77]. As compared with classical data mining techniques like clustering, classification, regression and etc., process mining was first proposed for business process modeling and analysis. There are also some research groups in the process mining community focusing on human behavior analysis [36, 37, 31] with the objective to capture the human’s reasoning process behind their behaviors.

However, the process mining algorithms found in the literature face challenges when applied to activity modeling. One issue is that it is hard to interpret the results obtained by the conventional process mining algorithms [78]. Figures 2.1 and 2.2 show how a process model (in the form of a flow graph) can be recovered from a smart home data set (Figure 2.1) using an existing process mining algorithm. Most of the process mining algorithms use primarily the ordering information of the appearance of Sensor_id. So, the result could only reflect the local relationships between two consecutive sensor triggerings. In reality, there are many other forms of variations found in the mobility patterns which are however ignored. This motivates the need for algorithms which can allow process models at different granularity levels to be obtained so that one can gain better understanding of the underlying processes.

Another issue of applying process mining algorithms to indoor trajectory data is that it is hard to separate concurrent and interleaved mobility traces [49, 18]. Due to the privacy concern and/or limitation of sensor hardware, it is common that there are no identifiers in the sensor events associating them to the corresponding individuals. While recovering the identities of the individuals is not the goal, properly reconstructing the correspondence of the sensor triggering events could be essential to gain correct understanding of the underlying human behaviors.
2.2 Activity modeling

As discussed in Chapter 1, modeling longer-range dependency among the sensor triggering events is an important issue to address for modeling behavioral patterns. Pastra et al. [59] explained the similarity between human activities and languages with respect to the sequence representations and the grammatical structures they share. In the literature, probabilistic grammar models have been widely used for representing languages so that variations as well as longer-range dependency in the observed sentences can be properly modeled and captured. They have also been applied to human activity analysis, including activity recognition [60, 39], gesture
recognition [28], maneuver recognition [32], activity segmentation [61], and activity prediction [44]. Also, different inference methods [7, 75] have been proposed for the model estimation. In this thesis, we extend probabilistic finite state automata [79] so that they can better model indoor human activities.

2.3 Daily routine discovery

For related work on daily routine discovery, different approaches have been propose in the literature. For example, matrix factorization (MF) [14, 85] models the human’s daily life as a set of basis vectors. The basis vectors with higher coefficient values correspond to the main routines of the person. Topic modeling [73, 17] coupled location and time together as word has been applied to discover routines as topics. Also the hybrid approach [16, 66] makes use of Latent Dirichlet Allocation (LDA) to discover the routine patterns of all individuals and models some selected groups of individuals by Author Topic model (ATM). In this thesis research, we focus on MF framework, which is an effective data analysis approach which has been studied for the past decades. Some recently proposed ones include maximum-margin matrix factorization [81], probabilistic matrix factorization [70], sparse probabilistic matrix factorization [35], as well as variants of nonnegative matrix factorization (NMF) [46, 6, 30, 23]. Most of them can only work for matrices with continuous values. In our case, we need to factorize nominal matrices with activity labels. In the literature, there exist few exceptions where factorization of matrices with non-continuous values are studied. Boolean matrix factorization [52] introduces the boolean operation to deal with binary valued matrices. It was then extended to handle ordinal cases by introducing a new set of operations [2]. Ternary matrix factorization (TMF) [50] uses three-valued logic recursively to approximate the discrete valued matrix with hard constraints such that each coefficient should be either zero or one. Ordinal matrix factorization (OMF) [55] can be formulated under a hierarchical probabilistic framework to model matrices with their elements taking a finite ordered set of values. The methodologies proposed in Chapters 4 and
5 are inspired by OMF. We leverage on a reasonable assumption that the similarity among the discrete labels is known (or can be estimated) so that the underlying nominal matrix can be factorized under a Bayesian framework.

2.4 Markov jump process

Discrete-time models have been wildly used for capturing the underlying mechanisms of different dynamic systems based on some observed event sequences. Models such as hidden Markov model [63] and its extensions such as dynamic Bayesian networks [53], factorial hidden Markov model [20], infinite hidden Markov model [1], etc. have been used for various applications. For example, location prediction can be achieved by using a Markov model to considering both spatial and temporal information [19]. Markov chain can be used in [10] to recognize activities from sensor triggering event sequences. However, all these models ignore the stochastic variation of duration of staying at each state. While one can discretize the time line into a finite set of consecutive time steps for the modeling, this is obviously not an effective approach in general, especially when the inter-event intervals are highly non-uniform. Markov jump process (MJP) is a continuous extension of discrete-time Markov chain where the timing information is also modeled, which has been adopted to model the transitions among states of dynamic systems. For example, MJP was used in [45] to model vehicular mobility among urban areas divided by the intersections of roads. It was also used to detect the periods of time in which a particular event process is active [33], and to model the interactions among chemical species [24]. In Chapter 5, we model the switchings between activities as the state changes in MJP so that variations in activity duration can be captured accordingly.

2.5 CASAS smart home test bed

Two publicly available smart home data sets which were collected via two test beds from Washington State University’s Center for Advanced Studies in Adaptive
Systems (CASAS) project [65] are used for this study. Each test bed is a single resident apartment equipped with binary sensors (e.g., passive infrared sensors), such sensor will be triggered if the resident in a corresponding area. There is one resident aged over 73 years performs his/her normal activities in each test bed. The CASAS collects these sensor triggering events in an unobtrusive way for further analysis. The two data sets we obtained contain 200 and 52 days sensor triggering events respectively, both of them are partially labeled with activity labels like meal preparation, work, sleep, etc.
Chapter 3

Elderly mobility analysis based on behavior-aware flow graph modeling

3.1 Introduction

In this chapter, we propose an unsupervised learning methodology for extracting behavioral patterns as representations of human daily activities from indoor trajectory data. The underlying challenges include the stochastic nature of human mobility, in particular their spatial and temporal variations even for the same activity to be repeated by the same person. Other challenges include the presence of sensor noises and errors, dynamic properties of the environment (moving objects or other individuals), etc. All these make conventional frequent pattern mining methods which represent a particular activity as a cluster of similar discrete frequent patterns not directly applicable.

To address the abovementioned challenges, we propose the use of a behavior-aware flow graph which is a probabilistic finite state automaton (PDFA) with the nodes attributed with local behavioral features. A state-merging approach is adopted for inferring a compact PDFA where states with similar local behavioral
features are merged during the inference. Behavioral patterns are then detected as subflows in the flow graph. The conjecture is that an activity is characterized by a set of states where, once reached one of them, there is a high chance of transiting and staying among the states before leaving them. We adopt a weighted kernel $k$-means algorithm in particular for the subflow extraction.

We evaluate the effectiveness of the proposed methodology using a publicly available smart home data set that contains digital trajectories of an elder living in a smart house for 219 days. Our experimental results show that our proposed approach can detect subflows which are more specific in terms of their correspondence to activities when compared with a recently proposed frequent pattern clustering approach [67].

The remaining chapter is organized as follows. Section 3.2 presents the proposed methodology for inferring the behavior-aware mobility flow model. Experimental results and discussions can be found in Section 3.3. Section 3.4 concludes the chapter.

3.2 Learning flow graph via a state-merging approach

Considering sequences of sensor triggering events as strings of alphabets generated by a stochastic sequence model, we infer a sequence model so that the probability distribution over the event sequences can be optimized. In principle, with the model inferred, tasks like identifying the most probable movement in the next step given a location can be supported. Among different sequence models, the probabilistic automaton is one of the representative ones and adopted in this paper.

Definition 3.2.1. A Deterministic Finite Automaton (DFA) $A$ is a 5-tuple, $(Q, E, \delta, q_0, F)$, where $Q$ is a finite set of states, $E$ is an alphabet, $q_0 \in Q$ is the initial state, $\delta : Q \times E \rightarrow Q$ is a transition function, and $F \subseteq Q$ is the set of final states. A Prefix Tree Acceptor (PTA) is a tree-like DFA generated by all the
prefixes of the observed strings as states, which can only accept the observed strings.

**Definition 3.2.2.** A Probabilistic Deterministic Finite Automaton (PDFA) is a 5-tuple \( A = (Q, E, \delta, \pi, q_0, F) \) where \( Q \) is a finite set of states, \( E \) is an alphabet, \( \delta : Q \times E \rightarrow Q \) is a transition function, \( \pi : Q \times E \rightarrow [0,1] \) is the probability of the next symbol given a state, \( q_0 \in Q \) is the initial state, and \( F : Q \rightarrow [0,1] \) is the end of string probability function.

Among the existing PDFA inference algorithms, we adopt the ALERGIA algorithm [7]. We first build a PTA from each observed sequence. To generalize for strings other than those observed, the algorithm introduces a *merge* operation. Let \( n_i \) denote the number of strings arriving at state \( q_i \), \( f_i(a) \) the number of strings following edge \( \delta_i(a) \) where \( a \) is the edge’s symbol, and \( f_i(\#) \) the number of strings ending at state \( q_i \). The probabilities for the string terminating at or leaving state \( q_i \) can be computed as \( f_i(\#)/n_i \), and \( f_i(a)/n_i \) respectively. According to ALERGIA, for each pair of states \((q_i, q_j)\) with common outgoing edges, they are *compatible* for merging if the probabilities of leaving them are close enough as controlled by a threshold parameter \( \theta \) as the confidence of the test. Specifically, the compatible test of ALERGIA is defined as follow,

\[
|\frac{f_i}{n_i} - \frac{f_j}{n_j}| < \left( \sqrt{\frac{1}{n_i}} + \sqrt{\frac{1}{n_j}} \right) \cdot \sqrt{\frac{1}{2} \ln \frac{2}{\theta}}
\]

where \( f_i \) equals either \( f_i(\#) \) or \( f_i(a) \), \( f_j \) is either \( f_j(\#) \) or \( f_j(a) \), and the threshold \( \theta \) controls the confidence of the test. A smaller \( \theta \) will block more merge operations, resulting in a larger PDFA. The search-and-merge process continues until there is no more possible state pair that can be merged. In the following sections, we explain how we modify ALERGIA for our application.

### 3.2.1 Inferring the behavior-aware mobility flow graph

To infer a mobility flow graph from the trajectory data, we consider the temporally ordered sensor triggering events over a fixed time interval per day as an observed
sequence, \( X = \{x_1, x_2, ..., x_M \} \) where each element corresponds to a sensor ID. Given \( N \) day observations, \( N \) observed sequences denoted as \( D = \{X^1, X^2, ..., X^N\} \).

We then infer a set of corresponding PTAs and the corresponding PDFA. As an illustration, given \( X = \{a, b, a, b, a, b\} \) as visualized in Figure 3.1. We first define for each of these sensor triggering events a particular edge. Then, we add a state denoted as \( q = (SID_q, I_q, O_q, F_{I_q}, F_{O_q}) \) with a unique state ID (SID) between each consecutive pair of the edges to link them up using the incoming and outgoing edges stored in \( q \) as \((I_q, O_q)\) to form an initial PTA. Initially, the frequencies of encountering \( I_q \) and \( O_q \) \((F_{I_q}, F_{O_q})\) are both set to be one, which are to be updated during the flow graph inference. The corresponding PTA is depicted as PTA 3 in Figure 3.2. Then, in principle, we can proceed with the state merging operation of the standard ALERGIA algorithm to obtain the PDFA. However, this will end up with a PDFA containing states which are not location specific, which is not desirable. The main reason is that the ALERGIA algorithm pairs up and merges states with the same outgoing edge labels. In our application, a state change happens when the sensor event associated with an incoming edge is triggered. To maintain the states after being merged to be location specific, states are paired up with the same \textit{incoming} edges labels \( I_q \) instead. In addition, we further propose a set of features characterizing the local movement to attribute each state, and only the states with their features close enough are to be merged. For the design of such “behavior-aware” features, spatio-temporal properties of local movement are considered, as detailed in the next section.

3.2.2 Behavior-aware features as state attributes

Merging states by considering only identical incoming edge labels will end up with trajectories of different activities which cross each other to be “tied” up. We propose a set of “behavior-aware” features to differentiate the states based on the spatio-temporal properties of their local movements. To compute that, we model a trajectory as a sequence of \textit{segments} where each segment is a \textit{locally maximal}
subsequence of triggering events associated with \( T \) different labels. A subsequence is *locally maximal* if any further sequential extension of the subsequence will end up with more than \( T \) different labels. To allow the local behavioral context (sensor triggering events before and after one particular sensor triggering event) to be smoothed over time, we use *overlapping segments*. For instance, if there is a subsequence ‘aaabbbccc’, the two overlapping segments will be ‘aaabbb’ and ‘bbbecc’.

If the subsequence is ‘abababccbc’, the two overlapping segments will be ‘ababab’ and ‘bcbcbe’.

Figure 3.2 presents three segments of sensor triggering events represented by three PTAs with \( T = 2 \). Referring to states \( q_{12} \) in PTA 1 and \( q_{34} \) in PTA 3, their incoming edges are both labeled with sensor \( a \). Considering merely their incoming edges, the two states could be merged. If we look at the sensor triggering events along the whole segment instead, \( q_{12} \) is a state corresponding to the situation of staying still near sensor \( b \), whereas \( q_{34} \) is corresponding to the situation of frequently hopping between sensors \( a \) and \( b \). The two situations are in fact quite different. We need to define some “behavior-aware” features so that states like \( q_{12} \) and \( q_{34} \) should not be merged.

