

Learning with Limited Data in Sensor-based Human Behavior Recognition

by

Wenchen Zheng

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

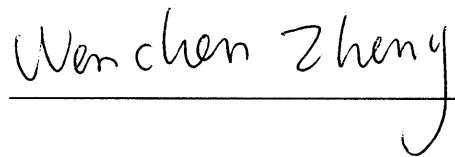
30 August, 2011, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

A handwritten signature in cursive script, reading "Wenchen Zheng", written over a horizontal line.

Wenchen Zheng

Learning with Limited Data in Sensor-based Human Behavior Recognition

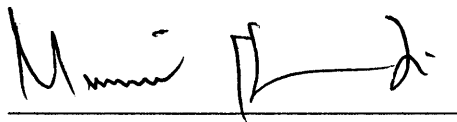
by

Wenchen Zheng

This is to certify that I have examined the above PhD thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

A handwritten signature in black ink, appearing to read 'Qiang Yang', written over a horizontal line.

Prof. Qiang Yang, Thesis Supervisor

A handwritten signature in black ink, appearing to read 'Mounir Hamdi', written over a horizontal line.

Prof. Mounir Hamdi, Head of Department

Department of Computer Science and Engineering

30 August 2011

Acknowledgment

First of all, I would like to express the most sincere gratitude to my supervisor, Prof. Qiang Yang. His wide knowledge and logical way of thinking have been of great value for me. His understanding, encouraging and intensive guidance make this dissertation become possible. I am very grateful for his consistent and generous support throughout my whole doctoral study.

I would like to thank many professors including Prof. Dik Lun Lee, Prof. Fugee Tsung, Prof. Jiannong Cao, Prof. Raymond Wong, Prof. Dit-Yan Yeung, Prof. Lei Chen and Prof. Nevin Zhang for serving as the committee members of my thesis defense, thesis proposal defense and PhD qualification exam. Their valuable comments are of great help to my research. I would especially thank two senior friends, Dr. Junfeng Pan and Dr. Dou Shen, for their generous and consistent help in giving me valuable advices on research, internship, job hunting and career planning. I also thank the members of the Artificial Intelligence and Statistics group: Rong Pan, Jialin Pan, Jie Yin, Kai Zhang, Ivor Tsang, Wei Xiang, Bin Cao, Nan Liu, Hao Hu, Qian Xu, Weike Pan, Yin Zhu, Erheng Zhong, Weizhu Chen, Si Shen, Wujun Li, Yu Zhang, Yi Zhen, Chung Lee, Tengfei Liu, Pingzhong Tang, Haodi Zhang, Ning Ding and so on, for their kind help and insightful discussions all the time.

I am grateful to the people who helped me a lot during my internships and oversea visiting. In Microsoft Research Asia, I got generous help from my mentors Dr. Xing Xie, Dr. Yu Zheng and many intern friends including Yukun Chen, Defu Lin, Kan Wu, Chun-Shuo Lin, Chengyang Zhang, Chunxin Xie, Jing Yuan, Hanqing Cui, Kai Jiang, Guwen Feng, Mohan Yang, Pengfei Qiu, Hyoseok Yoon, Dongyeop Kang, Qiang Hao, Xin Lu. In Microsoft Advertising team, I appreciate the great support from my mentor Dr. Kathy Dai, and many other team members including Dou Shen, Ying Li, Sandy Saul, Valeri Liborski, Xianfang Wang, Qing Xu, Mojdeh Jalali Heravi, Steven Hanks, Fei Cao, Deepankar Dubey, Ian Ferreira, Zhaoji Chen, Amit Shaked. I would like to thank Prof. Stephen Intille for hosting me to visit Massachusetts Institute of Technology and giving me valuable guidance. I also thank Dr. Kent Larson, Robert Marlatt, Ned Burns, Selene Mota, Jason Nawyn, Fahd Albinali, Yi Han, Dustin Smith for their help and discussion.

Last but not least, I would like to give special thanks to my parents Fuzhi Zheng and Zhenying Shi, my girlfriend Jinny Um. Their love and support is the biggest energy for me to make all this possible.

Contents

Title Page	i
Authorization Page	ii
Signature Page	iii
Acknowledgments	iv
Table of Contents	v
Abstract	xiii
1 Introduction	1
1.1 Three Problems in Sensor-based Human Behavior Recognition	2
1.1.1 Location Estimation	3
1.1.2 Activity Recognition	4
1.1.3 Mobile Recommendation	6
1.2 Sensors in Human Behavior Recognition	8
1.3 The Data Sparsity Challenges	9

1.4	Main Contribution	10
1.5	Thesis Outline	12
2	WiFi-based Location Estimation	14
2.1	Background of Learning-based Localization	15
2.1.1	Data Input and Output	15
2.1.2	Existing Learning Models	17
2.2	The Context-dependent Data Sparsity Challenge	19
2.2.1	Related Work	21
2.3	Latent Multi-Task Learning for Multi-Device Localization	22
2.3.1	Problem Formulation and Our Solution	22
2.3.2	Empirical Study	28
2.4	Transferred Hidden Markov Model for Time-varying Localization	31
2.4.1	Problem Formulation and Our Solution	32
2.4.2	Empirical Study	35
2.5	Summary	39
3	Sensor-based Activity Recognition	41
3.1	Background of Learning-based Activity Recognition	42
3.1.1	Data Input and Output	42
3.1.2	Existing Learning Models	44
3.2	The User-dependent Data Sparsity Challenge	49

3.2.1	Related Work	50
3.3	Collaborative Activity Recognition with Latent Aspect Model	51
3.3.1	Problem Formulation and Our Solution	52
3.3.2	Empirical Study	55
3.4	The Activity-dependent Data Sparsity Challenge	59
3.4.1	Related Work	60
3.5	Cross-Domain Learning for Activity Recognition	62
3.5.1	Problem Formulation and Our Solution	62
3.5.2	Empirical Study	65
3.6	Summary	71
4	GPS-based Mobile Recommendation	73
4.1	Background on Recommender Systems	74
4.1.1	Data Input and Output	74
4.1.2	Existing Models	75
4.2	Crowd-dependent Data Sparsity	80
4.2.1	Related Work	81
4.3	Collective Matrix Factorization for General Recommendation	82
4.3.1	Problem Formulation and Our Solution	83
4.4	Collective Tensor Factorization for Personalized Recommendation	85
4.4.1	Problem Formulation and Our Solution	86
4.5	Empirical Study	88

4.6	Summary	101
5	A Unified Learning Framework	104
5.1	Relationships among Location Estimation, Activity Recognition and Mobile Recommendation	104
5.2	Instantiation of The Unified Learning Framework	106
6	Conclusion and Future Work	110
6.1	Conclusion	110
6.2	Future Work	112
	Bibliography	114

List of Figures

1.1	Sensor-based behavior recognition.	1
1.2	WiFi indoor localization [1].	3
1.3	Sensors installed in a home setting for activity recognition [2].	6
1.4	Location data sharing service [3].	7
2.1	Signal variations over devices and time periods.	20
2.2	Setting for the device-dependent data sparsity challenge.	23
2.3	Experimental results for LatentMTL.	29
2.4	Convergence of <i>latentMTL</i> algorithm.	30
2.5	Setting for the time-dependent data sparsity challenge.	32
2.6	Hidden Markov Model.	32
2.7	Our TrHMM to adapt a localization model from time 0 to time t	33
2.8	Use 08:26am model to predict others.	37
2.9	Experimental results for TrHMM	38
3.1	Examples of DBN and hierarchical CRF in activity recognition.	49
3.2	An eHealth demo based on activity recognition.	50

3.3	Setting for the user-dependent activity data sparsity problem.	52
3.4	User-dependent aspect model.	53
3.5	Data statistics for collaborative activity recognition.	55
3.6	System performance for collaborative activity recognition.	56
3.7	Performance on each user.	58
3.8	An example of CDAR on the MIT PLIA1 dataset [2].	60
3.9	Extract Web data for activities.	63
4.1	A mobile recommendation system.	74
4.2	An example of location and activity recommendation by ranking.	74
4.3	Illustration of the data sparsity challenge in mobile recommendation.	81
4.4	Demonstration of our model.	83
4.5	Model illustration in a tensor/matrix form.	86
4.6	GPS user statistics.	89
4.7	GPS devices and data distribution.	89
4.8	Impact of the location features, w.r.t. λ_1	92
4.9	Impact of the activity correlations, w.r.t. λ_2	93
4.10	Impact of the location/activity types.	96
4.11	Impact of the user number.	101
4.12	Impact of model parameters.	102
5.1	Sensor-based behavior recognition revisited.	105

List of Tables

2.1	Review on previous work of location estimation.	21
3.1	Review on previous work of activity recognition.	51
3.2	Impact of the user latent factors [acc \pm std].	57
3.3	Impact of the model parameters on D_u and D_f	58
3.4	Review on previous work of activity recognition.	61
3.5	Algorithm performance on Amsterdam and Intel datasets.	67
3.6	Algorithm performance on MIT PLIA1 dataset	68
3.7	Algorithm performance with cosine and MMD Similarities.	69
4.1	Review on previous work of mobile recommendation.	82
4.2	Activities that we used in the experiments.	90
4.3	Rating criteria for locations and activities.	90
4.4	Comparisons under different p -values for $nDCG@p$	94
4.5	Impact of the stay region size.	95
4.6	Impact of the user number.	95
4.7	Comparison with baselines, by “mean \pm std”.	99

4.8	Personalized vs. General Recommendation	100
-----	---	-----

Learning with Limited Data in Sensor-based Human Behavior Recognition

Wenchen Zheng

Department of Computer Science and Engineering

Abstract

Human behavior recognition from sensor observations is an important topic in both artificial intelligence and mobile computing. It is also a difficult task as the sensor and behavior data are usually noisy and limited. In this thesis, we first introduce the three major problems in human behavior recognition, including location estimation, activity recognition and mobile recommendation. Solving these three problems helps to answer the typical questions in human behavior recognition, such as where a user is, what s/he is doing and whether s/he will be interested in doing something at somewhere. In our attempt to solve these problems, we find that in practice the biggest challenge comes from the data sparsity. Such data sparsity can be because we have limited labeled data for new contexts in localization, or limited sensor data for users / activities in activity recognition, or limited activity data for mobile recommendation. In order to address these challenges, we propose learning methods which can effectively incorporate domain-dependent auxiliary data in training and thus greatly relieve the sparsity problem. We conduct empirical studies with real-world data sets, and demonstrate the effectiveness of our algorithms over the competing baselines.

Chapter 1

Introduction

Thanks to the development of sensor technologies, the cheap and easy-to-use electronic sensors are now easy to get. Such sensors are used to perceive the ambient information about the mobile user, making the human behavior recognition become feasible. Advanced applications are gaining speed in areas such as pervasive computing, medical assistive technologies, security and environmental monitoring, gaming, local social network, and so on.

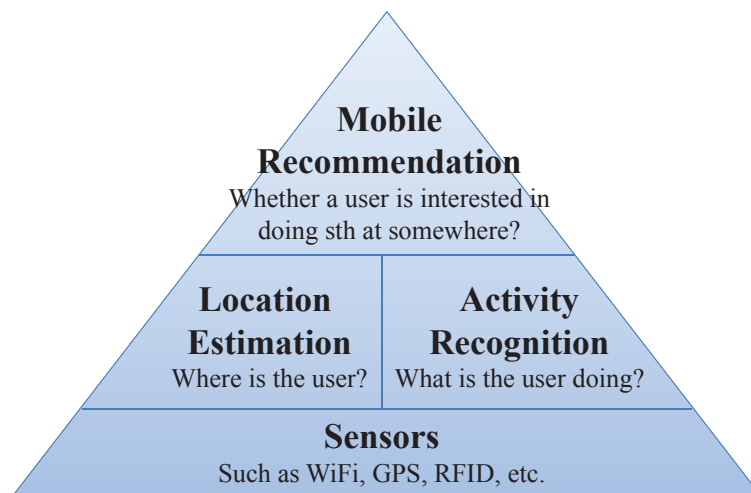


Figure 1.1: Sensor-based behavior recognition.

In this thesis, we study sensor-based human behavior recognition, which aims to answer the questions of where a user is, what s/he is doing and whether s/he is interested in doing something at somewhere from some sensor observations. As shown in Figure 1.1, for human behavior recogni-

tion we are interested three problems, including

- **Location Estimation.** Location is an important indicator about a user's behavior. In location estimation, the goal is to predict where a user is based on some sensor observations. For example, one can use WiFi received signal strengthes to achieve an localization accuracy of some meters in indoor environments [?, 4].
- **Activity Recognition.** In sensor-based activity recognition, the goal is to predict what a user is doing given some sensor observations. For example, one can use the radio frequency identification (RFID) sensors that are affixed to the kitchen objects to predict whether a user is making coffee or not [5].
- **Mobile Recommendation.** After knowing the location and activity history, it is also important to predict the users' interests (in the physical world). For example, we can predict whether a user will be interested in doing some activity at some place (e.g. dining in a restaurant near the bar street), by considering whether a location is popular, what kind of activities are popular there, and what kind of activities the user likes, etc. Such information can be obtained from the location and activity history data from all the users. Based on this, we can therefore provide useful recommendations [6].

We first briefly introduce these three problems in the following section, and then elaborate their backgrounds, challenges and our solutions in Chapters 2-4.

1.1 Three Problems in Sensor-based Human Behavior Recognition

In this thesis, we will summarize our work in human behavior recognition, which aims to predict a user's location, activity from sensor data, and further uses these pieces of information for mobile recommendation. In other words, we are interested in the following three typical problems in human behavior recognition: location estimation, activity recognition and mobile recommendation. We may not cover other important topics such as sensor system design or physio-phycological study.

1.1.1 Location Estimation

Location is one of the most important component of human behavior. In many applications, we want to know where a user is so that we can provide location-based services. The simplest one is navigation, such as Global Positioning System (GPS). Other than navigation, given such location information, we can either provide some instant location-based service like local search which integrates the current location information to for more accurate search results [7], or discover some long-term user behavior patterns. For example, Eagle and Pentland from MIT showed how to use mobile phones to record the user's movements and contacts with other people, and thus figure out their routine activity patterns and social behaviors [8]. Besides, some location-based social networks such as Foursquare¹ also emerge to provide place check-in and social gaming services.

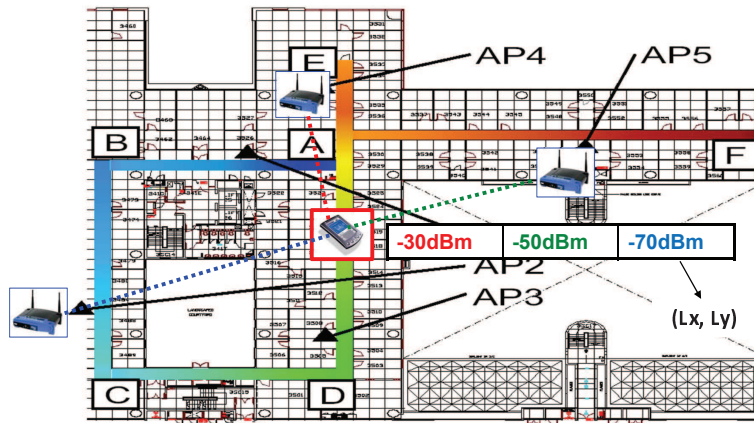


Figure 1.2: WiFi indoor localization [1].

In outdoor environment with open space, GPS is shown to work well in detecting the user's location. We will discuss how to use these GPS-based locations for mobile recommendation later. In this thesis, we also consider WiFi-based localization which can detect the user's location in the indoor environment [9, 10, 11, 12].

Example of WiFi-based Location Estimation: Figure 1.2 shows an example of indoor location estimation using WiFi signal strength. A user moving in the indoor environment carries a mobile device such as smart-phone or laptop. The mobile device can detect multiple WiFi signals from various access points. Then, the detected WiFi signal strength values are used to form a data vector. As shown in Figure 1.2, the mobile device receives a signal strength vector as

¹<http://foursquare.com>

$[-30dBm, -50dBm, -70dBm]$, where dBm is a standard signal strength measurement. Generally, the mobile device receives different signal strength vectors at different locations. As a result, given a signal-to-location mapping function, we can predict the user's location with her current signal strength vector. Such a signal-to-location mapping function is also referred as a localization model, which is capable to transform a signal vector to a location.

In general, there are two categories of approaches to get the localization model. One is using propagation model, which relies on radio propagation and trilateration/triangulation techniques for location estimation [13, 14, 15]. One typical example of propagation model is described in Bahl and Padmanabhan's paper [9]. Empirically, because the signal strength value decreases as the distance between the mobile device and the access point increases, one can model the received signal strength as a function of that distance and other factors like floor attenuation and wall attenuation. Therefore, given a received signal strength vector, one can first predict the distance of the mobile device to each access point, and thus use geometry constraints (e.g. trilateration) to figure out the location. The other is using learning model, which relies on accumulating many training data and then applying statistical learning to find out the signal-location mappings [16, 17, 18, 19]. The basic idea is to collect a set of WiFi data labeled with corresponding locations, and then train a classifier [18] or a regression function [20]. Both propagation model and learning model have their advantages and disadvantages, though according to some pilot studies [9, 4] that, the learning model tends to be more robust to the signal noises and thus generally performs better than the propagation model. In this thesis, we focus on learning model in location estimation.

1.1.2 Activity Recognition

Activity is another important component of human behavior. It tells what a user is doing under some circumstance (e.g. at some place, and/or using some objects). Before we give the definition of the activities that we are interested in, let us introduce two definitions of *world state* and *event*.

Definition 1. *World State*: *a world state describes the particular condition that someone or something of the interested world is in at a specific time.*

Definition 2. *Event*: *an event is a function that maps one world state to another.*

Notice that, in this thesis, we are interested in the events that can be detected by the existing sensors. For example, the event of moving a cabinet can be detected by using some radio frequency

identification tag. Finally, we give a definition on activity as:

Definition 3. Activity: *an activity refers to some repeatable event initiated by one or more agents such as a human.*

Some typical examples of the activities we are interested in are the Activities of Daily Living (ADLs)², which are extensively used in medical service and health care to measure the functional status of a person. For example, an MIT Technology Review article describes the scenario of using radio frequency identification tags and magnetic sensors discreetly affixed to mugs, a tea jar, a kettle and a cabinet to track each tea-making step³. In this example, the activity of “making tea” is a repeatable event and is initiated by a human user. There are some other events that are not initiated by any agent or not repeatable, such as the natural wind blowing outdoor. Therefore, they are not the activities we are interested in here. Note that, knowing user activities helps us to provide various services. For example, Pollack et al. showed how to use sensor to capture user’s activity status and further apply planning to give cognitive assistance on medication [21]. Yin et al. used WiFi to detect abnormal activities for security control [22]. Albinali et al. tried to estimate a user’s physical activities from accelerometer data so as to calculate the energy expenditure for health monitoring [23].

Note that, the activities to recognize can vary from task to task. There are several criteria we employ to define the interested activities in our activity recognition research, as shown in the following definition:

Definition 4. Activity Selection Criteria: *(i) The activities are interesting to the application. (ii) The activities are recognizable given the existing hardware and technology support.*

Activity recognition has its root in plan recognition, where a series of observed actions are used to infer the goal by first-order logic reasoning [24]. As sensors become available, recent activity recognition research focuses more on reasoning under uncertainty from the sensor observations.

Example of Sensor-based Activity Recognition: Figure 1.3 gives an example of how to use sensors in a home environment for activity recognition. In general, one may install various sensors such as RFID, motion sensor, etc. in the environment. Then, every action by the user can invoke some sensor observations. The series of sensor observations are finally used to infer the activity

²http://en.wikipedia.org/wiki/Activities_of_daily_living

³<http://www.technologyreview.com/computing/13237/>

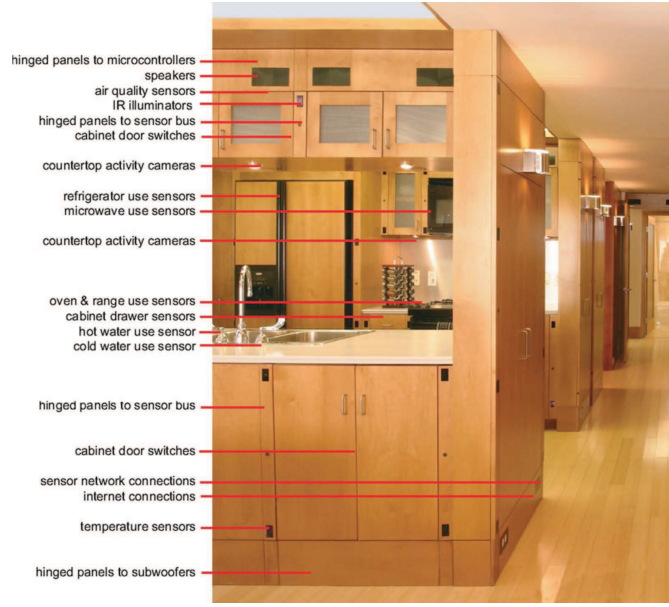


Figure 1.3: Sensors installed in a home setting for activity recognition [2].

through probabilistic reasoning [25, 5, 26]. For example, an activity of “making tea” may contain a series of RFID sensor observations related to “kitchen door”, “cabinet”, “coffee bean can”, “coffee brewer”, “water faucet” and “cup”, showing that the user touches / approaches the above objects.

In this thesis, we focus on sensor-based activity recognition, but do not cover the case of using vision for activity recognition [27, 28, 29, 30, 31], which is popular in vision pattern analysis.

1.1.3 Mobile Recommendation

User interest is the third important component of human behavior. Given the location and activity history of a user, we can predict the user’s interests to the locations and the activities for mobile recommendation.

Example of GPS-based Mobile Recommendation: Figure 1.4 gives such an example. As shown in the figure, a user shares his GPS trajectory showing the locations he visited (depicted as the red trajectory) and the activities he was doing (depicted by the comments in the small pink text boxes). If we know that a user usually goes for drink at some bars after work, and goes to watch movie in the nearby theaters at the weekend from his GPS data, we can then infer that his interests include

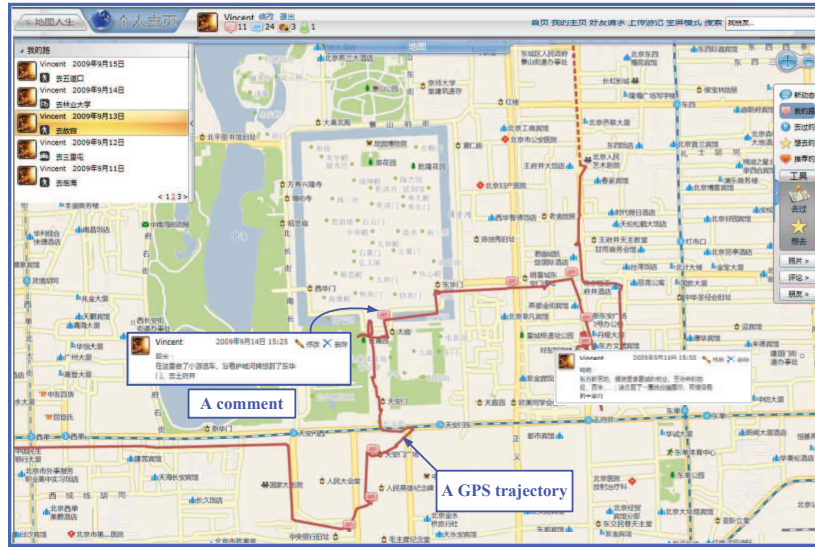


Figure 1.4: Location data sharing service [3].

“drinking” and “watching movie”. After having the location and activity history data from many users, we can know more; for instance what kind of locations are popular, and what activities are suitable at some places for some users? Such crowd interests encoded in the users’ location and activity histories can be used to provide useful mobile recommendations.

There are also some other examples about mobile recommendation: Park et al. designed a restaurant recommendation system based on user’s profile and location’s context [32]. Takeuchi and Sugimoto developed a shop recommendation system based on users’ shop visiting history [33]. Their system predicts a user to be interested in some shop if some similar users are also interested in that shop. Zheng et al. designed a tourism hot spot recommendation system by taking into account both a user’s travel experience (i.e. how many places she visited) and the interest of a location (i.e. how popular this place is) [34].

In general, recommendation techniques can be classified as two categories [35]: one is content-based filtering, and the other is collaborative filtering. In content-based filtering, the user will be recommended items (e.g. locations or activities in the mobile computing scenario) similar to the ones she preferred in the past. Specifically, for each user and each item, some content features are extracted; then based on these content features, one can find the most similar item(s) that the user may be interested in for recommendation [36]. In collaborative filtering, user or item contents are not necessary. Instead, the user will be recommended items that people with similar preferences

liked in the past. The collaborative filtering (CF) algorithms can be further classified into two categories: memory-based CF and model-based CF. Memory-based CF techniques usually use user rating data to compute similarity between users and / or items. Then, some neighborhood-based algorithm is employed to calculate the similarity between two users or items, produces a prediction for the user taking the weighted average of all the ratings [37, 38, 39]. Model-based CF techniques use data mining, machine learning algorithms to find patterns based on training data. Most of these techniques aim to discover the latent factors that explain the rating data. Some representative models include matrix factorization [40, 41, 42, 43, 44], tensor decomposition [45, 46], latent semantic analysis [47], etc. It is generally shown that, model-based CF performs better than memory-based CF [48]. It is also argued that collaborative filtering has several advantages over content-based filtering [49]; for example, it does not depend on error-prone machine analysis of content and it is able to model complex, hard-to-represent concepts, such as taste and quality. In this thesis, we focus on using model-based collaborative filtering, together with some content information, for mobile recommendation.

1.2 Sensors in Human Behavior Recognition

Like many previous technological innovations, sensor technology also helps usher a new era for mobile intelligence with far-reaching implications. The emergence of various sensor hardware has provided the possibility of mobile computing. For example, we can list some popular sensors here:

RFID Radio-frequency identification (RFID) uses an integrated circuit for storing and sending radio-frequency (RF) signal. A RFID reader device can both send and receive signals. Passive RFID tags can read signals when triggered by RF waves sent by the readers, whereas battery-powered active tags can generate and receive signals by themselves. The range of RFID tags is within several meters. RFID has been widely used in product tracking and identification, especially in logistics operations. In Hong Kong, RFID technology is daily used by millions of people in a subway and debit card known as the Octopus Card.

GPS The Global Positioning System (GPS) is a global system based on between 24 and 32 satellites which send RF signals. Receivers can determine their current locations based on the time-of-flight information carried by the RF signals. In an outdoor environment, based on the location sequences, high-level inference can be done to ascertain an agent's transportation modes,

goals and intentions [50, 3].

WiFi Most notebook computers, PDAs and mobile phones today are equipped with WiFi device that can communicate through an IEEE 802.11b/g/n wireless network in the 2.4GHz frequency bandwidth. In an indoor or even outdoor area, the mobile device can send and receive RF signals from various access points (APs). WiFi devices are especially useful for locating a user and tracking his/her movement in an indoor environment where GPS is often no longer available [9, 16].

Mobile Phones The mobile phones nowadays have advanced sensors for measuring some pre-defined activity of the mobile user, such as turning directions. They are known as the Inertial Navigation System (INS), which are motion sensors such as gyroscope, accelerometer and digital compass. With some Near Field Communication (NFC) sensors, people can even use their phones for mobile payments.

1.3 The Data Sparsity Challenges

In the area of mobile computing, we are generally interested in learning some models which can take noisy sensor data as input and some interested entities such as locations and activities as output. These learning algorithms are shown to perform well when there are sufficient labeled and/or unlabeled data in the interested domain. However, in the context of mobile computing, this condition may not be always true. Sensor data are usually sparse. There are several reasons: first, human labeling is very expensive, and thus labeled sensor data are usually scarce. Second, sometimes even the unlabeled data are not extensively available, at least not from the very beginning, because collecting such data can also involve many human efforts. For example, to collect the unlabeled signal data with one mobile device for localization may require carrying the device to walk around the environment. Therefore, these learning algorithms may suffer from such a data sparsity challenge.