<table>
<thead>
<tr>
<th>Time</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Sensor triggering sequence (PTA 3)} )</td>
<td>( \text{b} )</td>
<td>( \text{a} )</td>
<td>( \text{b} )</td>
<td>( \text{c} )</td>
<td>( \text{c} )</td>
<td>( \text{a} )</td>
</tr>
</tbody>
</table>

**Figure 3.1:** A sensor triggering sequence involving sensors \( a \), \( b \) and \( c \).

The key idea is to make use of some statistics of the time spending near each of the nearby sensors as the behavior-aware features. In particular, we first compute the occurrence frequency of each sensor as the proxy of the estimated time staying near the sensor. Then, the frequency value will be discounted if the corresponding triggering events do not occur consecutively. For instance, Figure 3.3 shows the occurrence frequencies of sensors \( a \) and \( b \) in PTA 2 and PTA 3, and their unigram statistics are identical. We then consider length-2 sequential patterns, or bigrams, as
Figure 3.2: Three PTAs for three segments of sensor triggering events.

PTA 1: \( q_{11} \rightarrow q_{12} \rightarrow q_{13} \rightarrow q_{14} \rightarrow q_{15} \rightarrow q_{16} \rightarrow q_{17} \)

PTA 2: \( q_{21} \rightarrow q_{22} \rightarrow q_{23} \rightarrow q_{24} \rightarrow q_{25} \rightarrow q_{26} \rightarrow q_{27} \)

PTA 3: \( q_{31} \rightarrow q_{32} \rightarrow q_{33} \rightarrow q_{34} \rightarrow q_{35} \rightarrow q_{36} \rightarrow q_{37} \)

Figure 3.3: Frequencies of the unigrams in PTA 2 and PTA 3.

Figure 3.4: Frequencies of the bigrams in PTA 2.

Figure 3.5: Frequencies of the bigrams in PTA 3.
shown in Figures 3.4 and 3.5. We use the bigram statistics to discount the unigram statistics for more accurate estimation of the corresponding portion of time staying close to each of nearby sensors. For instance, seeing more instances of pattern \((a, b)\) will end up with more discount on the time span near sensor \(a\) while seeing more instances of pattern \((a, a)\) should imply less discount.\(^1\)

The mathematical formulation of the proposed behavior-aware feature set is defined as follow.

Let \(\hat{X} = (x_1, x_2, ..., x_n)\) denote an observed segment represented as an ordered set of sensor triggering events, \(Y = (y_1, y_2, ..., y_m)\) the set of distinct sensor labels within \(\hat{X}\), \(SP = (y_i \in Y, y_j \geq i \in Y)\) the set of distinct 2-length sequential patterns of sensor IDs where \((y_i, y_j)\) and \((y_j, y_i)\) are considered equivalent and grouped,\(^2\) \(SP^a\) the subset of \(SP\) containing the sequential patterns in \(SP\) with at least one of the events labeled as \(a\), \(F_{y_i}\) the occurrence frequency of \(y_i\) in \(\hat{X}\), and \(F(\text{SP}_{ij})\) the occurrence frequency of the order-invariant sequential pattern \((y_i, y_j)\) in \(\hat{X}\).

\(^1\)And in general, one can consider \(n\)-gram statistics.

\(^2\)This is because we are considering the discounting factor.
For each sequential pattern $SP_{ij} = (y_i, y_j)$, we define $\rho_{yx}(SP_{ij})$ as the portion of time staying near $y_x \in Y$ within $SP_{ij}$, given as:

$$\rho_{yx}(SP_{ij}) = \frac{|\{y_x \cap SP_{ij}\}|}{\text{length}(SP_{ij})}$$

where $y_x \cap SP$ denotes that the element $y_x$ performs an element-wise intersection with the set $SP$.

So given a particular state $q$ with the label of the incoming edge $I_q \in Y$, the portion of time staying near $y_x$ averaged over $\hat{X}$ is estimated as:

$$\varrho_q(y_x) = \sum_{SP_{ij} \in SP^I_q} \rho_{yx}(SP_{ij})P(SP_{ij}|SP^I_q)F(SP_{ij}) = \frac{\sum_{SP_{ij} \in SP^I_q} \rho_{yx}(SP_{ij})F(SP_{ij})}{\sum_{SP_{ij} \in SP^I_q} F(SP_{ij})}.$$  \hfill (3.2.1)

**Definition 3.2.3.** Frequency Based Local Mobility (FBLM) feature set for state $q$ is defined as

$$f_q = \{(y_1, f_q(y_1)), \ldots, (y_{|Y|}, f_q(y_{|Y|}))\}$$  \hfill (3.2.2)

where

$$f_q(y_i) = \frac{\varrho_q(y_i)F(y_i)}{\sum_{y \in Y} \varrho_q(y_i)F(y_i)}.$$

Note that this formulation implies that the states within a segment will share the same feature set if they have the same incoming edge label. For more fine-grained modeling of the local context within one segment, $\varrho_q(y_x)$ in Eq.3.2.1 can be computed over a moving window with reference to state $q$ instead of $SP^I_q$. Figure 3.7 shows the two versions of the FBLM feature sets assigned to different states within a segment. The upper ones correspond to the less fine-grained modeling version and the lower ones correspond to the more fine-grained one.

Definition 3.2.3 can be further extended by leveraging the prior knowledge on the spatial arrangement of the installed sensors so that nearby sensors (with different labels) can still be matched where the missing feature values can be estimated via
“smoothing”. For instance, in PTA 2 of Figure 3.2, we extend the feature set from \(\{a, b\}\) to \(\{a, b, c\}\). More specifically, if sensor \(c\) is the common neighbor of sensors \(a\) and \(b\), we set the feature value \(f_q(c) = \Delta\) for state \(q \in \{q_{22}, q_{23}, ..., q_{27}\}\). If \(c\) is only the neighbor of \(b\), we set \(f_q(c) = \Delta \exp(-d(c, b))\) for only \(\{q_{25}, q_{26}, q_{27}\}\) where \(d(c, b)\) is the shortest path length between \(c\) and \(b\). For 1-hop neighbors, \(d = 1\). Also, overlapping segments can be considered so as to allow a longer range of local behavioral context propagation. As each state will have two FBLM feature sets defined, we combine them via a weighted average based on the length of the overlapping segments as shown in Figure 3.8. This version of feature set can give more robust results, with the formal definition given as:

**Definition 3.2.4. Robust Frequency Based Local Mobility (RFBLM):** Let \(C = \{c_i\}\) denote the set of common neighbors of the sensors in \(Y\) and \(C' = \{c'_i\}\)
denote the set of the neighbors of sensor $I_q$. We further define the Robust Frequency Based Local Mobility (RFBLM) feature set for state $q$ that exists in both segments $\hat{X}_i$ and $\hat{X}_{i+1}$ as

$$f_q^r = \{ (y_1, f_q^r(y_1)), \ldots, (y_{|V|}, f_q^r(y_{|V|})),
\quad (c_1, f_q^{rc}(c_1)), \ldots, (c_{|C|}, f_q^{rc}(c_{|C|})),
\quad (c'_1, f_q^{r'c}(c'_1)), \ldots, (c'_{|C'|}, f_q^{r'c}(c'_{|C'|})) \} \quad (3.2.3)$$

where

$$f_q^r(y_i) = \frac{|\hat{X}_i| * f_q(y_i)|_{\hat{X}_i} + |\hat{X}_{i+1}| * f_q(y_i)|_{\hat{X}_{i+1}}}{|\hat{X}_i| + |\hat{X}_{i+1}|}$$

$$f_q^{rc}(c_i) = \Delta$$

$$f_q^{r'c}(c'_i) = \Delta * \exp(-d(c'_i, I_q))$$

where $\Delta$ is the smoothing constant.

In this paper, we consider only 1-hop neighborhood. Depending on the spatial arrangement of the sensors, wider neighborhood can also be exploited. Also, the feature values in the feature set are normalized for each state before the next step of processing.

The overall procedure of computing the behavior-aware flow graph are summarized in Algorithms 1 and 2. We first divide the whole trajectory data into a set of sequences, each corresponding to a trajectory for one day. Prefix Tree Acceptor (PTA) is built based on such sequences as illustrated Algorithm 2, where $x_{i,j}$ corresponds to the $j$-th record of the $i$-th sequence. The RFBLM feature set for each state would be calculated once the PTA is built. Figure 3.6 shows the data model for each state in a PTA. After that, we scan the PTA to merge pairs of compatible states so that (1) the labels on their incoming edges should be identical in order to make sure they are corresponding to the same location, and (2) the $L_2$ distance computed between their RFBLM feature sets should be less than a threshold $\theta$ to ensure only behaviorally similar states to be merged. The time complexity of our approach is $O(n^2)$ and the space complexity is $O(n)$ which is linear with the number of records in the data set.
ALGORITHM 1: The Overall State-Merging Algorithm

**Input:** a set of observed sequences $D$, $\theta > 0$

**Output:** a PDFA $A$

$A \leftarrow \text{BuildingPTA}(D)$;

$Red \leftarrow \{q_0\}$;

$Blue \leftarrow \{q_a : q_a \in \delta(q_0, E)\}$;

while $(q_b \in Blue)$ is not empty do

if $\exists q_r \in Red : I_r = I_{q_r} \& \& |f_r - f_{q_r}|_2 > \theta$ then

$A \leftarrow \text{StateMerge}(A, q_r, q_b)$;

else

$Red \leftarrow Red \cup \{q_b\}$;

end

$Blue \leftarrow \{q_a \in \delta(q_b, E) \cap q_b \in Red\} \setminus Red$;

end

ALGORITHM 2: Building Prefix Tree Acceptor (PTA)

**Input:** a set of observed sequences $D$

**Output:** a PTA $A=(Q, E, q_0, \delta)$

$Q \leftarrow q_0, E \leftarrow \emptyset, QID \leftarrow 1$;

for $x_{i,j} \in D$ do

$QID \leftarrow QID + 1$;

if $j == 1$ then

$\delta(q_0, x_{i,j}) \leftarrow q_{QID}$

end

$\delta(q_{QID-1}, x_{i,j}) \leftarrow q_{QID}$;

$Q \leftarrow Q \cup q_{QID}$;

$E \leftarrow E \cup x_{i,j}$;

end

Calculate RFBLM for $q \in Q \setminus q_0$ based on Eq. 3.2.3;
3.2.3 Detecting subflows as activities using the weighted kernel \( k \)-means

The PDFA obtained as explained in the previous section summarizes the observed sequences as a directed flow graph. With the conjecture that the mobility pattern of an activity can be represented as a subflow in the flow graph, we propose to identify them by applying a graph partitioning method. Here we define a subflow as a subgraph where the number of edges within the subgraph is relatively higher than the number of edges going in and out. In the context of activity modeling, a subflow corresponds to a group of states where an individual once getting in will have a higher chance to move according to the state transitions modeled by the subflow before moving out.

To perform the graph partitioning, we extend a weighted kernel \( k \)-means algorithm [12] to work on the directed flow graph. As compared to the spectral clustering implementation, the weighted kernel \( k \)-means algorithm is more desirable as the high computational cost to compute the eigenvectors for a large matrix for obtaining the minimum \( k \)-cut can be avoided. Let \( Q \) be the finite set of states of the inferred PDFA \( \mathcal{A} \) and \((Q_1, Q_2, ..., Q_t)\) the set of \( t \) disjoint subflows where their union is \( Q \). The \( k \)-cut can be obtained as

\[
kCut(\mathcal{A}) = \min_{Q_1, ..., Q_t} \sum_{c=1}^{t} \frac{\text{links}(Q_c, Q\setminus Q_c) + \text{links}(Q\setminus Q_c, Q_c)}{\text{deg}^+(Q_c) + \text{deg}^-(Q_c)}
\]  

(3.2.4)

where \( \text{links}(Q_u, Q_v) \) are the sum of the frequency counts on the transitions between \( Q_u \) and \( Q_v \), \( \text{deg}^+(Q_c) \) is the sum of the out-degree of the states in the cluster \( Q_c \), \( \text{deg}^-(Q_c) \) is the sum of the in-degree of the states in the cluster \( Q_c \). Given that \( \text{links}(Q_c, Q\setminus Q_c) = \text{deg}^+(Q_c) - \text{links}(Q_c, Q_c) \) and \( \text{links}(Q\setminus Q_c, Q_c) = \text{deg}^-(Q_c) - \text{links}(Q_c, Q_c) \), the \( k \)-cut can thus be obtained by maximizing with respect to \((Q_1, Q_2, ..., Q_t)\) the criterion:
\[
\sum_{c=1}^{t} \frac{\text{links}(Q_c, Q_c)}{\text{deg}^+(Q_c) + \text{deg}^-(Q_c)}
\]
\[
= \sum_{c=1}^{t} \frac{x_c^\top M x_c}{x_c^\top D^+ x_c + x_c^\top D^- x_c}
\]
\[
= \sum_{c=1}^{t} \frac{x_c^\top M x_c}{x_c^\top D x_c}
\]
\[
= \sum_{c=1}^{t} \tilde{x}_c^\top M \tilde{x}_c
\]

where \( M \) is the adjacency matrix storing the transition frequencies of the states in \( A \), \( x_c \) is an indicator vector with its \( i \)-th element taking the value 1 if cluster \( c \) contains state \( i \) or 0 otherwise, \( D^+ \) is a diagonal matrix with \( D_{ii}^+ = \sum_{j=1}^{n} M_{ij} \), \( D^- \) is a diagonal matrix with \( D_{ii}^- = \sum_{j=1}^{n} M_{ji} \), \( D = D^+ + D^- \), and \( \tilde{x}_c = x_c / (x_c^\top D x_c)^{1/2} \).