In this proposal, we study the data sparsity problem of several typical mobile learning scenarios, covering location estimation, activity recognition and mobile recommendation. In particular, we are interested in the following problems:

- In location estimation, we observe that the sensor (such as WiFi) signals can significantly

change over devices or time periods. Therefore, the model learned from one device’s labeled data or one time period’s labeled data may not work well in other devices or other time periods. However, it is expensive to keep re-collecting labeled and even unlabeled data for the new context, we face a data sparsity challenge. We call it as *the context-dependent data sparsity challenge in location estimation*.

- In activity recognition, we try to build the personalized activity recognition for each user by considering the users’ behavior differences. However, as each user may have limited data, we face the data sparsity challenge in using such limited data to build personalized recognition models. We call this as *the user-dependent data sparsity challenge in activity recognition*. Besides, considering that as some activities may have much fewer data than the others or even no data at all in a short period after the data collection, we may also experience the data sparsity challenge in recognizing them. We call this as *the activity-dependent data sparsity challenge in activity recognition*.
- In mobile recommendation, we try to build some location and activity recommendation system which can answer typical queries, such as “where I should go if I want to do something (such as tourism)” and “what I should do if I go to somewhere (such as downtown)”. However, as each user has limited location and activity visiting history data, we face a data sparsity challenge in modeling users’ interests from their limited previous location and activity history data as the crowd may not be able to explore all the possible locations and activities. We call it as *the crowd-dependent data sparsity challenge in mobile recommendation*.

1.4 Main Contribution

In this thesis, we focus on human behavior recognition and study the challenges of data sparsity. We propose to use transfer learning with auxiliary information to address these data sparsity challenges. The main contributions are listed as follows:

- **WiFi-based location estimation.** From the pervasive computing perspective, we study the context-dependent data sparsity challenges, and show to well address them with auxiliary contexts’ information. From the learning perspective, because the auxiliary contexts’ data

may follow different data distributions with the target contexts' data distributions, we propose to solve the context-dependent data sparsity by *transfer learning with different data distributions*. We extend the standard multi-task learning algorithm and Hidden Markov Model to use data from multiple contexts for training at the same time. In particular, to address the device-dependent data sparsity challenge, we propose a latent multi-task learning algorithm, which enforces feature mapping and multi-task learning together in training. In this way, we can avoid inappropriately forcing different tasks to share the model parameters in the original feature space even their data distributions in it are very different. To address the time-dependent data sparsity challenge, we extend the Hidden Markov Model by incorporating it with a multiple linear regression model, so that the new model can exploit the signal correlations and the sequential information at the same time.

- **Sensor-based activity recognition.** From the pervasive computing perspective, we novelly put forward the user-dependent data sparsity challenge and the activity-dependent data sparsity challenge. These two challenges are not well studied before. We further provide a collaborative activity recognition model and a cross-domain learning model to address them. From the learning perspective, we propose to solve these two data sparsity challenges by *transfer learning with cross task or cross label spaces*. In particular, for the user-dependent data sparsity challenge, we provide a latent aspect model with cross user task spaces to formulate the user grouping information, so that each user can benefit from the similar users' data to get a personalized activity recognizer. For the activity-dependent data sparsity challenge, we provide a cross-domain learning algorithm which can do classification even when there is no training data for a test class (i.e. an activity in our scenario). We achieve this cross-label-spaces transfer learning by borrowing the data from other classes which have data in training. We novelly exploit the inter-class similarity and thus construct weighted pseudo data for training.
- **GPS-based mobile recommendation.** From the pervasive computing perspective, we for the first time give a clear problem formulation on collaborative location and activity recommendation, and show to well address the crowd-dependent data sparsity challenge. From the learning perspective, we address the data sparsity challenge by *transfer learning with collaborative filtering from multiple information sources*. We first provide a collective matrix factorization model to incorporate the auxiliary information about locations and activities for general recommendation. We further extend it to a collective tensor decomposition model

by formulating the user-location-activity relationships for personalized recommendation.

- **Empirical studies.** For each problem, we test our models with the real world sensor data sets, and show that our models can generally outperform the competing baselines. For WiFi-based location estimation, we test our models with some data sets collected from a WiFi building environment of around $64m \times 50m$. We show that our latent multi-task learning model and the transferred hidden Markov model can achieve 19.4% and 16.5% improvement respectively over the non-transfer learning baselines on average. For sensor-based activity recognition, we test our collaborative activity recognition model with a campus-scale WiFi data set, and show an average 10+% performance lift over the baselines that do not consider collaborative activity recognition. We test our cross-domain learning algorithm with several real-world sensor data sets, and show that our model of not using any labeled data for the target activities can still give reasonable activity predictions. It has much higher performances than the random guess solution (in the sense that there is no training data for the test activity), and more important, it can have comparable performance with those traditional supervised activity recognition models of using labeled data for the target activities. For mobile recommendation, we test our models with a real-world GPS data set that was collected for around 2.5 years in Beijing. Our general recommendation system has 7% improvement on activity recommendation and over 20% improvement on location recommendation over the simple baseline without exploiting any additional information. The corresponding personalized recommendation system is shown to further improve the recommendation performance.

1.5 Thesis Outline

Chapter 2 studies the problem of WiFi-based location estimation. First, we overview the general learning methods for WiFi localization. Then, we impose the context-dependent data sparsity challenge through some real-world examples. We survey the related work that tried to solve the problem and point out their advantages and disadvantages. Finally, we propose our transfer learning methods, latent multi-task learning and transferred hidden Markov model, to address the data sparsity challenges with respect to using different auxiliary mobile device’s data and auxiliary time period’s data. We evaluate our proposed methods by carrying out real-world studies on WiFi localization in a campus environment, and show that our methods can greatly outperform the competing

baselines.

Chapter 3 studies the problem of sensor-based activity recognition. First, we overview the general algorithms for sensor-based activity recognition. Then, motivated by our e-health demo system, we discuss the user-dependent data sparsity challenge and the activity-dependent data sparsity challenge respectively. We survey the related work in addressing each of the sparsity challenges, and thus propose an aspect model based collaborative activity recognition solution and a Web information retrieval based cross-domain activity recognition solution to solve the problem. We evaluate our proposed methods by using real-world data sets in both campus and home setting environments, and show the promising results of our methods compared with the competing baselines.

Chapter 4 studies the problem of GPS-based mobile recommendation. First, we overview the existing models in general recommendation systems. Then, we discuss the data sparsity challenge that exists in mobile recommendation scenario and survey the related work. Finally, we proposed a collective matrix factorization solution for general recommendation and a collective tensor decomposition solution for mobile recommendation. We evaluate our proposed methods on a real-world GPS data set that is collected for 2.5 years, and show the effectiveness of our methods through extensive empirical studies.

Chapter 5 offers a unified learning framework for all our proposed transfer learning solutions, and shows how to instantiate it to each previous chapter's problem. We also provide some further discussion on how to exploit more relationships among location estimation, activity recognition and mobile recommendation.

Chapter 6 summarizes our work in human behavior recognition, and lists several future research directions.

Chapter 2

WiFi-based Location Estimation

There are various sensors that can be used for learning-based localization, such as radio frequency (RF) sensors, infrared sensors, ultrasound sensors and others [51]. Among them, radio frequency is the most popular one, and it is extensively used for localization in cellular phone systems [52, 53], Active RFID systems [54, 11], and wireless network systems [10, 55, 16]. In such systems, mobile device moving in the environments can periodically receive beacon frames sent out by the system infrastructures. For example, in cellular phone system, a cell phone can periodically receive beacon signals from the base stations. In the active RFID systems, the tags can detect the signals sent out from the readers. In the wireless network, the mobile device receives the signals from access points (APs). In all these cases, once the mobile device detect the beacon signals, it can measure the corresponding Radio Signal Strength (RSS). Based on the RSS values, either propagation model based or learning based localization algorithms can be used to infer the location of the mobile devices.

Infrared-based localization makes use of ambient light sources, such as incandescent sources, fluorescent lighting and sunlight. Basically, the infrared systems use active beacons distributed at the pervasive computing environments. The beacons can transmit a coded infrared signal which allows the mobile device to identify the sender. Later, some infrared receiver computes the position by using a trilateration method with reduced processing time [56, 57]. However, the infrared signal usually has limited transmission range, which prohibits the infrared localization system to provide a large coverage. Another disadvantage is that the infrared network usually does not provide additional data sharing services [9].

Ultrasound-based localization makes use of ultrasound time-of-flight trilateration technique for location estimation. One famous ultrasound system is the Bat system [58] which improved the original Active Badge system. In that system, a Bat sensor can emit an ultrasound pulse to a grid of ceiling-mounted receivers to measure the signal travel time intervals, so that one can calculate its distances to the receivers. These distances are later used by a central control system to estimate the location by trilateration. Similar ultrasound systems exist, such as the Cricket system [59]. Generally, the ultrasound localization systems have rather high accuracy (in a centimeter level). However, the disadvantages are their high cost and limited scalability due to the requirements on a large amount of expensive ultrasound hardware [60].

Compared with the infrared and ultrasound localization systems, the devices sending and receiving radio frequency signal are much cheaper and more popular. Therefore, the RF-based localization attracts more and more interests of both research and industry community. In this thesis, we focus on using RF sensors, especially WiFi sensors which are extensively available in the real world after the 802.11 Wireless LAN becomes popular, for localization. Much work has been done on WiFi-based localization. In general, there are two categories of localization methods. One is propagation-based model, which tries to formulate the radio propagation and use trilateration/triangulation for localization [13, 14, 15]. The other is learning-based model, which uses statistical learning to formulate the signal-location mappings [16, 17, 18, 19]. Learning-based methods become popular in localization research because of its robustness to signal noises and strong capability in modeling the signal-location dependency [9, 4, 61].

2.1 Background of Learning-based Localization

2.1.1 Data Input and Output

For the learning-based localization, the data inputs include the sensor data obtained from some environment, as well as some possible location labels indicating where the user is given the observed sensor data. Such a location label can be a discrete grid ID or a continuous coordinate vector. Let's consider a two-dimensional indoor localization problem. In the environment, there are m access points (APs), from which we collect the receive signal strength (RSS) data. Each RSS vector is $\mathbf{x} = (x_1, \dots, x_d)' \in \mathbb{R}^d$, and its location label denoting the coordinates is $\mathbf{y} = (y_1, y_2) \in \mathbb{R}^2$.

Therefore, a localization model is a function $f : \mathbf{x} \rightarrow \mathbf{y}$.

Input and Output. In training, we can have a set of labeled sensor data $D_{trn}^{(l)} = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1, \dots, n_l\}$, together with some possible unlabeled data $D_{trn}^{(u)} = \{\mathbf{x}_i | i = n_l + 1, \dots, n_l + n_u\}$, as *inputs*. They are used to learn the localization function. At an online phase, give some sensor observations $D_{tst} = \{\mathbf{x}_j | j = 1, \dots, n_t\}$, one can use the trained localization function to predict the current location of the user as *output*. We call this process as localization from the sensor data. It's generally assumed that D_{tst} have the same data distribution with D_{trn} so that the trained localization model can work correctly in testing. We will later show that, such assumptions may not hold in practice, leading to the context-dependent data sparsity challenge.

In order to obtain the above inputs and outputs, one need to process the data from their raw sensor data formats and annotate them with location labels as below.

Sensor data format. As a user moves, the mobile device such as cell phone can periodically sniff wireless signals from various access points. In general, the received signal strength (RSS) is bigger if the mobile device is closer to an access point; but this is not necessarily always true given many noises in signal propagation. An example of the received signal strength value is “-50 dBm”, where “dBm” is the unit. At each sniff, all the RSS values are combined to generate a raw feature vector, with each feature denoting the RSS value from one AP. Consider an example of three APs, denoted as s_1 , s_2 and s_3 . At one sniff, if the RSS values from s_1 , s_2 and s_3 are “-50 dBm”, “-60 dBm” and “-30 dBm” respectively, then we can generate a raw feature vector of $\mathbf{x} = (s_1, s_2, s_3) = (-50, -60, -30)$. Such a feature vector is associated with a location, which is a discrete “Room 4215” or a point “(32.3, 61.5)” in some coordinate system such as latitude/longitude. The sampling rate of signal sniff is expected to be neither too high nor too low. A too high sampling rate may drain the battery quickly, while a too low sampling rate may not be enough to capture the user movement.

Location annotation. The location labels can be discrete or continuous. For the discrete cases, the interested location space can be divided into a set of regular grids [62, 63], or irregular grids composing a Voronoi graph [64], or semantic places such as rooms [65, 66]. For the continuous cases, the interested location can be represented as some point in the self-defined coordinate system [67, 20] or the geographic coordinate system with latitude and longitude [68, 61]. Given the location definitions, one may need to annotate the wireless signal data with respect to the ground truth location for training and testing. Such an annotation process can be done manually by the sub-

ject or automatically by the GPS device attached to the mobile device if in outdoor environments. Having the possible location labels, we can use some learning techniques to train the localization models.

2.1.2 Existing Learning Models

Generally, learning-based WiFi localization methods work in two phases: in an offline phase, a mobile device moving around the wireless environment is used to collect wireless signals from various access points. Then, the received signal strength values, together with the location information, are used as the training data to learn a statistical localization model. In an online phase, the learned localization model is used to infer the locations according to the real-time RSS values.

Most of the existing learning-based localization algorithms fall into two categories. One is *non-sequential model*, which does not utilize the sequential information in modeling; the other is *sequential model*, which encodes the sequence constraints in the model.

Non-sequential Model: This category of models does not consider the user's previous locations in the current prediction.

- **Nearest Neighbor Method.** Nearest neighbor is a simple yet effective approach for localization. The RADAR system developed by Microsoft Research [9] maps the signal to location by using nearest neighbor heuristics. Each signal strength sample is compared with the radio map, which is a set of signal strength samples collected at some offline phase. The coordinates of the best matches are averaged to give a location estimation. The accuracy of RADAR is about three meters with fifty percent probability. A similar technique is applied in the LANDMARC system [11] with reference RFID tags. This method computes the distances between the signal strength vectors received at the mobile tag and the signal vectors at the reference tags respectively. Then, the top k nearest distances are used as weights for averaging the reference tags' coordinates to estimate the mobile tag's location.
- **Other Non-probabilistic Methods.** There are various non-probabilistic classification or regression methods that can be applied to localization. For example, Nguyen et al. used kernel learning (i.e. support vector machine) to predict the locations of sensors by taking the transmitted signals from base sensors as features and the location regions as labels [69].

One can also consider the localization problem as dimensionality reduction problem, which maps a high-dimensional signal space to a low-dimensional location space. Therefore, many non-probabilistic dimensionality can be used. For example, Shang et al. tried to recover all the locations of sensor nodes simultaneously through multi-dimensional scaling (MDS), which requires the inter signal data distances to be consistent with the location proximities [70]. Patwari and Hero extended the MDS approach to manifold learning through Laplacian Eigenmap [71]. Similarly, Pan et al. applied a manifold regularization approach to reduce the calibration effort by using unlabeled sensor data in localization [20]. Some other dimensionality reduction techniques are also exploited for location estimation. For example, Pan et al. emphasized the correlation between the signal and physical spaces, and used Canonical Correlation Analysis (CCA) to find a latent feature space where the correlation between the two spaces is maximized [72].

- **Probabilistic Method.** The basic idea of probabilistic model in localization is to model the conditional probability $p(\mathbf{x}|y)$ over the location y based on its received signal strength vector \mathbf{x} , then try to maximize the posterior probability $p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \propto p(\mathbf{x}|y)p(y)$. Here, the likelihood function $p(\mathbf{x}|y)$ can be formulated by using histogram, kernel method [73] or Gaussian distribution [55]. There are further improvements on these basic probabilistic localization models. For example, to reduce the computation cost in inference, Youssef et al. proposed to use a joint clustering technique to group locations, and thus further infer the most probable location within the cluster [74].

Sequential Model: This category of models take the sequence information into account for localization.

- **Bayesian filtering Model.** To exploit the sequential information for accurate localization, much work has been done by using graphical models. The robotics navigation system described in [10] employs a Hidden Markov Model (HMM) to formulate the strong correlation between consecutive signal samples. It utilizes the spatial constraint of a user's movement trajectories to help refine the location estimation and reject the estimates that have significant changes in the physical location space. The system is shown to outperform the basic Bayesian inference method by an accuracy of 83% over 77% within 1.5 meters. A similar technique is used in [17]. This work discusses applying different Bayes filters, including Kalman filter and Particle filter [75, 76], to estimate a dynamic system's state (i.e. locations)

from noisy observations. Basically, these approaches fall into the category of employing *directed* graphical models for user tracking. Further work extends the idea of modeling the trace sequences by introducing the *undirected* graphical models. For examples, [77] proposed a semi-supervised statistical relational learning approach based on Conditional Random Fields (CRF). By exploiting a generalized EM algorithm coupled with domain constraints, their semi-CRF method was shown to reduce the calibration effort and increase the tracking accuracy.

- **Gaussian Process** A central difficulty in modeling sequential data is to determine a model which can capture the data nonlinearities without overfitting. In order to overcome this difficulty, Gaussian process (GP) is introduced to model the dynamics by averaging over the parameters rather than estimating them. In [78], Schwaighofer et al. showed how to apply GP to provide interpolation over continuous locations with direct modeling of uncertainty from the training data. In [79], Ferris et al. extended this technique to WiFi localization by combining the GP with graph-based tracking, allowing for accurate localization in large scale spaces. And in their following work [61], they proposed a WiFi-SLAM technique based on Gaussian process latent variable models (GP-LVM), for building wireless signal strength maps without requiring any location labels in the training data. Similarly, in [80], Wang et al. extended the GP-LVM with a Gaussian Process Dynamical Model (GPDM) to integrate out the model parameters in a closed form. It amounts to using Gaussian process priors for both the dynamics and the observation mappings.

2.2 The Context-dependent Data Sparsity Challenge

A major drawback of traditional localization methods is that they assume the collected signal data are context-independent. In other words, the signal data distribution can be the same even the context changes. However, this assumption usually does not hold in practice. For example, sensor signals may vary from device to device due to their different signal sensing capacities, or from time to time due to multi-path fading effects with signal refraction or diffraction. Figure 2.1 gives such an evidence. The received signal strengths can vary significantly on different devices or time periods, even though they were detected from a same access point at a same location. Through some real-world studies, we find that if we keep collecting sufficient labeled data in the

new context (including time and devices), we can provide some localization results with error distance at around 1.5 meters. However, if we do not collect labeled data in the new context, but use the old context’s data, we may have the localization performance greatly drop to 6 meters for the time-varying case and 18 meters for the device-varying case according to our real-world studies as shown later. This observation motivates us to well consider such signal variation (and thus have few/sparse data in the new context) problem.

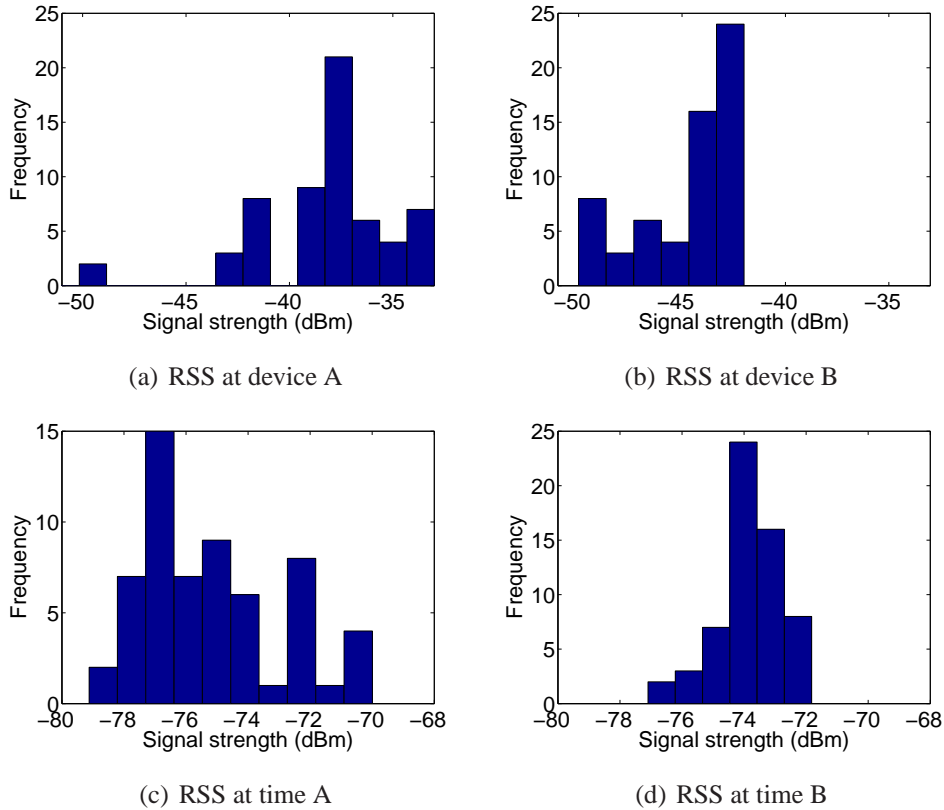


Figure 2.1: Signal variations over devices and time periods.

As we cannot afford collecting a large amount of labeled and/or unlabeled data all the time when the context changes, we are facing the data sparsity challenge in learning some localization model for a new mobile device or a new time period. Traditional learning algorithms may just ignore the signal data difference between different contexts, and use the other existing device’s data or previous time period’s data for model training. In general, such a simple strategy by overlooking the difference may greatly deteriorate the localization performance. Because of that, even some commercial localization software like Ekahau¹ only maintains a list of supported hardware devices.

¹<http://www.ekahau.com/>

Problem	Algorithm	Data-level	Model-level	Main drawback
Device dependent	MeanShift [55]	✓		assume Gaussian mean shift
	HLF [81]		✓	assume stable signal ratios
Time dependent	LEASE [16]		✓	require much hardware
	LANDMARC [11]		✓	require much hardware
	ModelTree [82]	✓		cannot use sequence
	LeManCoR [83]		✓	cannot use sequence

Table 2.1: Review on previous work of location estimation.

This motivates us to take these auxiliary data’s difference into account and carefully design the learning algorithm.

2.2.1 Related Work

There is few work addressing the context-dependent data sparsity problem in wireless localization. We list some typical work in Table 2.1. Haeberlen et al. treat signal variation as a Gaussian mean-value shift (denoted as *MeanShift* method in the table), and use a linear model $c(i) = c_1 \cdot i + c_2$ to fit the RSS value $c(i)$ on a target device, with the RSS value i on an auxiliary device for each access point [55]. Here, c_1 and c_2 are model parameters to be computed by least square fit. However, in a complex indoor environment, such a simple function may not capture possibly nonlinear signal correlations. Kjargaard and Munk proposed a hyperbolic location fingerprinting (HLF) method to address the device signal variation problem [81]. They turn the absolute RSS values into some ratios among different access points, and use them to train a localization model. Some possible drawback is that, it cannot handle the case when some APs are not observed . The *LANDMARC* system [11] and the *LEASE* system [16] both utilize some additional hardware equipments, including stationary emitters and sniffers, to obtain up-to-date RSS values and further apply some KNN-style algorithms to do location estimation. However, such methods may suffer from only being able to use limited number of sniffers in terms of hardware cost and limited modeling power from KNN. ModelTree [82] applies a regression analysis to learn the temporal predictive relationship between the RSS values received by sparsely located reference points and that received by the mobile device. Then it uses the newly observed RSS values at the device and the reference points for localization with some decision tree style algorithm. LeManCoR [83] is a semi-supervised manifold method. It treats different time as multiple views and uses a multi-view

learning framework to constrain the predictions on reference points to be consistent. We categorize the above methods into “data-level” methods if they try to fit the data in the new context; otherwise, we categorize them into “model-level” as shown in the table. In general, “model-level” methods have better performance than “data-level” methods because they do not suffer from the difficulty in choosing appropriate signal mapping function between two contexts.

Therefore, in the following, we first propose a learning framework to incorporate the auxiliary data, and then instantiate two possible ‘model-level’ solutions (to be precise, one of them is both “model-level” and “data-level”) to solve the device-dependent data sparsity challenge and the time-dependent data sparsity challenge.

2.3 Latent Multi-Task Learning for Multi-Device Localization

For device-dependent data sparsity challenge, rather than directly modeling the data correlations between devices as the “data-level” methods did, we instantiate the learning framework as a multi-task learning problem to exploit the model correlations. Multi-task learning was proposed to exploit the task relatedness for improving the learning performance [84, 85]. The idea behind multi-task learning is that it pools together the data from all the related tasks, such that the learning on each task can benefit from other tasks by sharing some model parameters. This coincides with our motivation by utilizing other auxiliary device’s data to address the sparsity problem. As the data distributions are usually different on different devices, we extend the multi-task learning for this multi-device localization problem by only requiring their model hypotheses for sharing are similar in a latent feature space. In other words, we look for appropriate feature mappings, by which we can map different devices’ data to a well-defined low-dimensional feature space. In this latent space, new device can benefit from integrating the data collected before by other devices to train a localization model.

2.3.1 Problem Formulation and Our Solution

Consider a two-dimensional indoor localization problem. In the environment, there are m access points, from which we collect the receive signal strength (RSS) data. Each RSS vector is $\mathbf{x} = (x_1, \dots, x_d)' \in \mathbb{R}^d$, and its location label denoting the coordinates is $\mathbf{y} = (y_1, y_2) \in \mathbb{R}^2$. On

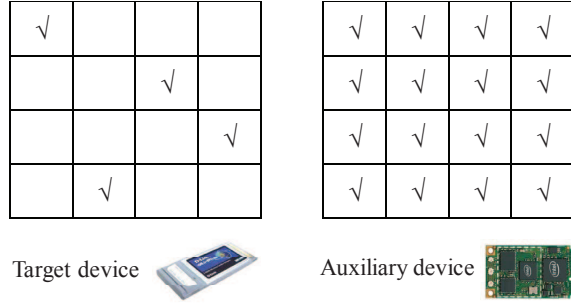


Figure 2.2: Setting for the device-dependent data sparsity challenge.

the auxiliary device, we have collected a large amount of labeled data $D_{aux} = \{(\mathbf{x}_{aux}^{(i)}, \mathbf{y}_{aux}^{(i)}) | i = 1, \dots, n_{aux}\}$; on the target device, we only collect a small amount of labeled data $D_{tar} = \{(\mathbf{x}_{tar}^{(i)}, \mathbf{y}_{tar}^{(i)}) | i = 1, \dots, n_{tar}\}$. Finally, we also have a test data set from the target device $D_{tar}^{tst} = \{(\mathbf{x}_{tar}^{tst(i)}, \mathbf{y}_{tar}^{tst(i)}) | i = 1, \dots, n_{tar}^{tst}\}$. The data setting is illustrated in Figure 2.2. A matrix in the figure denotes a 2D location space and a tick indicates labeled data collected in that location.

We model localization with a regression function $f(\mathbf{z}) = \mathbf{w} \cdot \mathbf{z} + b$ to output locations from some transformed signal vector $\mathbf{z} = \varphi(\mathbf{x}) \in \mathbb{R}^k$. Here, \mathbf{w} is a regression weight vector, b is a bias term and φ is a feature mapping function. In this multi-device problem, we treat T devices as T tasks², and each task $t \in \{1, \dots, T\}$ has a regression hypothesis parameterized³ as \mathbf{w}_t . We follow [85] to make \mathbf{w}_t share a common structure \mathbf{w}_0 by

$$\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t, \quad \forall t = 1, \dots, T,$$

where \mathbf{v}_t denotes the difference for each task t . We are interested in finding appropriate feature mappings φ_t which can map the raw signal data to a k -dimensional latent feature space where the learned hypotheses are similar, i.e. \mathbf{v}_t is “small”.

Finally, we provide a latent multi-task learning solution based on soft-margin support vector re-

² $T = 2$ as we consider one target device and one auxiliary device.