According to [12], it can be shown that the weighted kernel \( k \)-mean algorithm can be formulated as a trace maximization problem as

\[
\max_{\mathcal{D}^{1/2} \tilde{X}} \text{trace}((D^{1/2} \tilde{X})^\top D^{1/2} \phi^\top \phi D^{1/2} (D^{1/2} \tilde{X})) + \text{constant} \quad (3.2.5)
\]

where \( \phi^\top \phi \) is the kernel matrix for the data points, \( \tilde{x}_c \) is the \( c \)-th column of \( \tilde{X} \) and \( D \) is a diagonal matrix. Thus, one can just use \( M \) to replace \( \phi^\top \phi \) in Eq.3.2.5 and the subflow extraction can readily be solved using the weighted kernel \( k \)-means algorithm.

Figure 3.9 shows a subflow extracted from a real trajectory data set (upper) and the corresponding movement pattern in the smart home (lower). The location of the sensors are marked with \( M0XX \) in the floor plan, and each edge in the subflow is associated with a sensor label and the transition probability. It is obvious that the discovered subflow is corresponding to the activity of meal-preparation.

3.3 Experiments

In this section, we present the empirical results on evaluating the performance of the proposed methodology using several publicly available real data sets.
3.3.1 Real indoor trajectory data sets

We apply the proposed methodology to two publicly available data sets as described in Section 2.5. The two data sets are partially labeled with activity labels like meal preparation, work, sleep, etc. In this paper, we use the labels only for evaluation. Statistics of the two data sets are summarized in Table 3.1. For pre-processing, we first chop the whole data set into subsequences by day. Since the data set contains also traces of occasional visitors, we consider them as noise. To remove the corresponding sensor triggering events, we pre-compute a nearest neighbor graph of the sensors. If the geodesic distance between two consecutive events is larger than two (i.e., more than two sensors apart), we detect a new visitor and start tracing the triggering events of the nearby sensors as outliers.
### Table 3.1: Statistics of real trajectory data sets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of records</th>
<th># of sensors</th>
<th># of activities</th>
<th>durations (day)</th>
<th>sampling interval (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset1</td>
<td>716,798</td>
<td>35</td>
<td>11</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>Dataset2</td>
<td>598,988</td>
<td>27</td>
<td>17</td>
<td>52</td>
<td>5</td>
</tr>
</tbody>
</table>

#### 3.3.2 Performance on activity identification

To measure the effectiveness of the proposed methodology for identifying activities, we make use of the ground-truth activity labels and compute the entropy value as the performance metric, defined as

\[
Entropy = \sum_{i=1}^{t} \frac{n_i}{n} E(Q_i)
\]

\[
E(Q_i) = -\frac{1}{\log t} \sum_{j=1}^{t} \frac{n_j^i}{n_i} \log \frac{n_j^i}{n_i}
\]

where \(t\) is the number of distinct activity labels, \(n_j^i\) is the number of records in the subflow \(Q_i\) with the ground-truth label of activity \(j\), and \(n_i\) is the number of records in \(Q_i\). The overall entropy value is the sum of the individual subflow entropies weighted by the subflow size. If almost all the nodes in a detected subflow are associated with the same activity label, it means that the subflow is specifically corresponding to one activity and will give the lowest entropy value. If the labels of the nodes in a subflow are evenly distributed among the possible activities, it will be hard to say if the subflow is representing an activity or not, and the entropy value will be high.

In our experiments, we use 90% of the data for training the PDFA and the remaining 10% for testing (that is computing the entropy value). As there are some parameters needed to be set for the proposed methodology, we conduct the experiments to determine the optimal parameter setting. We first evaluate the use of different orders of statistics in defining the RFBLM feature set. According to Figure 3.13, we found that as higher order \(n\)-grams are used, the corresponding RFBLM feature can give a significantly lower entropy value in general. It is consistent to our understanding that considering a longer range of sensor triggering events for
representing behavioral patterns can give a more accurate model, which inevitably incurs additional computational cost. In addition, we conduct experiments to test the effect of different numbers of subflows \((t)\) on the sensitivity of the subflow detection quality (Figure 3.10), and the effect of different merging threshold values \((\theta)\) on the size of the inferred flow graph (Figure 3.11) and the subflow detection quality (Figure 3.12). According to Figure 3.10, the entropy value decreases until the number of subflows is sufficiently large \((t = 13)\). As shown in Figure 3.11, a higher value of \(\theta\) results in a smaller PDFA which is obvious as the algorithm will accept more state merging tests. For the entropy value, we see that it fluctuates within a small range given different values of \(\theta\) as shown in Figure 3.12. In the sequel, we set \(n = 4, t = 13\) and \(\theta = 0.08\).

For performance comparison, we implemented a frequent pattern clustering approach (FP) [67] as the baseline. We first demonstrate the effectiveness of adopting the proposed feature set (RFBLM) for representing each state to infer the flow graph.
Figure 3.11: Size of the resulting PDFA given different threshold values for state merging.

Figure 3.12: Sensitivity of subflow detection quality on the threshold values.
Figure 3.13: Performance comparison on subflow detection quality given different n-gram statistics adopted in RFBLM.

and then the embedded subflows. The entropy values of the clusters and subflows identified using FP and RFBLM respectively are shown in Table 3.2. Tables 3.3 and 3.5 show the portions of the ground-truth of activity labels mapped to each subflow based on the PDFA trained using dataset 1 and dataset 2 respectively. We observed that each identified subflow is mainly mapped to one or two activity labels. In particular, the identified subflows SF7, SF8 and SF12 in Table 3.5 are characterizing specifically the activities “Eat”, “Cook” and “Leave Home” respectively. While there are some subflows corresponding to the same activity (e.g. SF8, SF10 and SF11 in Table 3.5), they are in fact capturing three different patterns of leaving home via three different doors of the home. In the data set, they are all labelled as “Leave Home”. If we keep increasing the value of $t$, some trivial subflows which are not corresponding to any ground truth activity will be resulted. We also notice that some activities cannot be distinguished by our proposed method as they share very similar movement information. To verify that, we compute the similarities (based on the KL divergence) among activities according to the ground-truth labels as shown in Figure 3.14 and Figure 3.15. For example, in dataset 2, we find that the activities “Bathe”, “Personal Hygiene” and “Groom” are very similar with respect
<table>
<thead>
<tr>
<th></th>
<th>SF1</th>
<th>SF2</th>
<th>SF3</th>
<th>SF4</th>
<th>SF5</th>
<th>SF6</th>
<th>SF7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eating</td>
<td>0.01</td>
<td>0.04</td>
<td>0.62</td>
<td>0.10</td>
<td>0.02</td>
<td></td>
<td>0.25</td>
</tr>
<tr>
<td>Leave_Home</td>
<td>0.62</td>
<td>0.67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.06</td>
</tr>
<tr>
<td>Meal_Preparation</td>
<td>0.01</td>
<td>0.17</td>
<td>0.09</td>
<td>0.01</td>
<td>0.01</td>
<td>0.69</td>
<td></td>
</tr>
<tr>
<td>Relax</td>
<td>0.31</td>
<td>0.11</td>
<td>0.29</td>
<td>0.90</td>
<td></td>
<td>0.33</td>
<td></td>
</tr>
<tr>
<td>Resperate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sleeping</td>
<td>0.05</td>
<td></td>
<td>0.08</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work</td>
<td>0.01</td>
<td></td>
<td>0.05</td>
<td>0.66</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3.3: The portion of activity labels in each identified subflow extracted from dataset 1.

<table>
<thead>
<tr>
<th></th>
<th>Pattern 1</th>
<th>Pattern 2</th>
<th>Pattern 3</th>
<th>Pattern 4</th>
<th>Pattern 5</th>
<th>Pattern 6</th>
<th>Pattern 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eating</td>
<td>0.188</td>
<td>0.212</td>
<td>0.126</td>
<td>0.415</td>
<td>0.000</td>
<td>0.213</td>
<td>0.256</td>
</tr>
<tr>
<td>Leave_Home</td>
<td>0.120</td>
<td>0.059</td>
<td>0.000</td>
<td>0.000</td>
<td>0.252</td>
<td>0.032</td>
<td>0.000</td>
</tr>
<tr>
<td>Meal_Preparation</td>
<td>0.015</td>
<td>0.302</td>
<td>0.276</td>
<td>0.000</td>
<td>0.135</td>
<td>0.200</td>
<td>0.005</td>
</tr>
<tr>
<td>Relax</td>
<td>0.077</td>
<td>0.200</td>
<td>0.252</td>
<td>0.023</td>
<td>0.196</td>
<td>0.207</td>
<td>0.049</td>
</tr>
<tr>
<td>Resperate</td>
<td>0.000</td>
<td>0.001</td>
<td>0.226</td>
<td>0.355</td>
<td>0.000</td>
<td>0.032</td>
<td>0.000</td>
</tr>
<tr>
<td>Sleeping</td>
<td>0.283</td>
<td>0.201</td>
<td>0.000</td>
<td>0.000</td>
<td>0.005</td>
<td>0.002</td>
<td>0.690</td>
</tr>
<tr>
<td>Work</td>
<td>0.317</td>
<td>0.016</td>
<td>0.120</td>
<td>0.210</td>
<td>0.412</td>
<td>0.316</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 3.4: The portion of activity labels in each identified frequent pattern extracted from dataset 1.

to the distribution of sensors triggered by them.

While we can use the identified activity labels as inputs for the subsequent daily routine discovery task, we group SF8, SF10 and SF11 in Table 3.5 to be all labeled as “Leave Home” for the ease of latter interpretation. In principle, we can also label SF8, SF10 and SF11 as different ways of “Leave Home”. We do similar groupings for the two data sets. Finally, we identified 5 and 8 activities in dataset 1 and dataset 2 respectively.
<table>
<thead>
<tr>
<th>Activity Label</th>
<th>SF1</th>
<th>SF2</th>
<th>SF3</th>
<th>SF4</th>
<th>SF5</th>
<th>SF6</th>
<th>SF7</th>
<th>SF8</th>
<th>SF9</th>
<th>SF10</th>
<th>SF11</th>
<th>SF12</th>
<th>SF13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bathe</td>
<td>0.10</td>
<td>0.09</td>
<td>0.05</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bed_Toilet_Transition</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cook</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.40</td>
</tr>
<tr>
<td>Dress</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>Eat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.30</td>
</tr>
<tr>
<td>Groom</td>
<td>0.23</td>
<td>0.02</td>
<td>0.10</td>
<td>0.13</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Leave_Home</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.98</td>
</tr>
<tr>
<td>Morning_Meds</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personal_Hygiene</td>
<td>0.47</td>
<td>0.03</td>
<td>0.54</td>
<td>0.46</td>
<td>0.01</td>
<td>0.29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.11</td>
</tr>
<tr>
<td>Relax</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td>Sleep</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.32</td>
</tr>
<tr>
<td>Take_Medicine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>Toilet</td>
<td>0.16</td>
<td>0.22</td>
<td>0.15</td>
<td>0.71</td>
<td>0.02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.05</td>
</tr>
<tr>
<td>Wash_Dishes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.13</td>
</tr>
<tr>
<td>Watch_TV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.26</td>
</tr>
<tr>
<td>Work</td>
<td>0.02</td>
<td>0.07</td>
<td>0.01</td>
<td>0.19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.07</td>
</tr>
</tbody>
</table>

Table 3.5: The portion of activity labels in each identified subflow extracted from dataset 2.
3.3.3 Periodic patterns of indoor movement behaviors

Since people’s indoor behaviors for a period of time are anticipated to follow some periodic patterns, we first try to see how a person’s daily behaviors repeats within a month. For each month, we keep increasing the number of days being used to train the PDFA and would like to see if the PDFA inferred can “cover” the behaviors to appear in the remaining days of the month. Here, we define the notion coverage as the percentage of the test data that can be generated by the inferred PDFA. Given that the PDFA failed to generate after passing $n_f$ sensor triggering events, the coverage is defined as

$$Coverage = 1 - \frac{n_f}{n} \quad (3.3.6)$$
where $n$ is the total number of records in the testing data. According to Figure 3.16, we found that for most of the months, after the data for around the first 7 days were used for training the PDFA, its coverage will reach a relatively high level and then increases slowly as more data are further used for the training.

Next, we train a PDFA for each month and test its coverage and entropy based on the data of the current and the subsequent months. According to Figure 3.17, we observe that the coverage value keeps at a high level and the entropy value fluctuates around 0.55. For example, the coverage and entropy values of the PDFA trained based on December’s data and tested on itself would be 99.27% and 0.577 respectively. As the coverage and the entropy values do not allow us to compare the identified subflows obtained based on the data of different months, we adopt the similarity measure proposed in [3] and weighted it by RFBLMs.

We plot the similarity matrix between the sets of subflows of every two consecutive months as shown in Figure 3.18 to Figure 3.21. We can observe that some...
common behaviors whose patterns remain more or less the same for a number of months.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2010.12</td>
<td>99.27%</td>
<td>93.28%</td>
<td>93.72%</td>
<td>92.22%</td>
<td>93.24%</td>
</tr>
<tr>
<td></td>
<td>0.577</td>
<td>0.658</td>
<td>0.568</td>
<td>0.580</td>
<td>0.593</td>
</tr>
<tr>
<td>2011.01</td>
<td>99.27%</td>
<td>92.91%</td>
<td>92.59%</td>
<td>92.62%</td>
<td>92.90%</td>
</tr>
<tr>
<td></td>
<td>0.498</td>
<td>0.539</td>
<td>0.596</td>
<td>0.565</td>
<td></td>
</tr>
<tr>
<td>2011.02</td>
<td></td>
<td>99.21%</td>
<td>93.78%</td>
<td>92.90%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.569</td>
<td>0.592</td>
<td>0.561</td>
<td></td>
</tr>
<tr>
<td>2011.03</td>
<td></td>
<td></td>
<td>99.40%</td>
<td>93.31%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.560</td>
<td>0.606</td>
<td></td>
</tr>
<tr>
<td>2011.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.35%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.543</td>
</tr>
</tbody>
</table>

Figure 3.16: The coverage of the PDFA trained based on training data of different days.