³We fix b across tasks for computation simplicity.

gression (SVR) [86]:

$$\begin{aligned}
& \min J(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b, \varphi_t) \\
& = \underbrace{\sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*)}_{\text{loss}} + \underbrace{\frac{\lambda_1}{T} \sum_{t=1}^T \|\mathbf{v}_t\|^2}_{\text{knowledge share}} + \underbrace{\lambda_2 \|\mathbf{w}_0\|^2 + \frac{\lambda_3}{T} \sum_{t=1}^T \Omega(\varphi_t)}_{\text{regularization}} \quad (2.1) \\
& \text{s.t.} \quad \begin{cases} y_{it} - (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) - b \leq \varepsilon + \xi_{it} \\ (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) + b - y_{it} \leq \varepsilon + \xi_{it}^* \\ \xi_{it}, \xi_{it}^* \geq 0 \end{cases}
\end{aligned}$$

We explain each term in Eq.(2.1):

- In the *loss* term, ξ_{it} and ξ_{it}^* as slack variables measuring the errors. π_t are the weight parameters for each task t .
- In the *knowledge share* term, minimizing $\|\mathbf{v}_t\|^2$ regularizes the dissimilarity among the task hypotheses in the latent feature space $\varphi_t(\mathbf{x})$.
- In the *regularization* term, minimizing $\|\mathbf{w}_0\|^2$ corresponds to maximizing the margin of the learned models to provide the generalization ability. Generally, parameter λ_1 is larger than λ_2 to force the task hypotheses to be similar. $\Omega(\varphi_t)$ denotes the complexity of mapping function φ_t . To make our problem tractable, we consider $\varphi_t \in \mathbb{R}^{k \times d}$ as a linear transformation by letting $\varphi_t(\mathbf{x}) = \varphi_t \mathbf{x}$. We interpret $\Omega(\varphi_t) = \|\varphi_t\|_F^2$ as the Frobenius norm.
- The constraints follow the routine of the standard ϵ -SVR [86], with b as the bias term and ϵ as the tolerance parameter.

Notice that Eq.(2.1) consists of a convex objective function and bilinear constraints, the whole optimization problem is not convex. However, separately considering the parameters $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$ and φ_t , we can reduce the objective function to a convex optimization problem. Motivated by this, we employ an alternating optimization approach to solve the problem, by iteratively optimizing the parameters $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$ with fixed φ_t and optimizing φ_t with fixed $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$. Therefore, our latent multi-learning algorithm consists of two steps: the first step learns the parameters of the regression function and the second step learns low-dimensional feature mapping functions for all the tasks.

Learning the Regression Function Given mapping functions $\varphi_t, t = 1, \dots, T$, we re-formulate Eq.(2.1) as a standard ε -SVR problem by feature mapping. Notice that the regression functions are used to predict the location labels $\{y_{it}\}$ for signal data $\{\mathbf{x}_{it}\}$, this step corresponds to *offline training* localization model in a latent feature space $\varphi_t(\mathbf{x})$.

First, we consider the function $f_t(\varphi_t(\mathbf{x})) = \mathbf{w}_t \cdot \varphi_t(\mathbf{x}) + b, t = 1 \dots T$. This function can be identified by a real-valued function $F : X \times \{1, \dots, T\} \rightarrow \mathbb{R}$ with $F(\varphi_t(\mathbf{x}), t) = f_t(\varphi_t(\mathbf{x}))$, which takes $((\varphi_t(\mathbf{x}), t), y)$ as training examples. Then, we can define a feature mapping on $(\varphi_t(\mathbf{x}), t)$ as:

$$\phi((\varphi_t(\mathbf{x}), t)) = \left(\frac{\varphi_t(\mathbf{x})}{\sqrt{\mu}}, \underbrace{\mathbf{0}, \dots, \mathbf{0}}_{t-1}, \varphi_t(\mathbf{x}), \underbrace{\mathbf{0}, \dots, \mathbf{0}}_{T-t} \right), \quad (2.2)$$

where $\mathbf{0} \in \mathbb{R}^d$ is a zero vector, and $\mu = \frac{T\lambda_2}{\lambda_1}$.

Theorem 5. *The latent multi-task learning in Eq.(2.1) can be re-formulated as a standard ε -SVR problem:*

$$\begin{aligned} \min \tilde{J}(\mathbf{w}, \xi_{it}, \xi_{it}^*) &= \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) \\ s.t. \quad &\begin{cases} y_{it} - \mathbf{w} \cdot \phi((\varphi(\mathbf{x}_{it}), t)) - b \leq \varepsilon + \xi_{it} \\ \mathbf{w} \cdot \phi((\varphi(\mathbf{x}_{it}), t)) + b - y_{it} \leq \varepsilon + \xi_{it}^* \\ \xi_{it}, \xi_{it}^* \geq 0 \end{cases}, \end{aligned} \quad (2.3)$$

where $C = \frac{T}{2\lambda_1}$, and $\mathbf{w} = (\sqrt{\mu}w_0, v_1, \dots, v_T)$.

Proof. Consider $\mathbf{w} = (\sqrt{\mu}w_0, v_1, \dots, v_T)$. Hence, $\mathbf{w} \cdot \phi((\varphi(\mathbf{x}), t)) = (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi(\mathbf{x})$, and $\|\mathbf{w}\|^2 = \sum_{t=1}^T \|\mathbf{v}_t\|^2 + \mu \|\mathbf{w}_0\|^2$. Given φ_t , the term $\frac{\lambda_3}{T} \sum_{t=1}^T \Omega(\varphi_t)$ is constant. Hence, the objective function can be re-formulated as a standard ε -SVR:

$$\begin{aligned} \min J &= \sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) + \frac{\lambda_1}{T} \sum_{t=1}^T \|\mathbf{v}_t\|^2 + \lambda_2 \|\mathbf{w}_0\|^2 \\ &= \frac{T}{2\lambda_1} \sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) + \frac{1}{2} \left(\sum_{t=1}^T \|\mathbf{v}_t\|^2 + \frac{T\lambda_2}{\lambda_1} \|\mathbf{w}_0\|^2 \right) \\ &= C \sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) + \frac{1}{2} \|\mathbf{w}\|^2. \end{aligned}$$

□

Corollary 6. *The optimal solution of the regression function $f(\cdot)$ in Eq.(2.3) is given:*

$$f^*(\mathbf{x}) = \sum_{t=1}^T \sum_{i=1}^{n_t} (\alpha_{it} - \alpha_{it}^*) K(\mathbf{x}_{it}, \mathbf{x}) + b,$$

where $K(\mathbf{x}_{is}, \mathbf{x}_{jt}) = (\frac{1}{\mu} + \delta_{st})\varphi_s(\mathbf{x}_{is}) \cdot \varphi_t(\mathbf{x}_{jt})$. δ_{st} is an indicator function, which equals to 1 if $s = t$, and 0 otherwise.

Proof. We first derive the dual of Eq.(2.3). Denote $\phi((\varphi(\mathbf{x}_{it}), t))$ as $\hat{\mathbf{x}}_{it}$. The Lagrangian for Eq.(2.3) is

$$\begin{aligned} L = & \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*) \\ & + \sum_{t=1}^T \sum_{i=1}^{n_t} \alpha_{it} (y_{it} - \mathbf{w} \cdot \hat{\mathbf{x}}_{it} - b - \varepsilon - \xi_{it}) + \sum_{t=1}^T \sum_{i=1}^{n_t} \beta_{it} \xi_{it} \\ & + \sum_{t=1}^T \sum_{i=1}^{n_t} \alpha_{it}^* (\mathbf{w} \cdot \hat{\mathbf{x}}_{it} + b - y_{it} - \varepsilon - \xi_{it}) + \sum_{t=1}^T \sum_{i=1}^{n_t} \beta_{it}^* \xi_{it}^*. \end{aligned}$$

Let $\frac{\partial L}{\partial \mathbf{w}} = 0$, $\frac{\partial L}{\partial \xi_{it}^*} = 0$, and $\frac{\partial L}{\partial b} = 0$. We solve these equations and plug the solutions back to L . Then, we have the dual for Eq.(2.3) as follows:

$$\begin{aligned} \max_{\alpha_{it}, \alpha_{it}^*} & -\varepsilon \sum_{t=1}^T \sum_{i=1}^{n_t} (\alpha_{it} + \alpha_{it}^*) + \sum_{t=1}^T \sum_{i=1}^{n_t} y_{it} (\alpha_{it} - \alpha_{it}^*) \\ & - \frac{1}{2} \sum_{s=1}^T \sum_{i=1}^{n_s} \sum_{t=1}^T \sum_{j=1}^{n_t} (\alpha_{is} - \alpha_{is}^*) (\alpha_{jt} - \alpha_{jt}^*) K(\mathbf{x}_{is}, \mathbf{x}_{jt}) \\ \text{s.t.} & \sum_{t=1}^T \sum_{i=1}^{n_t} (\alpha_{it} - \alpha_{it}^*) = 0 \quad \text{and} \quad \alpha_{it}, \alpha_{it}^* \in [0, C\pi_t]. \end{aligned} \quad (2.4)$$

The optimal solution for the regression function can be obtained by solving this dual. \square

Based on $f^*(\mathbf{x})$, we can have the estimations of $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$ for the next step.

Learning the Latent Feature Space Given the regression function parameters $(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b)$ from the previous step, we can reduce the objective function to

$$\begin{aligned} \min \hat{J}(\varphi_t) &= \sum_{t=1}^T \|\varphi_t\|_F^2 \\ \text{s.t.} & \begin{cases} y_{it} - \mathbf{w}_t \cdot \varphi_t \mathbf{x}_{it} - \mathbf{b} \leq \varepsilon + \xi_{it} \\ \mathbf{w}_t \cdot \varphi_t \mathbf{x}_{it} + \mathbf{b} - y_{it} \leq \varepsilon + \xi_{it}^* \end{cases}. \end{aligned} \quad (2.5)$$

By optimizing over the feature mapping functions φ_t , we find the common latent space for the previous step's offline localization model training.

Theorem 7. Eq.(2.5) is equivalent to a standard Quadratic Programming (QP) problem.

Algorithm 1 Algorithm of Latent Multi-task Learning (LatentMTL)

Input: Labeled auxiliary device data D_{aux} , labeled target device data D_{tar} , unlabeled target device test data D_{tar}^{tst} .

Output: Labels for D_{tar}^{tst} .

begin

- 1: Initialize the latent space dimensionality k and the feature mapping functions φ_t ;
- 2: **repeat**
- 3: Learn the regression function $f^*(\mathbf{x})$ in the latent feature space φ_t , as shown in Eq.(2.3)
- 4: Learn the latent space by optimizing the feature mapping functions φ_t , as shown in Eq.(2.5);
- 5: **until** In Eq.(2.1), J 's change is less than a threshold ζ
- 6: **return** Label for each $\mathbf{x}_i \in D_{tar}^{tst}$ by $f^*(\mathbf{x})$.

end

Proof. First, we vectorize the mapping function matrix φ_t as $\tilde{\varphi}_t = [\varphi'_{t1}, \dots, \varphi'_{td}]'$, with each $k \times 1$ vector φ_{ti} is the i^{th} column of the matrix φ_t . Hence,

$$\mathbf{w}_t \cdot \varphi_t \mathbf{x}_{it} = (\mathbf{w}'_t \varphi_t) \mathbf{x}_{it} = [\mathbf{w}'_t \varphi_{t1}, \dots, \mathbf{w}'_t \varphi_{td}] \mathbf{x}_{it} = [\mathbf{x}_{it}^1 \mathbf{w}'_t, \dots, \mathbf{x}_{it}^d \mathbf{w}'_t] \tilde{\varphi}_t.$$

Define the data for task t as X_t , a $n \times d$ matrix with each row as a data point \mathbf{x}_{it} . Then, the conditions in Eq.(2.5) can be re-formulated as

$$A_1 \leq B \tilde{\varphi}_t \leq A_2,$$

where A_1 is a $n_t \times 1$ vector with each element $A_1^i = y_{it} - b - (\varepsilon + \xi_{it})$. Similarly, A_2 is a $n_t \times 1$ vector with each element $A_2^i = y_{it} - b + (\varepsilon + \xi_{it}^*)$. B is a Kronecker product over data matrix X_t and the transposed regression weight \mathbf{w}'_t , i.e. $B = X_t \otimes \mathbf{w}'_t$. Both A_1 , A_2 and B can be calculated from previous section by solving the standard SVR problem. Notice that the minimization in Eq.(2.5) is independent among each task t , we can re-formulate it as a QP problem:

$$\begin{aligned} \min \hat{J}(\varphi_t) &= \tilde{\varphi}'_t \cdot I \cdot \tilde{\varphi}_t \\ \text{s.t. } &A_1 \leq B \tilde{\varphi}_t \leq A_2. \end{aligned}$$

□

Any QP solver can be used to derive the optimal solution. The convergence is guaranteed in a finite number of iterations after alternating optimization [87]. The algorithm is given in 2.3.1.

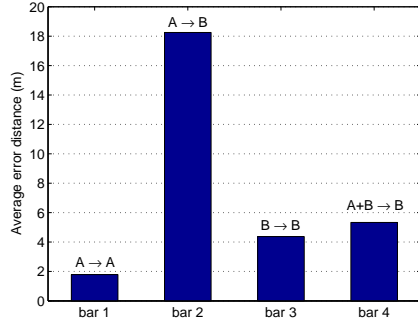
2.3.2 Empirical Study

In this section, we study the benefits of transferring multi-device localization models using our latent multi-task learning algorithm. Our experiments were set up in a $64m \times 50m$ department office area. The calibration data were collected by two laptops with different wireless adapters, denoted as A and B . Our goal is to calibrate the target device B with the help of the auxiliary device A . We collected training and test data in total 107 locations. At each location, we collected 20 samples for each device. For each device, we randomly splitted 50% of the data for training, and the rest for testing. The collected signal vector has 118 dimensions, and the latent space dimension k is set to be 60. We tested different k -values and found that $k = 60$ seems to give the best results. When k is too large, the computation cost is too high. When k is too small, the performance may suffer due to information loss. Since the localization problem is a regression problem, we follow [61] to report the average *error distance*⁴ and the standard deviation.

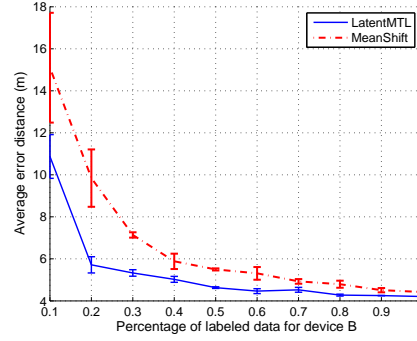
First of all, we would like to confirm our motivation for considering multi-device localization. Figure 2.3(a) shows the statistical results of five trials. In this figure, bar 1 corresponds to learning an SVR model on device A 's training data, and then test the model on device A 's test data. The average error distance is $1.79 \pm 0.08m$. Bar 2 in the figure corresponds to training an SVR model on device A 's training data, and test on device B 's test data. We get an average error distance of $18.25 \pm 1.82m$. Compared to bar 1, such a performance is far from satisfactory⁵. This testifies our observation that the data distributions on different devices are quite different. Thus we need to conduct adaptation between devices. Our next question is: how well our *LatentMTL* method can perform? To answer this question, we use 100% of the device B 's training data to train an SVR model, and test this model on device B 's test data. As shown in bar 3 in Figure 2.3(a), we get an average error distance of $4.37 \pm 0.05m$. Note that this error distance derived from device B is much larger than that from device A as shown by bar 1. From Figure 2.1, we can know the reason clearly: while device A detects a signal strength value of $-60dBm$, device B measures the same signal as $-80dBm$ from a same AP at a same location. This means that device B has a lower signal sensing capacity than device A , which testifies our argument that different devices have different capacities for signal sensing. In bar 4, we show that our method can be close to the result in bar 3

⁴Error distance is calculated by the euclidean distance between a predicted location and its ground truth value. The average error distance is the mean of the error distances over all the test data. We aim to minimize the error distance.

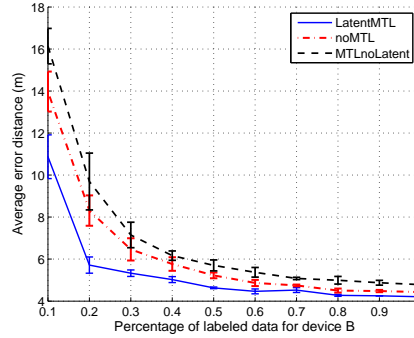
⁵We get similar result using device B 's data for training and using device A 's data for testing. Therefore, in the rest of the paper, we only consider adapting from device A (as auxiliary device) to device B (as target device).



(a) Empirical study of the need for device adaptation.



(b) LMTL-MeanShift



(c) LatentMTL vs. noMTL & MTLnoLatent

Figure 2.3: Experimental results for LatentMTL.

with $5.33 \pm 0.16m$, by using only 30% of device B’s training data and full device A’s training data.

Secondly, we compare our *LatentMTL* method with a baseline method known as *MeanShift* [55], to show that the signal variation is not a simple Gaussian mean shift. The basic assumption for *MeanShift* is that the effect of hardware variation on received signal strength is linear. Hence, for each AP signal, a linear model $c(i) = c_1 \cdot i + c_2$ can be designed to fit the RSS value $c(i)$ on a target device, using the RSS value i on an auxiliary device. In the above equation, c_1 and c_2 are model parameters, which can be computed by least square fit on both devices’ data collected from a same set of locations. We vary the number of device B’s labeled data for fitting the linear model. And after fitting the model, we apply SVR to learn a localization model for device B by using both the existing labeled data and the fitted data, in order to simulate *MeanShift*. As shown in Figure 2.3(b) by a five-trial test, our *LatentMTL* consistently outperforms the *MeanShift* method. That is because in a complex indoor environment, the linear assumption does not hold for the *MeanShift* method.

We also observe that our *LatentMTL* algorithm quickly converges as the number of device B’s data increases. Therefore, practically, we only need to collect at most 30% of the data to calibrate a new device, which gives a large reduction of the calibration effort compared to recollecting all the data over the whole region for a new device.

Thirdly, we design experiments to show the benefits of using latent multi-task learning for multi-device localization. We compare our method with two baselines. The first baseline algorithm corresponds to not using multi-task learning (*noMTL*) in localization, which means that we use only device B’s available labeled data for training. The second baseline corresponds to using multi-task learning in the original feature space (*MTLnoLatent*), which means that we perform multi-task learning on the data from both devices without any latent feature mapping. The second baseline corresponds to the approach given in [85]. As shown in Figure 2.3(c) in a five-trial test, our *LatentMTL* consistently outperforms the two baselines on increasing size of training data. It is interesting to observe that *noMTL* is consistently better than *MTLnoLatent*. This is because data from the two devices can have very different distributions in the original feature space, such that in this space their learned hypotheses may not be similar. Using traditional multi-task learning in this setting may harm the performance. Such observation testifies our motivation of only requiring the learned hypotheses to be similar in a latent feature space.

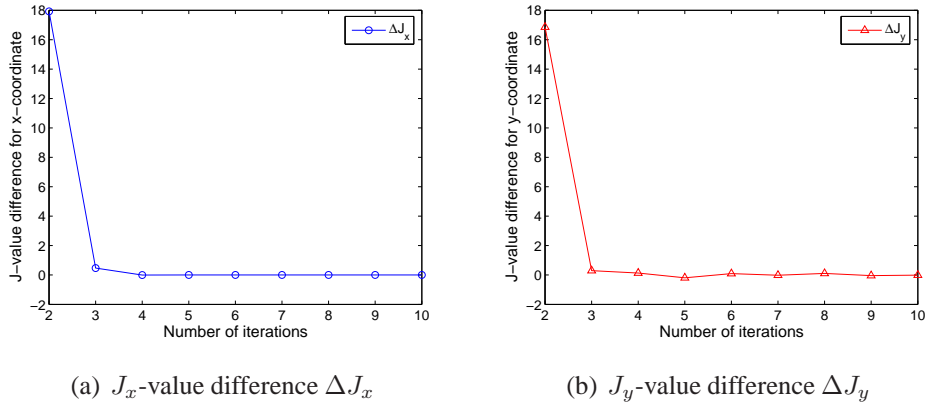


Figure 2.4: Convergence of *latentMTL* algorithm.

Finally, we show the convergence property of our algorithm. In our algorithm, we need to separately train a model on both x and y coordinates for a 2-D location space. By making use of all of device A’s training data and 10% of device B’s training data, we record the J -value as J_x and J_y for each iteration i . The convergence is measured by the J -value difference through $\Delta J_x = J_x(i + 1) - J_x(i)$ and $\Delta J_y = J_y(i + 1) - J_y(i)$. As shown in Figure 2.4, our *latentMTL*

algorithm quickly converges on this data set.

We also test the running time of *LatentMTL* on our Pentium 4 PC (2.13GHz, 1G RAM). Solving Equation 2.1 with 1,470 118-dimensional data samples requires less than 10 CPU minutes using MATLAB software. Since a new device only needs to be calibrated once, such running time for training a localization model on a device is practical.

2.4 Transferred Hidden Markov Model for Time-varying Localization

In the time-dependent signal variation case, as it is natural to install some sniffers to receive up-to-date signal data (which does not require any extra human effort), we can see “data-level” method as a feasible way to handle the data sparsity challenge. So we may be interested in designing some “model-level” method while integrating the “data-level” idea. Besides, we may also want to use other information such as sequences. Therefore, we are motivated to instantiate the learning framework with a transferred Hidden Markov Model (TrHMM). Our choice of HMM model is based on the following considerations. First, compared with other localization models which can use the RSS samples independently, HMM can also use the trajectory information. Second, HMM has well-structured parameters, with which we can easily transfer the knowledge.

An HMM is parameterized as $\theta = (\lambda, A, \pi)$, where λ is a radio map which measures RSS values at each location, A is a transition matrix which indicates user movement patterns, and π is the prior knowledge on where the user is. Among all these parameters, π is relatively stable as the basic human behavior does not change dramatically. For example, a professor usually stays at his office longer than his walking in a corridor. Therefore, it can be shared directly between a target time period and an auxiliary time period. A learned from the auxiliary data can provide a reasonable initialization, and we can use only a small amount of target sequence data to learn a final transition matrix for prediction. As signals change over time, the radio map λ also changes from time to time. Our idea to share such a radio map learned from the auxiliary data is that, we want to exploit the signal correlations among locations. If we know how the signal changes at some locations as time elapses, we may be able to predict how the signal changes at other locations.

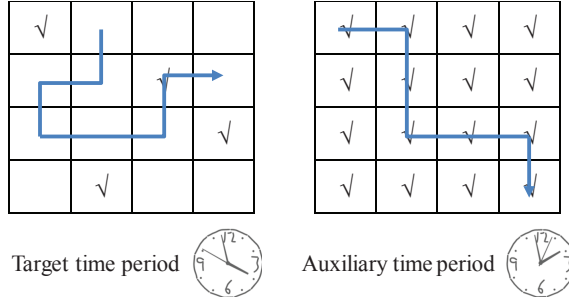


Figure 2.5: Setting for the time-dependent data sparsity challenge.

2.4.1 Problem Formulation and Our Solution

We consider localization as a classification problem. For an auxiliary time period, we have a set of labeled traces $T_{aux} = \{(tr_i, q_i) | i = 1, \dots, n_{aux}\}$, with each trace as $tr_i = (\mathbf{x}_1, \dots, \mathbf{x}_{|tr_i|})$ and its location sequence as $q_i = (y_1, \dots, y_{|tr_i|})$. Here, $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{D}$. For the target time period, we have a set of unlabeled traces $T_{tar} = \{tr_i | i = 1, \dots, n_{tar}\}$. Besides, we also collected signal data from a predefined set of locations L by installing some wireless adapters as reference points to automatically keep track of the signal changes. Let us denote the data collected at the auxiliary time period as $D_{aux}^{\in L}$ and at the target time period as $D_{tar}^{\in L}$. We also have some signal data collected from the locations other than L at the auxiliary time period as $D_{aux}^{\notin L}$ to build the signal correlations among locations. The data setting is illustrated in Figure 2.5. A matrix in the figure denotes a 2D location space, a tick indicates labeled data collected in that location, and the blue arrow line denotes sequence data collected across some locations.

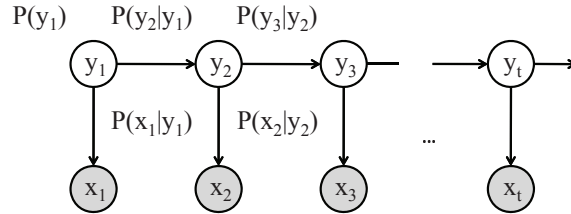


Figure 2.6: Hidden Markov Model.

For an HMM $\theta = (\lambda, A, \pi)$, the radio map $\lambda = \{P(\mathbf{x}_i | y_i)\}$ models the signal distribution, e.g. Gaussian distribution $P(\mathbf{x} | y) = \frac{1}{(2\pi)^{k/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \Sigma (\mathbf{x} - \boldsymbol{\mu})}$ at each location, where $\boldsymbol{\mu}$ is the mean vector of the observations, and Σ is the covariance matrix. By assuming the independence among the APs [10], we can simplify Σ as a diagonal matrix. The transition matrix

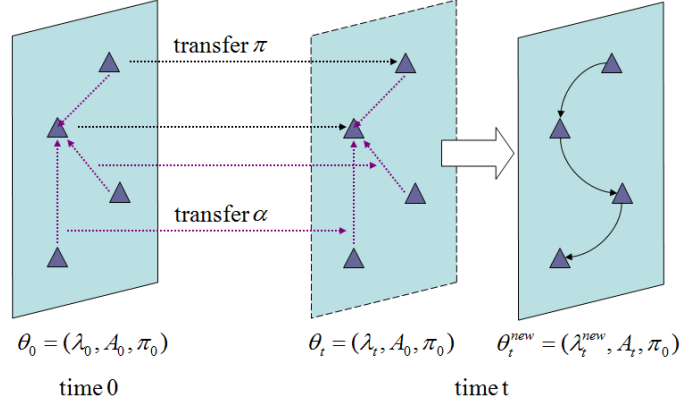


Figure 2.7: Our TrHMM to adapt a localization model from time 0 to time t .

$A = \{P(y_{i+1}|y_i)\}$ encodes the probability for a user to move from one location to another. The location prior π is set as a uniform distribution over all the locations, as a user can start from any location. It can be revised according to some domain knowledge.

Our transferred Hidden Markov Model has three stages. As shown in Figure 2.7, the triangles denote reference point locations in the area. First, we learn the signal correlation model α from the auxiliary time period 0 like the ModelTree method did [82]. Then, we apply it to the target time period t and re-estimate the radio map using some up-to-date signal data $D_{tar}^{\in L}$. Finally, we update the model, whose location prior π and transition matrix A are shared from the auxiliary time 0, by using the trace data T_{tar} .

Applying regression analysis at time 0 Our first step is to transfer the collected radio map λ_0 at time 0 to another time t . To do this, we propose to model the signal variation with time. Note that in a wireless indoor environment, signals over different locations are correlated, e.g. when the signal on a location l_1 increases with time, the signal on its neighboring location l_2 is also likely to increase. Motivated by this observation, we apply a regression model to learn the temporal predictive correlations between the RSS values by sparsely located reference points and that by the mobile device. Specifically, we apply a *Multiple Linear Regression* model on the data at time 0 over all the location grids, and derive the regression coefficients $\alpha^k = \{\alpha_{ij}^k\}$, which encode the signal correlations between reference point locations $\{l_c\}$ and one non-reference point location k . Specifically, we have:

$$s_j^k = \alpha_{0j}^k + \alpha_{1j}^k r_{1j} + \dots + \alpha_{nj}^k r_{nj} + \epsilon_j \quad (2.6)$$

where s_j^k is the signal strength received by the mobile device at location k from j^{th} AP, $\alpha_{ij}^k : 1 \leq i \leq n$ is the regression weights for j^{th} AP signal at location k , and $r_{ij} : 1 \leq i \leq n$ is the signal strength received by i^{th} reference point location from j^{th} AP.