Figure 3.17: Entropy / Coverage values of the PDFA trained from one month’s data (row) and tested on data of subsequent months (column).

For between-week similarity, we train PDFAs using the data of the first week of each month and test the coverage and entropy values using the data of the remaining weeks of that month. According to Figure 3.22, we observe that the coverage value keeps at a high level again and the entropy value fluctuates around 0.45. Also, according to Figure 3.23 to Figure 3.26, we find that the resident exhibits different behavior patterns when compared with those of the first week.
Figure 3.18: Similarity matrices of the subflows obtained from 2010-12 and 2011-01.

Figure 3.19: Similarity matrices of the subflows obtained from 2011-01 and 2011-02.
Figure 3.20: Similarity matrices of the subflows obtained from 2011-02 and 2011-03.

Figure 3.21: Similarity matrices of the subflows obtained from 2011-03 and 2011-04.
3.4 Summary

In this chapter, an unsupervised learning methodology is proposed to detect behavioral patterns for activity identification. Our experimental results show that the subflows obtained by applying a weighted kernel $k$-means to the proposed behavior-aware mobility flow graph can identify more activity-specific behavioral patterns than the existing frequent pattern clustering approach.
Figure 3.23: Similarity matrices of the subflows obtained from week1 and week1.

Figure 3.24: Similarity matrices of the subflows obtained from week1 and week2.
Figure 3.25: Similarity matrices of the subflows obtained from week1 and week3.

Figure 3.26: Similarity matrices of the subflows obtained from week1 and week4.
Chapter 4

Daily routine pattern discovery via matrix factorization

4.1 Introduction

After detecting the subflows as activities as presented in Chapter 3, we tag each sensor triggering event in the trajectory data with an activity label as detected and then carry out the daily routine discovery task. A matrix with each column being a sequence of daily activity labels ordered by the time within a day is formed. To carry out daily routine analysis based on the acquired sequences of nominal data, the conventional way [14] is to decompose each sequence of nominal data (labels) into multiple sequences of binary data (one for each label) so that the existing matrix factorization techniques can (at least) be applicable. However, such a practice will multiply the size of the data, causing the tractability issue. In addition, the observed labels are often correlated due to, for example, their spatial or behavioral closeness. How to leverage the label correlation for factorizing multiple binary matrices is not obvious, and thus often ignored. In the literature, most of the recently proposed matrix factorization techniques are mostly dealing with matrices with real values, with few exceptions including [47] [55].

In this chapter, we propose to factorize the nominal data matrix directly in this
chapter. On one hand, we avoid multiplying the size of the data. Also, we want to capture the correlation of the activities, which is hard to be done after converting it into multiple binary matrices. To achieve that, we assume that the similarity of detected subflows can be estimated. We embed the discrete labels (or nominal values) onto a $d$-dimensional continuous space. Then, a hierarchical probabilistic model is proposed for the factorization, which is important as the activity labels identified for the sensor triggering events are noisy. To carry out the model inference, Gibbs sampling is adopted.

In the experiment, we benchmark the performance of the proposed methodology with a number of matrix factorization techniques with highly promising results obtained in terms of the accuracy of extracting basis vectors of nominal values as daily activity routines using both synthetic and benchmarking data sets.

The remaining chapter is organized as follows. Section 4.2 describes the probabilistic nominal matrix factorization model and its inference. Section 4.3 gives the experimental results on a synthetic data set and a real smart home data set. Section 4.4 concludes the chapter.

### 4.2 Bayesian nominal matrix factorization for mining daily routine patterns

A daily routine refers to a sequential pattern of activities that a person follows within a day. To discover daily routines, we first make use of the identified activity subflows and detect their occurrence in the data. The data $D^{M \times N}$ is then converted to traces of occurrence of $t$ distinct activity labels in $L = (L_1, ..., L_t)$. We denote the converted data as $D_A^{M \times N}$ where each column corresponds to a sequence of subflow labels for one particular day. To do the conversion, for each column, we first find the states corresponding to the location of the first sensor triggering event. Then, states which can generate the longest sub-sequence based on the transition function $\delta$ would have their subflow labels assigned to the sensor triggering events. We continue
this process until every record in $D^{M\times N}$ is assigned with a subflow label. We notice
that this label assignment process can occasionally end up with occasional transient
jumps between two activity subflows due to the relatively strong assumption of hard-
partitioning for the subflow identification. We eliminate these transient jumps by
prohibiting jumps if they are shorter than one minute.

After that, each element in $D_A$ will take one of the labels in $L$ as its value,
forming a nominal matrix. We assume that $D_A$ could have noise and missing labels
as discussed before. Thus, a true label in $D_A$ could be replaced by another one.
We further assume that the noise is not uniformly distributed over all the possible
labels but follows a distribution so that those more similar to the true one will have
a higher probability to be the replacement. The research problem here is how to
factorize the nominal matrix $D_A^{M\times N}$ into $K(<< \min(M, N))$ basis vectors (or daily
routines) $U_{K\times M}$ with each element of the vectors corresponding to one of the labels
in $L$ as its value, so that matrix $D_A$ can be “well” reconstructed by combining the
basis vectors via a coefficient matrix $V_{K\times N}$.

4.2.1 Probabilistic nominal matrix factorization

As presented in Chapter 2, there exist a lot of algorithms for matrix factorization
[69, 41, 70]. However, work on factorizing matrices with nominal values is rare.
While one can just convert each nominal label to a numerical value and then factorize
the matrix, this is invalid as the multiplication and addition operations on labels
are undefined. With the assumption that the similarity among the labels is known
or can be estimated, we can embed the labels onto a $d$-dimensional continuous
vector space so that at least some ordering of the labels can be recovered. The
value of $d$ needed is to be set so that the similarity relations of the labels can
be preserved. A higher value of $d$ can keep the similarity information better in
general. With the embedding performed, we can replace all the value of labels
in the nominal matrix $D_A$ by the corresponding vectors in the embedding space
and factorize it. The problem becomes similar to the ordinal matrix factorization
problem where solutions exist. Our ultimate goal is to come up with a factorization which can ensure that (i) the basis vectors obtained are robust to noise caused by the imperfect activity identification step and that (ii) each row of the matrix $D_A$ can be well-represented by a small number of basis vectors as sparse as possible for enhancing the interpretability of basis vectors. The former suggests the use of a probabilistic model and compute the factorization by $\arg \max_{U,V} p(D_A|U,V)$. The latter suggests that special attention is needed to handle how the basis vectors of the continuous values should be converted back to the labels with the interpretability considered. We propose a probabilistic nominal matrix factorization with a Bayesian Lasso adopted. The flowchart is illustrated in Figure 4.1.

![Flowchart](image)

**Figure 4.1**: The proposed Probabilistic Nominal Matrix Factorization for daily routine discovery.

### 4.2.2 Label embedding

To embed the labels onto a vector space, we start with the inferred mobility flow graph and shrink the nodes in each activity subflow to from a meta-node. The set of meta-nodes will give a new graph $A'$ where the nodes are corresponding to the $t$ subflows ($Q := Q_1, Q_2, ..., Q_t$) and the edges corresponding to links among the sub-flows which are weighted by their frequencies. We apply the node2vec algorithm [26] to embed the $t$ distinct labels onto a $d$-dimensional continuous space. The resulting embeddings of the labels can encapsulate local and global structural relationships among the nodes (for our case the similarity of the labels) as far as possible. It then gives a matrix $\mu := [\mu^{L_1}, ..., \mu^{L_t}]$ of size $t \times d$, where $\mu^{L_i} : Q_i \rightarrow \mathbb{R}^d$.

Then, we replace each element in $D_A$ by the corresponding $d$-dimensional vector $\mu(m, n)$, resulting in a new matrix $D_\mu = \{\mu(m, n) \in \mathbb{R}^d\}$ with each element being a
vector as visualized in Figure 4.1. By further unfolding $D_{\mu}$ to $[D_{\mu 1}; D_{\mu 2}; ...; D_{\mu d}]$, a matrix $D^{unfold}$ of dimension $M \times d \times N$ will be resulted. This unfolding step allows most of the existing matrix related techniques to be applicable again. However, it also implies that the $d$ dimensions of the vector representation are considered independently during the factorization. In general, it is not straightforward to ensure that the basis vectors obtained by the factorization can be “folded” back to give a valid label representation.

4.2.3 An extended hierarchical model

![Graphical model for factorization](image)

Figure 4.2: The graphical model for factorization (daily routine discovery).

We use a hierarchical model for the factorization. By assuming that the $d$ dimen-
sions of the vector representations of the label are independent, we model the (soft) range of values of the \( j \)-th dimension of \( \mu^{L_i} \), denoted as \( r_j^{L_i}(i = 1, ..., t, j = 1, ..., d) \), using a normal distribution \( \mathcal{N}(\mu_j^{L_i}, \sigma_j^{L_i}) \), where \( \mu_j^{L_i} \) is the projected value of label \( L_i \) on the \( j \)-th dimension, and \( \sigma_j^{L_i} \) indicates the standard deviation of the range corresponding to \( L_i \). By defining \( g_j(m,n) \) as a latent variable to indicate the particular range that \( D_{\mu_j}(m,n) \) falls into, the conditional probability of \( r_j(m,n) = L_i \) given \( g_j(m,n) \) is

\[
p(r_j(m,n) = L_i | g_j(m,n)) = \frac{\mathcal{N}(g_j(m,n); \mu_j^{L_i}, \sigma_j^{L_i})}{\sum_i \mathcal{N}(g_j(m,n); \mu_j^{L_i}, \sigma_j^{L_i})} \quad (4.2.1)
\]

where the uncertainty of \( g_j(m,n) \) is modeled as

\[
p(g_j(m,n)|h_j(m,n)) = \mathcal{N}(h_j(m,n), 1). \quad (4.2.2)
\]

\( h_j(m,n) \) can be interpreted as the mean value of \( g_j(m,n) \). We then recover the basis vectors from the matrix \( h \). Specifically, \( h_j(m,n) \) is modeled as a linear combination of matrices \( U_j \) and \( V \) with rank \( K \), given as:

\[
h_j(m,n) = U_j(\cdot, m)^\top V(\cdot, n) + \varepsilon_j(m,n)
\]

where \( X(\cdot, y) \) represents the whole \( y \)-th column of matrix \( X \), \( \varepsilon_j(m,n) \sim \mathcal{N}(0, \gamma_j^{-\infty}) \) is zero-mean Gaussian noise with \( \gamma_j^{-1} \) being the standard deviation, i.e.

\[
h_j(m,n) \sim \mathcal{N}(u_j(\cdot, m)^\top v(\cdot, n), \gamma_j^{-1}). \quad (4.2.3)
\]

### 4.2.4 Bayesian Lasso

Matrix factorization methods typically allow the observed vectors to be reconstructed by some linear combinations of the basis vectors. However, for our case, as the elements of the basis vectors correspond to different labels, addition of “labels” is generally an undefined operation under the matrix factorization setting. We adopt the Bayesian Lasso (Least Absolute Shrinkage and Selection Operator) [58] to avoid such addition as far as possible.

It has been pointed out in [76] that robust Lasso regression in the Bayesian setting can be achieved by assuming the coefficients \( v(\cdot, n) \) to follow
\[ p(v(\cdot, n)|\sigma^2) = \prod_{i=1}^{k} \frac{\lambda}{2\sqrt{\sigma^2}} \exp\left(-\frac{\lambda}{\sqrt{\sigma^2}}|v(\cdot, n_i)|\right). \]

We adopt this to achieve a sparse structure of the coefficient matrix. According to [76, 58], this conditional Laplace prior is equivalent to integrating out a hierarchical model for \( v_n \) with additional variables \( \tau_1^2, \tau_2^2, \ldots, \tau_k^2 \), where \( v_n \) is modeled as a zero-mean Gaussian with a diagonal covariance matrix scaled by \( \sigma \) and \( \tau \), given as

\[ p(v(\cdot, n)|\sigma^2, \tau_1^2, \tau_2^2, \ldots, \tau_k^2) \sim \mathcal{N}(0, \sigma^2 D) \]

where

\[ D = \text{diag}(\tau_1^2, \tau_2^2, \ldots, \tau_k^2) \]

\[ \tau_1^2, \tau_2^2, \ldots, \tau_k^2 \sim \prod_{j=1}^{p} \frac{\lambda^2}{2} e^{-\lambda^2 \tau_j^2/2} d\tau_j, \quad \tau_1^2, \tau_2^2, \ldots, \tau_k^2 > 0 \]

\[ \sigma^2 \sim \frac{1}{\sigma d\sigma^2} \]

with \( \tau_1^2, \tau_2^2, \ldots, \tau_k^2 \) and \( \sigma^2 \) assumed independent among them.

For modeling \( U_j(\cdot, m) \), \( \gamma \) and \( \sigma_j^{Li} \) for each of the \( d \) dimensions, we simply use the exponential conjugate families, given as:

\[ p(u_j(\cdot, m)|\mu_j, \psi_j) = \mathcal{N}(u_j(\cdot, m); \mu_j, \psi_j) \]

\[ p(\mu_j, \psi_j) = \mathcal{N}(\mu_j|m_j, |\mu|, \kappa|, |\nu|, |T|) \]

\[ = \mathcal{N}(\mu_j|\mu_{0j}, (\kappa_j\psi_j)^{-1}) \mathcal{W}_{|\nu|}(\psi_j|T_j) \]

\[ p(\sigma_j^{Li}|\alpha_j^{Li}, \beta_j^{Li}) = \Gamma(\alpha_j^{Li}, \beta_j^{Li}) \propto \gamma_j^{\alpha_j^{Li}-1} \exp(-\frac{\sigma_j^{Li}}{\beta_j^{Li}}) \]

\[ p(\gamma_j|\alpha_j, \beta_j) = \Gamma(\alpha_j, \beta_j) \propto \gamma_j^{\alpha_j-1} \exp(-\frac{\gamma_j}{\beta_j}). \]

where

\[ \mu_j = \frac{\kappa \mu_{0j} + m \bar{u}_j}{\kappa + m} \]

\[ \bar{u}_j = \frac{1}{M} \sum_m u_{jm} \]

\[ T_j = T_{j0} + Z_j + \frac{\kappa m}{\kappa + m} (\mu_{0j} - \bar{u}_j)(\mu_{0j} - \bar{u}_j)^T \]

\[ Z_j = \sum_{i=1}^{n} (x_{ji} - \bar{x}_j)(x_{ji} - \bar{x}_j)^T \]

\[ \nu_j = \nu_{j0} + m \]

\[ \kappa_j = \kappa_{j0} + m. \]
Figure 4.2 shows the graphical model for the proposed matrix factorization.

4.2.5 Inference

We infer the model using Gibbs sampling. To update the model parameters \(\{u_j(\cdot, m), v(\cdot, n)\}\), the hyper-parameters \(\{\mu_j, \psi_j, \sigma^2, \frac{1}{\tau^2}, \sigma^{L_i}_j, \gamma_j\}\) and the latent variables \(\{h_j(m, n)\}\) and \(\{g_j(m, n)\}\) on each of the \(d\) dimensions, we make reference to [55, 58, 11]. The corresponding conditional probabilities for the sampling steps are listed as follows:

For sampling \(u_j(\cdot, m), v(\cdot, n)\),

\[
\begin{align*}
  u_j(\cdot, m) &\sim \mathcal{N}((\psi_j \mu_j + \gamma_j \sum h_j(m, n)v(\cdot, n)), \Sigma_{m_j}) \quad (4.2.4) \\
  \Sigma_{m_j} &= (\psi_j + \gamma_j \sum v(\cdot, n)v(\cdot, n)^\top)^{-1} \\
  v_n &\sim \mathcal{N}(B^{-1}U^\top H, \sigma^2 B^{-1}) \quad (4.2.5) \\
  B &= U^\top U + D^{-1}
\end{align*}
\]

For sampling \(\mu_j, \psi_j, \sigma^2, \frac{1}{\tau^2}, \sigma^{L_i}_j, \gamma_j\),

\[
\begin{align*}
  \mu_j, \psi_j &\sim \mathcal{N}(\mu_j, (\kappa_j \psi_j)^{-1}) Wi_\nu_j(\psi_j | T_j) \quad (4.2.6) \\
  \sigma^2 &\sim \Gamma^{-1}(\frac{n-1}{2} + k, \frac{1}{2} (H - Uv_n)^\top (H - Uv_n) + \frac{v_n^\top D^{-1}v_n}{2}) \quad (4.2.7) \\
  \frac{1}{\tau^2} &\sim \mathcal{N}^{-1}(\mu', \lambda') \quad \mu' = \sqrt{\frac{\lambda^2 \sigma^2}{v_n^2}} \quad \lambda' = \lambda^2 \\
  \gamma_j &\sim \Gamma(\alpha_j, \beta_j) \quad (4.2.9) \\
  \sigma^{L_i}_j &\sim \Gamma(\alpha^{L_i}_j, \beta^{L_i}_j)
\end{align*}
\]

For sampling \(h, g\) and \(r^1\), we first sample \(g\) from \(p(g|r, \chi, \gamma)\) where \(\chi = u_j(\cdot, m)^\top v(\cdot, n)\). In particular, \(p(g|r, \chi, \gamma)\) is derived based on (4.2.1) and (4.2.2),

\[
\begin{align*}
p(g|r_i, \chi, \gamma) = \frac{p(r_i|g)p(g|\chi, \gamma)}{p(r_i|\chi, \gamma)} &\propto p(r_i|g)p(f|\chi, \gamma) \\
&= \frac{\mathcal{N}(\cdot; \mu^\ell_i, \sigma^\ell_i)}{\sum_i \mathcal{N}(\cdot; \mu^\ell_i, \sigma^\ell_i)} \mathcal{N}(\chi, \infty + \gamma^{-\infty}).
\end{align*}
\]

\(^1\)In the sequel, we denote \(h_j(m, n), g_j(m, n)\) and \(r^1_i\) as \(h, g\) and \(r_i\) respectively for the sake of presentation clarity.
Then, with reference to (4.2.2) and (4.2.3), we sample $h$ from $p(h|g, \chi, \gamma)$, given as

$$p(h|g, \chi, \gamma) \propto p(g|h)p(h|\chi, \gamma)$$

$$= \mathcal{N}(h, 1)\mathcal{N}(\chi, \gamma^{-1})$$

$$\propto \mathcal{N}(\frac{g + \gamma \chi}{1 + \gamma}, (1 + \gamma)^{-1}).$$  \hspace{1cm} (4.2.10)

The pseudo code of the Gibbs sampler of our model is given in Algorithm 3.

### 4.2.6 Folding up the $d$ dimensions

After the model inference step, we fold $D^{unfold}$ back such that each element is represented by a $d$-dimensional vector. The predicted value of $r(m, n)$ is the label that gives the highest probability of generating $g$ in different dimensions of the projected space, given as

$$r(m, n) = \arg \max_{r(m, n)} \prod_d p(r_d(m, n)|f_d(m, n)).$$

### 4.3 Experimental results

In this section, we present the experimental results on evaluating the performance of the proposed methodology using a synthetic data set and a publicly available smart home data set. We also illustrate how it can be used to discover the daily activity routines.

#### 4.3.1 Evaluation of daily activity routine discovery

We apply our proposed nominal matrix factorization method to extract daily routines as explained in Chapter 4.2. Since we do not have ground-truth information for daily activity routines, we first made use of a synthetic data set specially created with ground-truth labels for evaluating the effectiveness of the proposed methods. Then, the four benchmark data sets were further tested.
Algorithm 3: Gibbs sampling

Input: $U, V, \mu, \psi, \sigma, D, \gamma$, non-empty $r_{mn}$

repeat

for items $m = 1, \ldots, M$ do

sample $h_{mn}|r_{mn}, u_m, v_n, \gamma$ (4.2.10);

sample $u_m|h_{mn}, v_n, \gamma, \mu, \psi$ (4.2.4);

end

for items $n = 1, \ldots, N$ do

sample $h_{mn}|r_{mn}, u_m, v_n, \gamma$ (4.2.10);

sample $v_n|h_{mn}, u_m, \gamma, \sigma$ (4.2.5);

end

sample $\mu|U, \psi$ and $\psi|U$ (4.2.6);

sample $\frac{1}{\tau^2}|V, \sigma^2$ and $\sigma^2|\tau V$ (4.2.7), (4.2.8);

sample $\gamma|H, U, V$ (4.2.9);

until sufficient samples have been taken;

Evaluation metrics

Mean absolute error (MAE) is used to measure the percentage of incorrectly recovered labels with reference to the ground truth labels. For synthetic data, both the ground-truth matrix and the ground-truth basis vectors are known. The mean absolute error of recovering $D$ (data reconstruction MAE) is defined as,

$$MAE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} Dist(\hat{r}_{ij}, r_{ij})$$

where $Dist(\cdot)$ is a distance function defined as $Dist(\hat{r}_{ij}, r_{ij}) = 0$ if $\hat{r}_{ij} = r_{ij}$ or 1 otherwise. Similarly, we can also measure the basis vector reconstruction MAE. We argue that even if the data reconstruction MAE is small, it does not necessarily mean that the latent basis vectors can be discovered. And, if a low MAE for the basis vectors can be obtained with respect to the ground truth, we consider the discovery to be accurate for human interpretation.
Performance evaluation

We implemented the proposed method and four related matrix factorization methods for performance comparison, including

- Ternary Matrix Factorization (TMF) [50]: Nominal matrix is firstly transformed into a binary matrix by mapping each label to a binary vector. The binary matrix is then recursively approximated where hard constraints are imposed to make the low rank matrices binary.

- Bayesian Non-negative Matrix Factorization (P_NMF_order) [70]: Labels are ranked based on the node2vec \( d = 1 \) results and then replaced in order by integers.

- Probabilistic Ordinal Matrix Factorization (POMF) [55]: The labels are embedded using node2vec \( d = 1 \) to both positive and negative regions on the real number line, with the prior on \( v_n \) similar to that of \( u_m \).

- Probabilistic Non-negative Ordinal Matrix Factorization with Lasso (+POMF_Lasso): The labels are embedded to only the positive real number line \( d = 1 \) and Lasso is applied on \( v_n \).

- Probabilistic Nominal Matrix Factorization (PNoMF): This is the method proposed in this paper where the labels are embedded to only the positive side for all the \( d \) dimensions, and the Lasso is applied on \( v_n \). Also, the noise model for labels is considered.

Results on synthetic data

We generate synthetic data sets with eight labels \{A, B, C, D, E, F, G, H\}. We further group the labels into subsets \{\{A, B\}, \{C, D\}, \{E, F\}, \{G, H\}\}, where labels within the same subset are supposed to be close to each other. Proximity information of the labels can be expressed by a weighted adjacency matrix as shown in Figure 4.3. We then randomly pick one label from each subset and use these four
labels to generate four ground-truth basis vectors. Repeating those basis vectors gives the noise free data matrix. For evaluation, we generate three noise free data matrices ($SD_1, SD_2, SD_3$) with 384, 576, and 768 elements from basis vectors of dimension 12, 18, and 24 respectively. We set the number of basis vectors $K$ to 4 and check if the four ground truth basis vectors can be recovered. Also, we add noise to the data matrices at different levels by randomly replacing labels based on adjacency matrix to test their robustness to noise.

Figure 4.5 and 4.6 shows the experimental results based on SD3. $PNoMF(2D)$ outperforms significantly all the state-of-art factorization methods at all noise levels in terms of MAE for basis vector discovery while keeping a good performance in data reconstruction. For all the other factorization methods, even though the data reconstruction MAE is low, the basis vector reconstruction MAE is large. That is the discovered latent factors are inaccurate. Similar results are obtained for SD1 and SD2.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.00</td>
<td>0.90</td>
<td>0.15</td>
<td>0.10</td>
<td>0.15</td>
<td>0.10</td>
<td>0.10</td>
<td>0.01</td>
</tr>
<tr>
<td>B</td>
<td>0.90</td>
<td>1.00</td>
<td>0.20</td>
<td>0.15</td>
<td>0.20</td>
<td>0.15</td>
<td>0.15</td>
<td>0.10</td>
</tr>
<tr>
<td>C</td>
<td>0.15</td>
<td>0.20</td>
<td>1.00</td>
<td>0.90</td>
<td>0.15</td>
<td>0.10</td>
<td>0.20</td>
<td>0.10</td>
</tr>
<tr>
<td>D</td>
<td>0.10</td>
<td>0.15</td>
<td>0.90</td>
<td>1.00</td>
<td>0.10</td>
<td>0.01</td>
<td>0.15</td>
<td>0.10</td>
</tr>
<tr>
<td>E</td>
<td>0.15</td>
<td>0.20</td>
<td>0.15</td>
<td>0.10</td>
<td>1.00</td>
<td>0.90</td>
<td>0.20</td>
<td>0.15</td>
</tr>
<tr>
<td>F</td>
<td>0.10</td>
<td>0.15</td>
<td>0.10</td>
<td>0.01</td>
<td>0.90</td>
<td>1.00</td>
<td>0.15</td>
<td>0.10</td>
</tr>
<tr>
<td>G</td>
<td>0.10</td>
<td>0.15</td>
<td>0.20</td>
<td>0.15</td>
<td>0.20</td>
<td>0.15</td>
<td>1.00</td>
<td>0.90</td>
</tr>
<tr>
<td>H</td>
<td>0.01</td>
<td>0.10</td>
<td>0.15</td>
<td>0.10</td>
<td>0.15</td>
<td>0.10</td>
<td>0.90</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Figure 4.3: Weighted adjacency matrix of the labels for the synthetic data set.

Results on smart home data

We apply $PNoMF$ to two real indoor trajectory data sets. In our experiment, we set the number of basis vectors as 6 and 4 for dataset 1 and dataset 2 respectively according to Figures 4.7 and 4.8. For dataset 1, we also split it into 5 different
Figure 4.4: Visualization of label embedding of synthetic data.

Figure 4.5: Basis vector reconstruction MAE of different algorithms at different noise levels based on SD3.
Figure 4.6: Data reconstruction MAE of different algorithms at different noise levels based on SD3.

subsets, covering the sensor readings collected in Dec 2010, Jan 2011, Feb 2011, Mar 2011, and Apr 2011 for daily routine analysis per month.