Rebuilding the radio map at time t After learning the regression weight α^k for each non-reference point location k , we can use them to rebuild the radio map at time t . This is done by updating non-reference point locations' signal strengths with the newly collected signal strengths on the reference point locations $\{l_c\}$, whose locations are known beforehand. Considering that there may be a possible shift for the regression parameters over time, we add a tradeoff constraint to derive the new λ_t :

$$\begin{aligned} \mu_t &= \beta \cdot \mu_0 + (1 - \beta) \cdot \mu_t^{reg}, \\ \Sigma_t &= \beta \cdot \left[\Sigma_0 + (\mu_t - \mu_0)(\mu_t - \mu_0)^T \right] + (1 - \beta) \cdot \left[\Sigma_t^{reg} + (\mu_t - \mu_t^{reg})(\mu_t - \mu_t^{reg})^T \right] \end{aligned} \quad (2.7)$$

where μ is the mean vector of Gaussian output for each location, Σ is the covariance matrix. We balance the regressed radio map $\lambda_t^{reg} = (\mu_t^{reg}, \Sigma_t^{reg})$ and the base radio map $\lambda_0 = (\mu_0, \Sigma_0)$ by introducing a parameter $\beta \in [0, 1]$.

Using EM on unlabeled traces at time t In the previous steps, we first trained an HMM $\theta_0 = (\lambda_0, A_0, \pi_0)$ at time 0 as the base model. Then, in another time t , we improve λ_0 by applying the regression analysis, and obtain a new HMM $\theta_t = (\lambda_t, A_0, \pi_0)$. Now we will try to further incorporate some unlabeled trace data to improve the derived θ_t . We achieve this by applying the *expectation-maximization* (EM) algorithm.

Given a set of unlabeled traces $T = \{tr_i | i = 1, \dots, |T|\}$ where tr_i is a sequence of RSS observations on a trace i , EM is used to adjust the model parameters $\theta_t = (\lambda_t, A_0, \pi_0)$ iteratively to find a θ^* such that the likelihood $P(T|\theta^*)$ is maximized. In this way, the model θ_t is adapted to best fit the up-to-date unlabeled trace data, so that the HMM can be more accurate for the new time period t 's data. Let us denote q_i as the i -th trace's locations. The EM algorithm has two steps in each iteration: an Expectation step (E-step) and a Maximization step (M-step). For the k -iteration, in the E-step, we calculate the conditional probability $P(q|tr, \theta^k)$, *i.e.* location estimations q given the RSS observations tr , from the unlabeled trace data T by using the θ^k from last iteration's M-step:

$$P(q|tr, \theta^k) = \frac{P(tr, q|\theta^k)}{P(tr|\theta^k)} = \frac{P(tr|q, \theta^k)P(q|\theta^k)}{\sum_q P(tr|q, \theta^k)P(q|\theta^k)} \quad (2.8)$$

where $P(tr|q, \theta^k) = \prod_{n=1}^{|tr|} P(\mathbf{x}_n|y_n, \theta^k)$ is the likelihood of observing a trace tr given the mobile device's location sequence is q . Notice that this term can be calculated from the last M-step

k 's radio map λ^k , since λ^k already encodes the conditional probability of $P(\mathbf{x}_n|y_n)$. The term $P(q|\theta^k) = P(y_1|\theta^k) \times \prod_{n=1}^{|tr|} P(y_n|y_{n-1}, \theta^k)$ is the probability of q being location sequence in a user trace. This can be calculated from prior knowledge π_0 and transition matrix A^k , because π_0 encodes the probabilities of $P(y_n)$ and A^k encodes the conditional probabilities $P(y_n|y_{n-1})$ for different n 's. In the M-step, an expected loglikelihood (*i.e.* Q -function) is maximized over the parameter θ based on the E-step. The parameter θ^k is then updated to obtain θ^{k+1} :

$$\begin{aligned}\theta^{k+1} &= \arg \max_{\theta} Q(\theta, \theta^k) \\ &= \arg \max_{\theta} \sum_{tr \in T} \sum_q P(q|tr, \theta^k) \log P(tr, q|\theta)\end{aligned}\tag{2.9}$$

In particular, by following the derivation of [88], we show the update for each parameter in $\theta^{k+1} = (\lambda^{k+1}, A^{k+1}, \pi)$. Note that since π is the prior knowledge of the user locations, it's set to be fixed and not involved in EM. Specifically, the radio map $\lambda^{k+1} = \{P(\mathbf{x}_j|y_i)^{(k+1)}, \forall i, j\}$ is shown to be updated by:

$$\begin{aligned}\boldsymbol{\mu}_y^{(k+1)} &= \frac{\sum_{tr \in T} \sum_{n=1}^{|tr|} \mathbf{x}_n P(y_n = y|tr, \theta^k)}{\sum_{tr \in T} \sum_{n=1}^{|tr|} P(y_n = y|tr, \theta^k)} \\ \Sigma_y^{(k+1)} &= \frac{\sum_{tr \in T} \sum_{n=1}^{|tr|} (\mathbf{x}_n - \boldsymbol{\mu}_y)(\mathbf{x}_n - \boldsymbol{\mu}_y)^T P(y_n = y|tr, \theta^k)}{\sum_{tr \in T} \sum_{n=1}^{|tr|} P(y_n = y|tr, \theta^k)}\end{aligned}\tag{2.10}$$

And the transition matrix $A^{k+1} = \{P(l_j|l_i)^{(k+1)}, \forall i, j\}$ is updated as:

$$P(y'|y)^{(k+1)} = \frac{\sum_{tr \in T} \sum_{n=1}^{|tr|-1} P(y_n = y, y_{n+1} = y'|tr, \theta^k)}{\sum_{tr \in T} \sum_{n=1}^{|tr|-1} P(y_n = y|tr, \theta^k)}\tag{2.11}$$

The EM algorithm guarantees the likelihood $P(T|\theta^{k+1}) \geq P(T|\theta^k)$ and the parameter θ eventually converges to a stable θ^* . The new HMM is finally updated as $\theta_t^{new} = (\lambda_t^{new}, A_t^{new}, \pi_0)$. In the online phase at time t , the derived parameters in θ_t^{new} are used to infer the most probable location sequence for the queried trace based on Viterbi algorithm [88], given the obtained RSS values. We summarize our TrHMM in Algorithm 2.

2.4.2 Empirical Study

In this section, we empirically study the performance of our transferred HMM in addressing the time-dependent data sparsity challenge. Our experiments were set up in an academic building

Algorithm 2 Algorithm of Transferred Hidden Markov Model (TrHMM)

Input: Labeled traces at time 0, labeled RSS samples collected from reference points and unlabeled traces at time t

Output: Adapted model $\theta_t^{new} = (\lambda_t^{new}, A_t^{new}, \pi_0)$.

- 1: **At time 0,**
 - 2: Build an HMM base model $\theta_0 = (\lambda_0, A_0, \pi_0)$ using labeled traces from time 0;
 - 3: Learn the signal regression weights α^k among referent points and the rest, using labeled trace data from time 0;
 - 4: **At time t ,**
 - 5: Rebuild the radio map using the labeled RSS samples collected from reference points at time t , and update the model to $\theta_t = (\lambda_t, A_0, \pi_0)$;
 - 6: Apply EM to improve θ_t as $\theta_t^{new} = (\lambda_t^{new}, A_t^{new}, \pi_0)$, by using unlabeled traces from time t ;
 - 7: Return the HMM model $\theta_t^{new} = (\lambda_t^{new}, A_t^{new}, \pi_0)$.
-

equipped with 802.11b/g/n wireless network. The area is $64m \times 50m$, including five hallways. It's discretized into a space of 118 grids, each measuring $1.5m \times 1.5m$. Our experimental evaluation method is based on classification accuracy, which is calculated as the percentage of correct predictions over all predictions. The problem is difficult because a random guess would result in less than 1% in accuracy.

We collected calibration data over three time periods: 08:26am, 04:21pm and 07:10pm. We use 08:26am data to build the base model (i.e. auxiliary time period) and carry out adaptation on other time periods (i.e. target time periods). 60 samples were collected at each grid. We randomly splitted 2/3 of the data as training and 1/3 as testing. Traces for building HMM were also collected at each time period. For training, we have 30 labeled traces for 08:26am data, each having 20 samples on average. In the remaining time periods, we obtained 30 unlabeled traces, each having 250 samples for training. For testing, we have 20 traces for each time period, each having 250 samples on average. In the experiments, β is set as 0.4. We tested different β -values, and found that for different time periods, the performances with different β -values are similar, and $\beta = 0.4$ gives roughly the best results. This coincides with our intuition of allocating around half the weight to the regressed radio map and old radio map in Eq.(2.7).

Impact of distribution variation We test the localization accuracy over different data distributions without adaptation. We use the 08:26am dataset to build the base model θ_0 , and then apply

θ_0 to predict the labels for test data traces of the three time periods. We use 10% of locations as reference points and 5 unlabeled traces for adaptation. As shown in Figure 2.8, the localization accuracy of 08:26am data is the highest, at 92%⁶. This high accuracy is due to the fact that the test data follow the same distribution with the training data. As time goes by, the signals become more noisy and changing, and the performance drops. At 04:21pm, the busiest time in the work area, the noise level reaches the highest because of many people walking around at that time. During this period, the accuracy thus drops to the lowest point to about 68%, which is unsatisfactory. This observation implies a need for transferring the localization model over different data distributions.

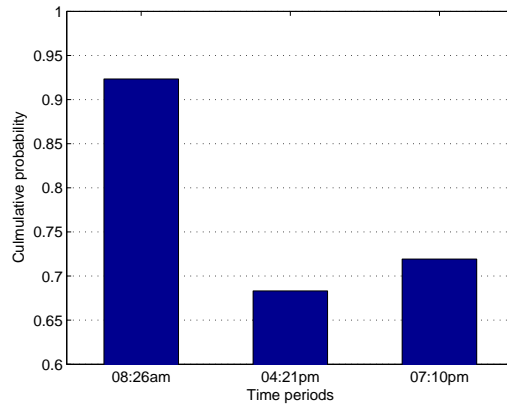
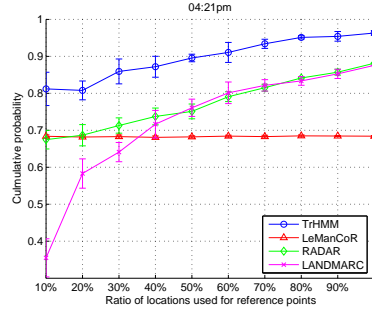


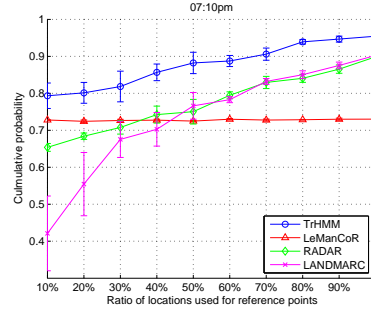
Figure 2.8: Use 08:26am model to predict others.

Impact of reference points We next study the impact of using different number of reference points, *i.e.* different amount of labeled data, for adaptation. We compare TrHMM with three baselines using reference points for adaptation, including RADAR [9], LANDMARC [11] and LeManCoR [83]. RADAR is a K-nearest-neighbor method. The number of nearest neighbors is set to be five, as tested in [9]. LANDMARC is a nearest-neighbor weighting based method. In this experiment, the number of nearest neighbors is set as two, since only few reference points are sparsely deployed in the environment. LeManCoR is a semi-supervised manifold method. It treats different time as multiple views and uses a multi-view learning framework to constraint the predictions on reference points to be consistent. In this experiment, we use five unlabeled traces for both TrHMM and LeManCoR. We run the experiments for five times and report the error bar charts. As shown in Figures 2.9(a) and 2.9(b), our TrHMM consistently outperforms the other methods over time, especially when the number of reference points is small. For RADAR

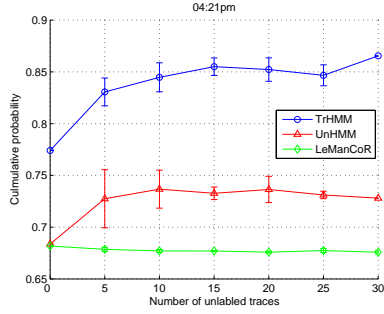
⁶The error distance is set to be three meters, which means the predictions within three meters of the true location are all counted as correct predictions.



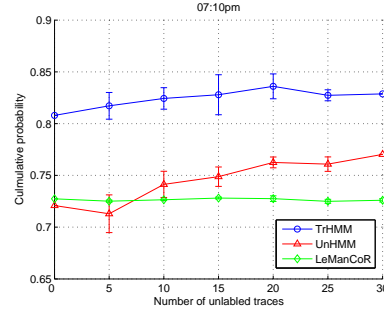
(a) Impact of reference points for 04:21pm



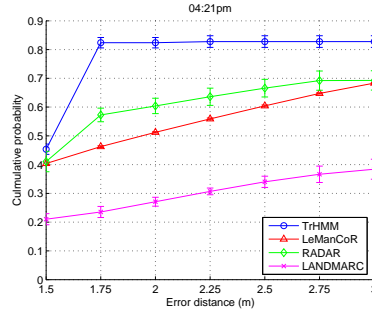
(b) Impact of reference points for 07:10pm



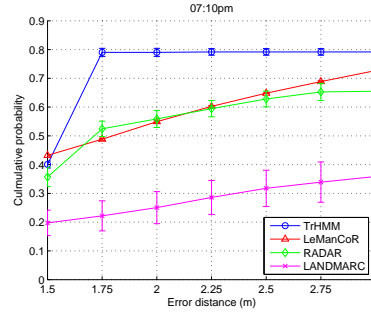
(c) Impact of unlabeled data for 04:21pm



(d) Impact of unlabeled data for 07:10pm



(e) Sensitivity to error distance for 04:21pm



(f) Sensitivity to error distance for 07:10pm

Figure 2.9: Experimental results for TrHMM

and LANDMARC, they work well only when the environment is densely installed with reference points. For LeManCoR, it cannot benefit from the trace sequential information and the increasing number of reference points (as shown in the experiments of [83]), so our TrHMM method can outperform it consistently.

Impact of unlabeled data We also study the impact of using different number of unlabeled traces for adaptation. We compare TrHMM with two baseline methods. The first method is unsupervised HMM (UnHMM) [63], which only uses unlabeled traces. UnHMM builds a base model θ_0 , and then directly use EM to iteratively update the model θ_0 using the unlabeled traces. The second method is LeManCoR, which uses unlabeled data for manifold regularization in adaptation. We use 10% of locations as reference points for both TrHMM and LeManCoR. We run the experiments for five times and report the error bar chart. As shown in Figures 2.9(c) and 2.9(d), our TrHMM consistently outperforms the baselines. Our TrHMM can outperform UnHMM because TrHMM carefully models the signal variation over time while UnHMM not. Our method also outperforms LeManCoR because TrHMM better models the signal variations and also uses the sequential information in traces.