As the ground-truth basis vectors are not available, we can only compare the data reconstruction MAE. According to Tables 4.1 and 4.2, PNoMF gives the best or close to the best result as compared to the other methods for data reconstruction. Also, the use of label embedding of a higher dimensional space will lower the MAE which is consistent to our understanding explained earlier. As PNoMF is designed to find highly interpretable basis vectors without sacrificing much the matrix reconstruction accuracy, Figures 4.9 to 4.18 show the daily routines and the corresponding coefficient matrices obtained based on the Dec, Jan, Feb, March and April data subsets. To ease our interpretation of the daily activity routines obtained for the five months, we put a red asterisk to the coefficient of the highest value for each day per month. Each corresponds to the most representative daily routine for the particular day. For example, by referring to Figure 4.9, the routine #5 can be interpreted as having a day with eating and relaxing for most of the time with some working hours at around 4-6pm. And according to Figure 4.10, this routine happens
for most of the days in Dec 2010 except for some particular days where they follow other basis vectors with more time being spent in the kitchen at different periods of time. Also, we compare the daily routines obtained for different months and found that they are quite different. For example, there is quite a long period of working time in the afternoon for the routine #5 of Dec. 2010 as compared to a much shorter one around noon for the routine #2 of Mar. 2011. Both are the most representative daily routines for the corresponding months. As shown in Figure 4.19 and Figure 4.20, the routine #2 happens for most of the days in dataset 2, which indicates that the resident wakes up at around 9am, relaxes for most of the time during the day and have a tea break at around 3pm. As discussed in [42], allowing the elderly to be able to discuss their own daily routines and exceptions can help them reflect more their daily activities, which in turn may provide better support to their awareness on potential deterioration of their functional capabilities. We believe the methodology developed in this chapter can provide more information to the elderly for more in-depth reflection in general.

![Figure 4.7: Performance of PNoMF on dataset 1 with different numbers of basis vectors.](image)

<table>
<thead>
<tr>
<th>Dataset 1</th>
<th>PNoMF (1D)</th>
<th>PNoMF (2D)</th>
<th>PNoMF (3D)</th>
<th>POMF</th>
<th>+POMF_lasso</th>
<th>P_NMF_order</th>
<th>TMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.429(0.004)</td>
<td>0.359(0.011)</td>
<td>0.304(0.005)</td>
<td>0.462(0.002)</td>
<td>0.548(0.006)</td>
<td>0.635(0.011)</td>
<td>0.307(0.002)</td>
<td></td>
</tr>
<tr>
<td>Dataset 2</td>
<td>0.580(0.006)</td>
<td>0.525(0.015)</td>
<td>0.451(0.005)</td>
<td>0.802(0.010)</td>
<td>0.794(0.009)</td>
<td>0.877(0.003)</td>
<td>0.419(0.001)</td>
</tr>
</tbody>
</table>

Table 4.1: Mean MAE(±SD) of smart home data.
Figure 4.8: Performance of PNoMF on dataset 2 with different numbers of basis vectors.
<table>
<thead>
<tr>
<th>Month</th>
<th>PNoMF (1D)</th>
<th>PNoMF (2D)</th>
<th>PNoMF (3D)</th>
<th>POMF</th>
<th>+POMF_Lasso</th>
<th>P_NMF_order</th>
<th>TMF</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC 2010</td>
<td>0.500(0.002)</td>
<td>0.201(0.001)</td>
<td>0.235(0.013)</td>
<td>0.458(0.001)</td>
<td>0.517(0.030)</td>
<td>0.838(0.006)</td>
<td>0.179(0.001)</td>
</tr>
<tr>
<td>JAN 2011</td>
<td>0.252(0.006)</td>
<td>0.118(0.001)</td>
<td>0.111(0.029)</td>
<td>0.189(0.047)</td>
<td>0.223(0.030)</td>
<td>0.851(0.001)</td>
<td>0.104(0.001)</td>
</tr>
<tr>
<td>FEB 2011</td>
<td>0.291(0.004)</td>
<td>0.133(0.012)</td>
<td>0.109(0.002)</td>
<td>0.242(0.006)</td>
<td>0.375(0.001)</td>
<td>0.314(0.003)</td>
<td>0.099(0.003)</td>
</tr>
<tr>
<td>MAR 2011</td>
<td>0.406(0.002)</td>
<td>0.270(0.009)</td>
<td>0.262(0.018)</td>
<td>0.377(0.002)</td>
<td>0.415(0.002)</td>
<td>0.508(0.001)</td>
<td>0.277(0.008)</td>
</tr>
<tr>
<td>APR 2011</td>
<td>0.299(0.007)</td>
<td>0.167(0.001)</td>
<td>0.161(0.008)</td>
<td>0.331(0.000)</td>
<td>0.351(0.003)</td>
<td>0.598(0.008)</td>
<td>0.166(0.001)</td>
</tr>
</tbody>
</table>

Table 4.2: Mean MAE(±SD) of subsets of dataset 1.
4.3.2 Applicability to other data sets

We compare the basis vector obtained by our proposed method with [50], using the same voting records data set from UCI Machine Learning Repository. The data set contains voting records from the U.S House of Representatives. 435 representatives
from Republican Party (GOP) and Democratic Party (D) provide their opinions on 16 political issues by voting “yea”, “nay” or “unknown”. Notice that the “unknown” opinion here does not indicate a missing value. We set these three labels to be not similar with each other, and project the label based on their proximity onto a two dimensional space. According to Figure 4.22, PNoMF(2D), PNoMF(3D) and TMF have very similar performance in terms of data reconstruction MAE. Here we cannot test the basis vector reconstruction MAE as the ground truth is not available. Still,
we try to evaluate the basis vectors obtained by different methods qualitatively.

Figure 4.21 shows the hamming distance of the voting records among the representatives, with the first 168 representatives belonging to Republican Party and the remaining 267 belonging to Democratic Party. It is easy to observe that the voting records of representative from the two parties are not always distinct, i.e., some representatives of a party may have similar voting patterns to those of representatives
from the other party, and vice versa. We set $K = 4$ and presented the obtained basis vectors in Figures 4.24 and 4.25. The first ($k1$) and the last ($k4$) basis vectors are very similar to the ones discovered in [50]. We consider them as the party-specific voting patterns of GOP and D. The second voting pattern ($k2$) describes the voting pattern that votes for “unknown” in almost all 16 issues. The third voting pattern ($k3$) may indicates that some representatives remain neutral for some issues, which is different from the voting pattern of their own parties. Such internal disputes can be shown in Figure 4.23, especially for issue 2 (handicapped-infants) and issue 10 (mx-missile). Then, given the set of basis vectors, a representative from GOP with
the coefficients [0.72, 0.03, 0.24, 0.03] for the four basis vectors indicates that he votes like a typical GOP representative \((k1)\), but may sometimes vote for another side \((k3)\).

*Figure 4.21: Hamming distance matrix of the 435 representatives.*
Figure 4.22: Data reconstruction MAE.
Figure 4.23: Distributions of voting records of two parties in 16 political issues.
<table>
<thead>
<tr>
<th>TMF</th>
<th>issue 1</th>
<th>issue 2</th>
<th>issue 3</th>
<th>issue 4</th>
<th>issue 5</th>
<th>issue 6</th>
<th>issue 7</th>
<th>issue 8</th>
<th>issue 9</th>
<th>issue 10</th>
<th>issue 11</th>
<th>issue 12</th>
<th>issue 13</th>
<th>issue 14</th>
<th>issue 15</th>
<th>issue 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
</tr>
<tr>
<td>$k_2$</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
</tr>
<tr>
<td>$k_3$</td>
<td>nay</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
</tr>
<tr>
<td>$k_4$</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>nay</td>
<td>unknown</td>
<td>yea</td>
</tr>
</tbody>
</table>

Figure 4.24: Voting patterns $U$ obtained by TMF.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td></td>
</tr>
<tr>
<td>$k_2$</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>yea</td>
<td>unknown</td>
<td>nay</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td></td>
</tr>
<tr>
<td>$k_3$</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td></td>
</tr>
<tr>
<td>$k_4$</td>
<td>yea</td>
<td>nay</td>
<td>yea</td>
<td>nay</td>
<td>nay</td>
<td>nay</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td>yea</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.25: Voting patterns $U$ obtained by PNoMF.
Another nominal data that we can find in UCI Machine learning Repository for our testing is the lenses data set. The data set contains 24 patients and each patient has 4 categorical attributes, namely age, spectacle prescription, astigmatic, and tear production rate. Each attribute has its own set of labels. There are altogether 9 labels. By allowing the possible labels of each attribute to be close and the labels not corresponding to the same attribute farther apart, a block-diagonal matrix is used for representing the adjacency matrix. As there are three classes in the data set, we thus set $K = 3$, and then compare different methods in terms of data reconstruction MAE. The result is shown in Table 4.3 where PNoMF(3D) gives the best result, followed by PNoMF(1D), PNoMF(2D), and then TMF.

4.4 Summary

In this chapter, a probabilistic framework with Bayesian Lasso is proposed for nominal data matrix factorization with application to smart home data mining. As shown in our experimental results using both synthetic and real smart home data sets, the basis vectors obtained by our proposed methodology achieve a lower data reconstruction error and at the same time a much lower basis vector reconstruction error as compared to a number of the state-of-the-art matrix factorization methods, making direct interpretation of the basis vectors as daily activity routines possible. We believe that presenting the results to the senior adults living in the smart home will be beneficial to them to support their awareness of their daily activities, and potential degradation.
Chapter 5

Discovering shift-invariant daily routine patterns using Markov Jump Process

5.1 Introduction

The way how a person performs daily activities within a day can varies due to different factors which can be social [9] and environmental [29]. The occurrence of the activities may “shift” and their durations obviously will vary. The consequence is that even though a person follows a certain daily routine, the specific start time and duration of different daily activities involved do vary as illustrated in Figure 5.8 which shows the plot of a raw smart home data. Without carefully taking these variations into consideration, applying the matrix factorization methods directly, as adopted in a number of related work, will end up with incorrect daily routines to be extracted. The key reason is that most matrix factorization methods assume that each column is assumed to be corresponding to the same feature. For the application of daily routine discovery, this is essentially expecting that the same activity of a daily routine happens at more or less the same time on different days. It is an unrealistic assumption.
In this chapter, we first adopt the probabilistic nominal matrix factorization (PNoMF) framework proposed in Chapter 4 to extract daily routines by factorizing a sensor data matrix. We then extend it by introducing a Markov jump process (MJP) as the prior of the daily routines. MJP is a particular continuous version of discrete-time Markov chain where the transitions from one activity to another are modeled as the transitions among the states of the MJP. The time interval for each state transition is also modeled. The use of the MJP allows us to model the variations of activity durations so that shift-invariant daily routines can be extracted. Also, with the objective of improving the interpretability of the basis vectors, we are inspired by the prototype-based Bayesian Case Model [38] and link up PNoMF and MJP by obtaining the most probable data instance w.r.t. an MJP as the basis vectors. We evaluate the proposed method using a synthetic data set and two publicly available smart home data sets. In our experiments, we demonstrate that the proposed method (PNoMF-MJP) can discover the ground truth basis vectors with smaller mean absolute errors (MAE) compared with the other state-of-the-art algorithms. Also the proposed PNoMF-MJP can predict the future activity based on the user’s historical activities with higher accuracy.

In the followings, we first present the basic background of Markov jump process in Section 5.2. In Section 5.3, we illustrate how a MJP can be incorporated as the prior based on a Bayesian nominal matrix factorization framework. The experimental results on both synthetic and real data are discussed in Section 5.4. Section 5.5 provides the concluding remark.

5.2 Markov jump process (MJP)

A family \( S = \{S(t) : t \geq 0\} \) of random variables \( S(t) : \Omega \rightarrow S \) is called a *continuous-time stochastic process* on the state space \( \Sigma \), where \( S \) is the sample space. If \( S(t) = y \), the process is said to be in state \( y \) at time \( t \). For some given \( s \in \Omega \), the \( S \) valued set \( \{X(t, s) : t \geq 0\} \) is called a *path* of the stochastic process \( S \) associated with \( s \).

A continuous-time stochastic process \( \{S(t) : t \geq 0\} \) is called a *Markov process*, if
the state space $S$ is countable, and for any $t_{i+1} > t_i > \ldots > t_0$ the Markov property

$$P[S(t_{i+1}) \in S|S(t_i), \ldots, S(t_0)] = P[S(t_{i+1}) \in S|S(t_i)]$$  \hspace{1cm} (5.2.1)$$

holds. If the right hand side of Eq.5.2.1 only depends on the time difference $t_{i+1} - t_i$, but not on $t_i$, the Markov process is called *homogeneous*. Given a homogeneous Markov process, the function $p : \mathbb{R}^+ \times S \times S \rightarrow \mathbb{R}^+$ defined by

$$p(t, x, y) = P[S(t) = y|S(0) = x]$$

is called the *stochastic transition function* which describes the transition probability of moving from $x$ to $y$ within time $t$. The probability distribution

$$\pi_0(x) = P[S(0) = x]$$

is called the *initial distribution*. If there is a single $x \in S$ such that $\pi_0(x) = 1$, then $x$ is called the initial state.

In the following, we focus on homogeneous Markov process. We also assume that $p(0, x, y) = \delta_{xy}$, where $\delta_{xy} = 1$, if $x = y$ and zero otherwise. This guarantees that no transition can take place at zero time. Also the transition probabilities are assumed continuous at $t = 0$: $\lim_{t \to 0^+} p(t, x, y) = \delta_{xy}$ for every $x, y \in S$. It guarantees that the paths of $S(t) : t \geq 0$ are right continuous functions. As the state space is discrete, for an $s \in S$ and $t \geq 0$, there exists $\Delta t > 0$ so that

$$S(t + \Delta t, s) = S(t, s).$$

This fact motivates the name Markov jump process (MJP).

Assume that $\{S(t) = s \ \forall t \in (t_1, t_2)|S(t_1) = s\}$ describes an MJP in state $s$ at time $t_1$ will remain in state $s$ over the interval $(t_1, t_2]$. We denote such probability by $p(s, t_1, t_2)$, so that

$$p(s, t_1, t_2) = P(S(t) = s \ \forall t \in (t_1, t_2]|S(t_1) = s).$$

Denoting $\tau(t_1)$ as the time of the next jump after $t_1$, we also have

$$p(s, t_1, t_2) = P(\tau(t_1) > t_2|S(t_1) = s), \ t_2 > t_1.$$
$p(s, t_1, t_2)$ is monotonically non-increasing in $t_2$ and satisfies $\lim_{h \to 0^+} p(s, t, t + h) = 1$ as well as

$$p(s, t_1, t_2) = p(s, t_1, \tilde{t})p(s, \tilde{t}, t_2) \text{ for } t_1 < \tilde{t} < t_2.$$  

Consequently, $p$ has the representation

$$p(s, t_1, t_2) = \exp(-\int_{t_1}^{t_2} q(s, u)du)$$

for a nonnegative function $q(s, u)$. We can treat the state of the MJP evolving with time, and $q(s, u)$ is corresponding to the rate of leaving state $s$ at time $u$.

The Markov jump process will enter a new state after leaving the current state. As shown in [21], such a probability is given as

$$P(u < \tau(t_1) \leq v, S(\tau(t_1)) \in \Sigma | S(t_1) = s) = \exp(-\int_u^v q(s, \omega)d\omega)\left(\int_u^v \exp(-\int_u^t q(s, \omega)d\omega)q(s, t)P_t(s, \tilde{\Sigma})dt\right) \quad (5.2.2)$$

The first factor in Eq.5.2.2 is the probability of staying in state $s$ over the interval $(t_1, u]$. The integral in the second factor $\exp(-\int_u^v q(s, \omega)d\omega)q(s, t)$ is the probability for the first jump after time $u$ to occur at time $t$. Specifically, $\exp(-\int_u^v q(s, \omega)d\omega)$ describes the probability of staying in state $s$ over the interval $(u, t]$, while $q(s, t)$ models the waiting time until the jump. The distribution $P_t(s, \tilde{\Sigma})$ represents the probability of jumping from state $s$ into some measurable set $\tilde{\Sigma}$, conditioned on the jump time at $t$. Finally, we integrate over all possible values of $t$ in $(u, v]$.

When the state space $S$ of a Markov jump process is countable, it is common to map it to the space of natural numbers, i.e. $S = \{1, 2, \ldots, N\}$. We can then define a rate matrix $A$ as follows

$$A_{ss'} = \begin{cases} 
-q(s, t) & \text{for } s = s' \\
q(s, t)p(s, s') & \text{for } s \neq s'; s, s' \in S.
\end{cases} \quad (5.2.3)$$

It is an $N$-by-$N$ matrix with non-negative off diagonal elements, $q(s, t)$ is the leaving rate of state $s$ and $p(s, s')$ is the transition probability of $s$ and $s'$. $A_{ss'}$ can be interpreted as the rate of transiting from state $s$ to state $s'$. The diagonal entry of $A$ is defined as:

$$A_{ss} = -\sum_{s \neq s'} A_{ss'}$$
Thus, each row in the rate matrix sum to 0. $|A_{ss}|$ characterizes the total rate of leaving state $s$, where $|x|$ refers to the absolute value of $x$. The Markov jump process can be fully characterized by the rate matrix $A$ and the initial distribution over states $\pi_0$.

We further restrict the MJP on a finite time interval $[t_{\text{start}}, t_{\text{end}}]$. Assume that a particular path of the MJP jumps from one state to another at the ordered times $(t_1, \ldots, t_n; t_1 < t_2 < \cdots < t_{n-1} < t_n)$. We define $T \equiv (t_0, t_1, \ldots, t_n, t_{n+1})$ the set of time stamps where $t_0 = t_{\text{start}}$ and $t_{n+1} = t_{\text{end}}$. $S$ is the corresponding sequence of states, i.e. $S = (s_0, s_1, \ldots, s_n, s_{n+1})$ where $s_i = S(t_i)$. Given the rate matrix and the initial distribution we can sample $(S, T)$ by Gillespie’s algorithm [22] (Algorithm 4).

Figure 5.1 visualizes pairs $(S, T)$ on a MJP path.

**ALGORITHM 4:** Gillespie’s algorithm for sampling an MJP path on the interval $(t_{\text{start}}, t_{\text{end}})$

**Input:** The rate matrix $A$ and the initial distribution over states $\pi_0$.

**Output:** An MJP path $S(t) \equiv (s_0, S, T)$.

Draw $s_0 \sim \pi_0$ to create an initial MJP. Set $t_0 = t_{\text{start}}$ and $i = 0$.

repeat

Draw $z \sim \text{exp}(|A_{ss}|)$.

Increase $i$ by 1.

Let $t_i = t_{i-1} + z$.

The MJP jumps to a new state $s_i$ at time $t_i$ with $p(s_i|s_{i-1}) \propto A_{s_{i-1}s_i}, s_i \neq s_{i-1}$

until $t_i + z > t_{\text{end}}$.
5.3 Incorporating MJP for nominal matrix factorization

We further extend the hierarchical model proposed in Chapter 4 to discover shift-invariant daily routine patterns. To make this chapter self-contained, we restate the key concepts. We embed the nominal labels in matrix $D_A$ onto a $d$-dimensional continuous vector space by assuming that the similarity information among the labels is known. We also assume that the $d$ dimensions of the vector representations of the label are independent, so we model the (soft) range of values of the $j$-th dimension of $\mu^L_i$, denoted as $r^L_i(j = 1, ..., t, j = 1, ..., d)$, using a normal distribution $\mathcal{N}(\mu^L_i, \sigma^L_i)$, where $\mu^L_i$ is the projected value of label $L_i$ on the $j$-th dimension, and $\sigma^L_i$ indicates the standard deviation of the range corresponding to $L_i$. By defining $g_j(m,n)$ as a latent variable to indicate the particular range that $D_\mu_j(m,n)$ falls into, the conditional probability of $r_j(m,n) = L_i$ given $g_j(m,n)$ is

$$p(r_j(m,n) = L_i|g_j(m,n)) = \frac{\mathcal{N}(g_j(m,n); \mu^L_i, \sigma^L_i)}{\sum_i \mathcal{N}(g_j(m,n); \mu^L_i, \sigma^L_i)}$$

where the uncertainty of $g_j(m,n)$ is modeled as

$$p(g_j(m,n)|h_j(m,n)) = \mathcal{N}(h_j(m,n), 1)$$

with $h_j(m,n)$ interpreted as the mean value of $g_j(m,n)$. We then recover the basis vectors from the matrix $h$. Specifically, $h_j(m,n)$ is modeled as a linear combination of matrices $U_j$ and $V$ with rank $K$, given as $h_j(m,n) = U_j(\cdot, m)^T V(\cdot, n) + \varepsilon_j(m,n)$ where $X(\cdot, y)$ represents the whole $y$-th column of matrix $X$, $\varepsilon_j(m,n) \sim \mathcal{N}(t, \gamma^{-\infty})$ is zero-mean Gaussian noise with $\gamma^{-1}_j$ being the standard deviation, i.e.

$$h_j(m,n) \sim \mathcal{N}(u_j(\cdot, m)^T v(\cdot, n), \gamma^{-1}_j).$$

We adopt the Bayesian Lasso (Least Absolute Shrinkage and Selection Operator) [58] on $\nu$ as follow to ensure the sparsity of the coefficient matrix, and thus the interpretability of the discovered basis vectors.

$$p(\nu(\cdot, n)|\sigma^2, \tau_1^2, \tau_2^2, ..., \tau_K^2) \sim \mathcal{N}(0, \sigma^2 D_\tau)$$
where

\[ D_\tau = \text{diag}(\tau_1^2, \tau_2^2, ..., \tau_k^2) \]

\[ \tau_1^2, \tau_2^2, ..., \tau_k^2 \sim \prod_{j=1}^{p} \frac{\lambda^2}{2} e^{-\lambda^2 \tau_j^2 / 2} d\tau_j^2, \quad \tau_1^2, \tau_2^2, ..., \tau_k^2 > 0 \]

\[ \sigma^2 \sim \frac{1}{\sigma^2} d\sigma^2 \]

with \( \tau_1^2, \tau_2^2, ..., \tau_k^2 \) and \( \sigma^2 \) assumed independent among them.

For modeling \( \gamma \) and \( \sigma_j^{L_i} \) on each of the \( d \) dimensions, we simply use the exponential conjugate families given as

\[ p(\sigma_j^{L_i} | \alpha_j^{L_i}, \beta_j^{L_i}) = \Gamma(\alpha_j^{L_i}, \beta_j^{L_i}) \propto \gamma_j \alpha_j^{L_i-1} \exp(-\frac{\sigma_j^{L_i}}{\beta_j^{L_i}}) \]

\[ p(\gamma_j | \alpha_j, \beta_j) = \Gamma(\alpha_j, \beta_j) \propto \gamma_j^{\alpha_j-1} \exp(-\frac{\gamma_j}{\beta_j}). \]

**MJP prior**

Given a particular vector in the nominal matrix \( D_{\mu} \), each column of the coefficient matrix \( v(\cdot, n) \) can be interpreted as the portion that it comes from the different basis vectors. From another perspective, we can partition instances in \( D_{\mu} \) into \( K \) clusters by sampling columns of \( V \). MJP is chosen to capture the shift-invariant patterns in each cluster. The instance in each cluster with the highest likelihood w.r.t. the corresponding MJP is chosen to form one of the basis vectors in \( U \). To learn the MJPs, we rearrange the nominal matrix \( D_A \) into another form \( D_{MJP} \) which is no longer a matrix. For each locally maximal subsequence with one label, we only keep the label and its beginning time. It will be transformed into tuples \((S_{ij}, T_{ij})\), which can be interpreted as the start time of \( j \)-th activity on the \( i \)-th day \( S_{ij} \) is \( T_{ij} \).

For example a sequence ‘aaabbb’ will be transformed to \{\((a, 1), (b, 4)\)\}. A random element \((S, T)\) generated by an MJP follows the distribution:

\[
P(S, T) = \pi_0(s_0)\left(\prod_{i=1}^{T-1} A_{s_i-1} | A_{s_i-1} \exp(-|A_{s_i-1}|(t_i - t_{i-1}))(A_{s_i-1}^{s_i})\right)
\]

\[
\exp(-|A_{s_i-1}|(t_i - t_{i-1})))
\]

\[
= \pi_0(s_0)\left(\prod_{i=1}^{T-1} A_{s_i-1}^{s_i} \exp(-\int_{t_{start}}^{t_{end}} |A_S(s)| dt)\right). \tag{5.3.4}
\]

74
The $i$th factor in the product in Eq. 5.3.4 is the probability of waiting for $t_i - t_{i-1}$ in state $s_{i-1}$ (at rate $A_{s_{i-1}}$), and then transiting to state $s_i$. The last factor is the probability of waiting for longer than $t_{|T|} - t_{|T|-1}$ in state $s_{|T|-1}$, and does not correspond to an MJP transition.

**Inference**

We infer the MJP via Markov chain Monte Carlo (MCMC) which iterates between two Gibbs sampling steps: i) Sample MJP paths given current model parameters, and ii) Update parameters given the sampled paths. For the first step, we follow the ideas in [64], which samples a new path by running the forward-filtering backward-sampling algorithm (FFBS) [8] on a random discretization of time. In every Gibbs iteration, this discretization is re-sampled conditional on the old path and the current parameters. Specifically, we sample a set of time stamps from a Poisson process with rate $\Omega + A_{ss}$, where $\Omega > \max_i(-A_{ii})$. A random discretization of time is obtained by combining the newly sampled time stamps with the transition times of the old path. A new path is sampled by running FFBS over these time stamps, with the initial distribution $\pi_0$ and Markov transition matrix $I + \frac{1}{\Omega} A$. We can view sampling an MJP path as sequentially sampling waiting time from an exponential density and new states from a discrete distribution based on the current state.

For the second step, we put independent gamma priors on the diagonal elements $|A_{ss}|$ which are the leaving rates of the states and put independent Dirichlet priors on the probabilities of transitioning from one state to another. In particular, defining $p_{s,s'} = \frac{A_{s,s'}}{|A_{ss}|}$, we let:

$|A_{ss}| \sim \text{Gamma}(\alpha_1, \alpha_2)$

$(p_{s,1}, \ldots, p_{s,s-1}, p_{s,s+1}, \ldots, p_{s,N}) \sim \text{Dirichlet}(\beta)$.

Given a path $S(t)$ the sufficient statistics required to calculate the posterior distributions are the total amount of time spent in each state and the number of transitions between each pair of states. Thus, let $n_{s,s'}$ be the number of transitions from state $s$ to $s'$, $n_s$ be the number of times the MJP leaves state $s$, and $T_s$ be
the total amount of time spent in state $s$. The posterior distributions for the Gibbs sampling are

$$|A_{ss}||S, T) \sim \text{Gamma}(\hat{\alpha}_{1,s}, \hat{\alpha}_{2,s})$$

$$(p_{s,1}, \ldots, p_{s,s-1}, p_{s,s+1}, \ldots, p_{s,N})(S, T) \sim \text{Dirichlet}(\beta + (n_{s,1}, \ldots, n_{s,s-1}, n_{s,s+1}, \ldots, n_{s,N}))$$

where $\hat{\alpha} = \alpha + n_s$, $n_s = \sum_{s' \in S} n_{s,s'}$ and $\frac{1}{\hat{\alpha}_{2,s}} = \frac{1}{\alpha_{2,s}} + \frac{1}{T_s}$. Other parameters are sampled in the same way as mentioned in Section 4.2.5.

### 5.4 Experimental results on daily routine discovering

We applied our proposed nominal matrix factorization method with Markov jump process prior to extract daily routines based on the occurrence of the identified subflows as explained in Section 5.3. Since we do not have ground-truth information for daily activity routines, we first made use of a synthetic data set specially created with ground-truth labels for evaluating the effectiveness of the proposed methods. Then, the two real trajectory data sets were further tested.

#### Performance evaluation

For performance comparison, we implemented the proposed method and four related matrix factorization methods, including:

- **Ternary Matrix Factorization (TMF) [50]:** Nominal matrix is firstly transformed into a binary matrix by mapping each label to a binary vector. Then this binary matrix is recursively approximated where hard constraints are imposed such that elements in low rank matrices should be zero or one.

- **Bayesian Non-negative Matrix Factorization (P-NMF$_{order}$) [70]:** Labels are ranked based on the node2vec ($d = 1$) results and then replaced in order by integers.
• Probabilistic Ordinal Matrix Factorization (POMF) [55]: The labels are embedded using the node2vec \((d = 1)\) to both positive and negative regions on the real number line, with the prior on \(v_n\) similar to that of \(u_m\).

• Probabilistic Nominal Matrix Factorization (PNoMF) [43]: The labels are embedded to only the positive side for all the \(d\) dimensions, and the Lasso is applied on \(v_n\). Also, the labels are allowed to be replaced by other labels based on a noise model, instead of having their corresponding range of \(f\) value truncated.

• Probabilistic Nominal Matrix Factorization with Markov jump process prior (PNoMF-MJP): The labels are embedded to only the positive side for all the \(d\) dimensions, and the Lasso is applied on \(v_n\). Also, the labels are allowed to be replaced by other labels based on a noise model, instead of having their corresponding range of \(f\) value truncated. Basis vectors are instances from the target matrix.

**Evaluation metrics**

Mean absolute error (MAE) is used to measure the percentage of incorrectly recovered labels with reference to the ground truth labels. For synthetic data, both the ground-truth matrix and the ground-truth basis vectors are known. The mean absolute error of recovering \(D\) (data reconstruction MAE) is defined as,

\[
MAE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} Dist(\hat{r}_{ij}, r_{ij})
\]

where \(Dist(\cdot)\) is a distance function defined as \(Dist(\hat{r}_{ij}, r_{ij}) = 0\) if \(\hat{r}_{ij} = r_{ij}\) or 1 otherwise. Similarly, we can also measure the basis vector reconstruction MAE.

We argue that even if the data reconstruction MAE is small, it does not necessarily mean that the latent basis vectors can be discovered. And, if a low MAE for the basis vectors can be obtained with respect to the ground truth, we consider the discovery to be accurate for human interpretation.
We argue that the discovered basis vectors with low MAE compare to ground truth is accurate for human interpretation, but it is not enough. A small error rate on predicting the test data indicates a better generalization of the obtained basis vectors. Thus, other than measure the data reconstruction MAE and basis vector reconstruction MAE, we measure the prediction error rate. In our experiments, we use 90% of the data for training the daily routine patterns and the remaining 10% for testing. For each sequence in the test data, we keep the first two locally maximal subsequence and compare different methodologies on predicting the next locally maximal subsequence. The error rate is defined as,

$$\text{Error rate} = \frac{1}{\sum_{i=1}^{\tilde{M}} \tilde{N}_i} \sum_{i=1}^{\tilde{M}} \sum_{j=1}^{\tilde{N}_i} \text{Dist}(\hat{r}_{ij}, r_{ij})$$

where \(\text{Dist}(\cdot)\) is a distance function defined as \(\text{Dist}(\hat{r}_{ij}, r_{ij}) = 0\) if \(\hat{r}_{ij} = r_{ij}\) or 1 otherwise. \(\tilde{M}\) and \(\tilde{N}_i\) are the number of sequences in \(D_{\text{test}}\) and the number of elements need to be predicted in \(i\)-th sequence. To calculate the error rate for methodologies except \(P\text{NoMF-MJP}\), we first estimate the coefficients for different basis vectors based on the given locally maximal subsequences via non-negative least square algorithm [5]. The linear combination of the basis vectors weighted by the obtained coefficients gives the reconstruction of the next locally maximal subsequence. For \(P\text{NoMF-MJP}\), the coefficients are estimated as likelihoods of MJPs given locally maximal subsequences by Eq.5.3.4. The prediction is computed by the rate matrix of the MJP with the highest likelihood value.

**Results on synthetic data**

We generate synthetic data sets for performance evaluation with sixteen dissimilar labels from \(A\) to \(P\). We further assign labels into subsets (\(\{A, B\}, \{C, D\}, \{E, F\}, \{G, H\}, \{I, J\}, \{K, L\}, \{M, N\}, \{O, P\}\)), where labels within the same subset are close to each other. To generate data with different shift levels, we first randomly pick one label from each subset and use these eight labels to generate four ground-truth basis vectors. Repeating those basis vectors gives the shift free and noise free data matrix. The shift is generated by lengthening or shortening the length of the
locally maximal subsequence of labels by different percentages of the subsequence length. To recover the basis vectors, we set the number of basis vectors $K$ to 4 to see if the four ground-truth basis vectors can be recovered. In addition, we also add noise to the data matrices at different noise levels by randomly replacing labels based on the label adjacency matrix to test their robustness to noise.

Figures 5.2 to 5.4 show the experimental results based on one set of noise free synthetic data at different shift levels. $PNoMF-MJP$ outperforms all the state-of-art factorization methods at all shift levels in terms of MAE for basis vector reconstruction and next activity prediction while keeping a good performance in data reconstruction. After that, we pick the synthetic data with $shift\_level = 10\%$ and add noise for further testing. As shown in Figures 5.5 to 5.7, $PNoMF-MJP$ also outperforms all other methods for both basis vector reconstruction and next activity prediction without sacrificing much the matrix reconstruction accuracy. We find that the shifts introduced will make quite some elements in the original nominal matrix to take other values differently due to different levels of shift, which in turn confuses the factorization task to a great extent. This causes the poor performance of most matrix factorization methods. The performance of TMF is surprisingly good for basis vector reconstruction. We believe that it is the binarization that leaves each element to be either one or zero, instead of different ranges of values as in $PNoMF$. The confusion caused by the shift is reduced. However, the corresponding basis vector reconstruction error is still much larger than that of $PNoMF-MJP$.

<table>
<thead>
<tr>
<th></th>
<th>$PNoMF-MJP$</th>
<th>$PNoMF (1D)$</th>
<th>$PNoMF (2D)$</th>
<th>POMF</th>
<th>$P_NMF_order$</th>
<th>TMF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Dataset 1</strong></td>
<td>0.676(0.072)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
</tr>
<tr>
<td><strong>Dataset 2</strong></td>
<td>0.578(0.081)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
<td>1.0(0.0)</td>
</tr>
</tbody>
</table>

Table 5.1: Error rate($\pm SD$) on activity prediction based on the smart home data sets.
Figure 5.2: Comparison of different methodologies on basis vector reconstruction MAE at different shift levels.

Figure 5.3: Comparison of different methodologies on matrix reconstruction MAE at different shift levels.
Figure 5.4: Comparison of different methodologies on activity prediction error rate at different shift levels.

Figure 5.5: Comparison of different methodologies on basis vector reconstruction MAE at different noise levels.
Figure 5.6: Comparison of different methodologies on matrix reconstruction MAE at different noise levels.

Figure 5.7: Comparison of different methodologies on activity prediction error rate at different noise levels.
Results on Smart Home Data

We then apply the proposed nominal matrix factorization with Markov jump process prior for daily routine discovery. In our experiment, we set the number of basis vectors as 6 and 4 for dataset 1 and dataset 2 respectively as explained in Section 4.3. For the real world data, the ground-truth basis vectors are not available and thus we choose to compare the MAE on data reconstruction and error rate on next activity prediction. According to Table 5.1, $PNoMF-MJP$ gives the best result on activity prediction while keeping a good performance on data reconstruction MAE as compared to the other methods. We visualize the ground-truth activities in Figure 5.8. Figures 5.9 to 5.16 show the daily routines and the corresponding MJPs obtained based on dataset 2. For example, by referring to Figure 5.10, the routine can be interpreted as having a day with leaving home and relaxing activities for most of the time with an early sleeping time at around 9:30pm. Another daily routine as visualized in Figure 5.12 can be interpreted as relaxing for the most of the time with waking up late and sleeping late. We can find that MJP provides a new abstraction for visualizing daily routines where richer information can be presented. For example, Figure 5.10 only shows two instances of one particular daily routine, it is not easy for people to explain the underlying routine. Figure 5.9 offers us a way to inspect the interactions among activities. We can interpret the routine as: After cooking breakfast (‘Cook bf’) for around 16 minutes, there is a high probability to start eating breakfast (‘Eat bf’), followed by relaxing for around 36 minutes. Such a well summarized information about the durations of different activities and the interactions among them cannot be revealed by only presenting the instances of basis vectors as what all the existing methods do. We believe that this new abstraction can better our understanding of human’s daily activity patterns.
Figure 5.8: Visualization of activities in data set 2.
Figure 5.9: Visualization of generator matrix of MJP corresponding to daily routine #1.
Figure 5.10: Visualization of instances of daily routine #1.
Figure 5.11: Visualization of generator matrix of MJP corresponding daily routine#2.
Figure 5.12: Visualization of instances of daily routine #2.
Figure 5.13: Visualization of generator matrix of MP corresponding daily routine #3.
Figure 5.14: Visualization of instances of daily routine #3.
Figure 5.15: Visualization of generator matrix of MJP corresponding daily routine #4.
5.5 Summary

In this chapter, a probabilistic framework with Markov jump process prior and Bayesian Lasso is proposed for shifted-invariant nominal data matrix factorization with application to daily routine discovery. As shown in our experimental results using both synthetic and real smart home data sets, the basis vectors obtained by our proposed methodology achieve a much lower basis vector reconstruction error and activity prediction error. At the same time, a much lower test data reconstruction error as compared to a number of the state-of-the-art matrix factorization methods can be obtained. It provides a new abstraction for visualizing daily routines. We believe that presenting the results to the senior adults living in the smart home will be beneficial to them to support their awareness of their daily activities, and potential degradation.
Chapter 6

Conclusions

6.1 Summary of thesis

In this thesis, we focus on elderly mobility and daily routine analysis using unsupervised learning methods in a smart home setting.

In Chapter 3, we presented an unsupervised learning methodology for extracting behavioral patterns as representations of human daily activities from indoor trajectory data. We make use of probabilistic finite state automaton (PDFA) with local behavioral features to model such data. A state-merging approach is adopted for inferring a compact PDFA where states with similar local behavioral features are merged during the inference. The conjecture is that an activity is characterized by a set of states (local movement patterns) where, once reached one of them, there is a high chance of transiting and staying among the states before leaving them. Behavioral patterns are then detected by partitioning the PDFA into subflows via kernel $k$-means algorithm.

In Chapter 4, we presented a principled methodology to decompose a nominal matrix for applications like discovery of daily routines in a smart home. To achieve that, we assume that the similarity of detected activities can be estimated. We embed the discrete labels onto a $d$-dimensional continuous space. Then, a hierarchical probabilistic model with Bayesian Lasso is proposed for the factorization to ensure the interpretability of the basis vectors. Gibbs sampling is adopted to carry out the
model inference.

In Chapter 5, we further extend the model proposed in Chapter 4 to discover shift-invariant daily routine patterns. Markov jump process is introduced as prior for basis vectors to model the variations of durations of activities. The model is also in Bayesian framework and can be inferred via Gibbs sampling.

6.2 Contributions

Unsupervised learning methodology for activity detection is rare, especially that the longer-range dependency of sensor triggering event is to be taken into consideration. The proposed features in Chapter 3 are specially designed and shown particularly effective to group triggering events exhibiting similar local movement behaviors. Also this is the only work that model an activity as a subflow in a flow graph.

The probabilistic nominal matrix factorization framework (PNoMF) presents in Chapter 4 is a novel extension of ordinal matrix factorization (OMF) [55] where not only the noise model of the labels and the label similarity, but also the interpretability of the resulting basis vectors are all considered in a unified graphical model to achieve robust nominal matrix factorization. Other than daily routine discovery, the proposed factorization method is also applicable to other applications as far as the label similarity information is available.

PNoMF-MJP is the first work that introduces MJP as the prior to find shift-invariant patterns in a matrix factorization framework. It also offers a new abstraction for visualizing daily routines other than simply showing the basis vectors so that, richer information can be presented. We believe it is applicable to some longitudinal pattern monitoring tasks in dynamic environments.

6.3 Future work

For our activity detection framework, it is noted that a number of thresholds are needed. One possible direction is to find a generic way to determine the values
of these thresholds, such as the optimal number of subflows for a given PDFA, the optimal value for checking the compatibility of two candidate merging states, etc. Another possible direction is to deal with the overlapping among subflows instead of assuming no overlapping as in Chapter 3. Last but not least, in our work we only consider one resident case. For the case of multiple residents, trajectory segmentation has to be carried out before the proposed techniques can be applied. Also, how to extend the methodology to model the interactions, if the activities are related to interactions of more than one individual would be worth pursuing.

For our matrix factorization framework, one possible research direction is to speed up the sampling method to improve the scalability. Another possible direction is automatic determination of the optimal number of basis vectors and the optimal dimensions of the label embedding space. In general, it will be interesting to integrate the label embedding and the matrix factorization into one unified framework.
Bibliography


CURRICULUM VITAE

Academic qualifications of the thesis author, Mr. LI Chen:

- Received the degree of Bachelor of Engineering from Jiangxi Normal University, July 2009.
- Received the degree of Master of Science from Hong Kong Baptist University, November 2011.

June 2018