Sensitivity to error distance We study the sensitivity of TrHMM to different error distances. In experiment, 10% locations were used as reference points for collecting labeled data and five traces were used for unlabeled data. We run the experiments for five times and report the error bar charts. As shown in Figures 2.9(e) and 2.9(f), our TrHMM method is insensitive to the error distance in calculating the localization accuracy⁷. In addition, compared to other localization methods, our TrHMM can work much better even when the error distance is small. Both of these observations testifies our TrHMM method can provide accurate location estimation with small calibration effort.

2.5 Summary

In this chapter, we introduce two context-dependent data sparsity challenges, including the WiFi signal change over different mobile devices and different time periods. We provide two transfer learning solutions that make use of the auxiliary device or time period’s data to help reduce the calibration effort. For the device-dependent data sparsity problem, we propose a latent multi-task

⁷Note that the predictions of HMM are discrete hidden variables, which in our case are $1.5m \times 1.5m$ grids, so the accuracy is stable from 1.75m to 3m.

learning (LatentMTL) algorithm which models different devices as different tasks and enforces them to share some model knowledge in learning. We formulate our method as an optimization problem with convex objective function and bilinear constraints, and employ an alternating optimization approach to solve it. The contribution of our work is that, we only require that the learned hypotheses (i.e. the localization models) to be similar in a latent space, which broadens the application range of multi-task learning. For the time-dependent data sparsity challenge, we propose a transferred hidden Markov model (TrHMM) which combines a hidden markov model and a multi-linear regression function to address the signal variations over time. Our contribution is to exploit the signal correlations and the sequential information at the same time for solving the time-dependent data sparsity challenge. We both evaluated our LatentMTL algorithm and TrHMM algorithm on the real-world data sets, and compared them with competing baselines. We show that our proposed algorithms can successfully address the distribution differences of the auxiliary data in knowledge transfer, and are able to outperform the baselines.

In the future, we plan to extend our LatentMTL algorithm to incorporate the unlabeled data in training, and also exploit the kernel trick to improve our feature mapping function from a linear form to a nonlinear form. We also want to study the online learning strategy for our TrHMM algorithm to utilize the new trace data in a real time manner. Besides, we will consider how to optimally place the reference points for better signal correlation analysis.

Chapter 3

Sensor-based Activity Recognition

With the proliferation of sensor technologies, the task of recognizing human's activities from a series of low-level sensor readings has drawn many interests in AI and ubiquitous computing areas. Early activity recognition algorithms are based on logic. They generally keep track of all logically consistent explanations of the observed actions, and use them to infer the possible plans or goals. Kautz provided a formal theory of plan recognition [24], where he described plan recognition as a logical inference process of circumscription. The recognizer's knowledge is represented by a set of first-order statements in an event hierarchy. Given the definitions on abstraction, decomposition and functional relationships between different types of events, one can use first-order logic to infer the goal from a series of events. Because the reasoning process needs to keep track of all possible consistent plans, the proposed plan recognition framework can have an exponential time complexity in worst case w.r.t. the input hierarchy size. Lesh and Etzioni further improved Kautz's plan recognition system with higher efficiency in reasoning [89]. Specifically, they introduced compact representations for goal recognition on large plan libraries, and repeatedly pruned inconsistent plans from time to time as new actions arrive.

As the sensor data became available, the user action and goal are generally represented with some sensor observations which are possibly noisy in nature. Because the traditional logic-based approaches are limited in modeling such data uncertainty and learning patterns from the data, recent activity recognition research starts to focus more on learning methods [25, 5].

3.1 Background of Learning-based Activity Recognition

3.1.1 Data Input and Output

For the sensor-based activity recognition, the data inputs include the sensor data obtained from some sensor environment and optionally some activity labels indicating what kind of activities the user is doing. Consider an activity recognition problem: in the environment, there are m sensors, from which we collect the signal data. By appropriately segmenting the sensor data sequence, we have a sensor data vector as $\mathbf{x} = (x_1, \dots, x_d)' \in \mathbb{R}^d$, and its activity label as $y \in \{a_1, a_2, \dots, a_k\}$ where each a_i can indicate an activity like “making coffee”. Finally, an activity recognition model is a function $f : \mathbf{x} \rightarrow y$.

Input and Output. In training, we have a sequence of labeled sensor data vectors $D_{trn}^{(l)} = \{(\mathbf{x}_t, \mathbf{y}_t) | t = 1, \dots, n_l\}$, together with some possible unlabeled data sequence $D_{trn}^{(u)} = \{\mathbf{x}_t | t = 1, \dots, n_u\}$ with $D_{trn}^{(u)} \cap D_{trn}^{(l)} = \emptyset$, as *inputs*. They are used to learn the activity recognition model. At an online phase, given some sensor data sequence $D_{tst} = \{\mathbf{x}_t | t = 1, \dots, n_t\}$, one can use the trained activity recognition model to predict the corresponding activities of the user as *output*. We call this process as activity recognition from the sensor data. It’s generally assumed that D_{trn} is sufficient for each user and/or each activity to train the model. We will later show that, such an assumption may not hold in practice, leading to the user-dependent data sparsity and activity-dependent data sparsity challenges.

In order to obtain the above inputs and outputs, one need to process the data from their raw sensor data formats and annotate them with carefully selected activity labels as below.

Raw sensor data format. In general, the sensor data we obtained from the environment are in the form of sensor event sequences. Each sensor event is generally triggered by a happening activity (and sometimes by noise). For example, consider the state-change sensors used in the home setting activity recognition research [90, 2]. These state-change sensors can be affixed to the objects in the environments such as kitchen cabinet door, TV remote control, etc. When an object is moved by the user, for instance, when she is trying to get some coffee beans out of the cabinet and thus open the cabinet door, a sensor event is triggered. Such a sensor event can be captured and some sensor reading value is recorded. Notice that, such a sensor event indicates one activity that the user is possibly trying to making coffee. Together with some other successive sensor events such

as user touching the coffee machine, preparing sugar and milk for coffee drinking, we can then finally infer that the user is doing the activity of “making coffee”.

Sensor data processing. One general way to process a sensor data sequence for further training is to segment it with a constant time interval. Each time interval is also referred to as a time slice. Consequently, the data observations within a time slice are combined to generate a raw feature vector, with each feature denoting one sensor’s reading within the time slice. Consider an example of three state-change sensors, denoted as s_1 , s_2 and s_3 . Given a time slice, for example of one minute, if sensor s_1 fires with reading “1” and the other two sensors are not triggered with readings “0”, then we can generate a raw sensor feature vector of $(s_1, s_2, s_3) = (1, 0, 0)$. Such a feature vector can be possibly mapped to an activity such as “making coffee”, if s_1 refers to a sensor affixed to the coffee machine. Sometimes, feature transformation is employed for data processing if necessary [91]. One can expect that the time interval length needs to be appropriately set, so that a time slice can capture enough sensor signals for recognition and meanwhile be not too long to mix all the activities. In some case, the battery issue also needs to be considered when setting the interval length [90]. There is also some work that considers varying length w.r.t. different activity durations [90]. But in general, the activity duration can also be modeled with fixed length time segments. Therefore, in this thesis, we only focus on handling data sequences with fixed length time segments.

Activity selection. Generally, the research community aims to recognize Activities of Daily Living (ADLs), which refer to daily self-care activities within an individual’s place of residence, in outdoor environments, or both. Examples of ADLs include bathing, dressing, grooming, work, homemaking, leisure, etc. Health professionals routinely refer to the ability or inability to perform ADLs as a measurement of the functional status of a person, particularly in regards to people with disabilities and the elderly¹. However, in the real-world research, the activities to recognize can vary with different environment settings and different applications. For example, Intille et al. focused more on home-setting activity recognition, and they defined a set of 89 activities that covered common activities, user mobility, postures and locations in a home environment [2]. Liao et al. focused on the outdoor activity recognition, and they defined several activities including work, home and visiting friend, etc. [92]. Generally, the selected activity should be meaningful to the desired applications, and recognizable given the deployed sensors.

¹http://en.wikipedia.org/wiki/Activities_of_daily_living

Activity annotation. Given a set of selected activities to recognize, one may need to annotate the sensor sequence data along the time axis with respect to some ground truth. There are usually three data annotation strategies: direct observation by researchers, subject self-report of activities, and experience sampling [93]. Each strategy has its own disadvantage. The direct observation strategy can be costly and scales poorly for the study of large subject populations. The subject self-report recall surveys are prone to recall errors and lack the temporal precision [94]. Finally, the experience sampling method requires frequent interruption of subject activity, which agitates subjects over an extended period of time.

After sensor data annotation, we can possibly have some labels for the sensor data. Notice that there are cases with more than one activity happening at the same time. Such concurrent activity phenomenon was studied in a series of previous work, in both psychology [95, 96] and computer science research [97, 98, 99]. We also discussed this problem in our previous work [100] by considering the concurrent and interleaving activities together. In this thesis, we do not discuss this multi-activity phenomenon, and only focus on the case that at each time slice there is at most one activity happening.

3.1.2 Existing Learning Models

Once we have the sensor data, as well as some possible activity labels, as inputs, we can apply various machine learning algorithms to train the activity recognition model and thus use it for further online prediction. There have been many learning algorithms proposed for activity recognition, and based on their modeling capabilities, we can generally categorize them into two types: one is *non-sequential* model, and the other is *sequential* model.

Non-sequential Model. This category of learning models does not take the sequence information into account. Instead, they treat each time slice’s sensor signals as a data vector. Then, they either classify each data vector as one of the predefined activity labels or discover some activity routines by summarizing the data in an unsupervised fashion. Based on whether the learning models use labels in training or not, we can further categorize these non-sequential models into two types as follows.

- **With labels.** Typical non-sequential models using label information include naive Bayes

[93], decision tree [101], support vector machine [102], etc. For example, Logan et al. collected the signals from various sensors (including wired switches, RFID tags, wireless object usage detections, electrical current and water flow detectors, etc.) installed in a home environment, and then organized them into data vectors by segmenting the sensor signal sequence. Finally, they tested both naive Bayes and decision tree on the data vectors and predict the activity labels [101]. Similarly, Bao and Intille extracted the FFT (Fast Fourier Transformation) features from the acceleration signal data from some accelerometer sensors attached to the user's body, and then used both naive Bayes and decision tree for activity recognition [93]. Bulling et al. derived eye movement features such as saccades, fixations and blinks from the electrooculographic readings, and then they applied multi-class SVM to predict the activities [102]. Guan et al. employed a co-training style semi-supervised learning model for activity recognition [103]. In particular, they first trained three base classifiers such as decision tree, naive Bayes and K-nearest-neighbor, on the same labeled training data set. Then, they used these three base classifiers to predict the labels of unlabeled training data through majority voting, and picked some of them to be merged into the labeled training data. The above process iterates and finally outputs an ensemble classifier given the three adapted base classifiers for activity recognition. In this way, they utilized unlabeled data in training and showed to improve the classification performance.

- **Without labels.** There are also some research studies claiming that they can discover the activity routines in an unsupervised manner. Typical learning models in this category include Principal Component Analysis (PCA) [8], Latent Dirichlet Analysis (LDA) [104, 105], etc. One should note that, these models are generally used to find the activity routines rather than to recognize activities in the real time. These models can be applied to both annotated activity data and directly the sensor data. For example, Eagle and Pentland collected 100 MIT students' location data, and thus further used them to construct a binary date-vs-(location \times time-of-day) data matrix for each student [8]. Such a data matrix indicates that given some date and some time moment (e.g. 2pm of the day), where the student subject was. Then, they applied PCA on the data matrix of each student, and discovered some activity routines, or so called eigenbehavior. Such an eigenbehavior can be a typical weekday activity pattern with midnight to 9:00 at home, 10:00 to 20:00 at work, and returning home again at 21:00; or a typical weekend pattern with still remaining at home after 10:00 and going to downtown in the evening. Each day of the subject's location history can be seen as a (linear) combination of such eigenbehaviors. Eagle and Pentland further applied PCA to the bluetooth proximity

data and showed interesting social eigenbehavior patterns. Farrahi and Gatica-Perez applied another unsupervised model LDA to model the activity routines from the MIT location data set [105]. In particular, they divided a day into 30-minute timeslots and assigned them with the single location labels which occurred for the longest duration in each time slot. Here, the location labels include home, work, other and no-reception. Consequently, they treated each day's location data as a document, which consists of several *location words*. Each location word is a four-dimensional vector, containing three consecutive locations within a coarse time slot as well as the time slot ID. Then, an LDA was applied to find the activity routines, or so-called topics, from the data. Each day can be seen as a mixture of the topics. Compared with PCA, LDA is a probabilistic model and is claimed to be beneficial to data clustering [105]. Farrahi and Gatica-Perez further encoded the user information and applied Author Topic Model to discover the user-specific activity routines. Similarly, LDA was also employed to find the activity patterns on the accelerometer data [104].

Sequential Model. Another big category of activity recognition models uses the sequential information, including Hidden Markov Model (HMM) [5], Hidden Semi-Markov Model (HSMM) [106], Dynamic Bayesian Network (DBN) [107, 5] and its special case of Coupled HMM (CHMM) [108], Conditional Random Fields (CRF) [109] and its variants: Semi-Markov Conditional Random Fields (SMCRF) [110, 111], Factorial CRF (FCRF) [98, 112], hierarchical CRF [113], etc.

For *Hidden Markov Model*, the states are activities and the observations at each time slice are sensor feature vectors. There are generally two types of implementations for activity recognition. As discussed in [5], the first implementation can be building a set of one-state HMM for each activity and further using the HMM with the highest likelihood score to estimate the activity. The second implementation can be building one multi-state HMM which directly encodes each activity as one state in the model. Compared with the first implementation, the second implementation captures the inter-activity transitions and thus is argued to be better in practice [5].

For *Conditional Random Fields*, similarly the states are activities and the observations at each time slice are sensor feature vectors. Different from HMM which is a generative model, CRF is a discriminative model and is generally shown to have better performance than HMM in classification [109].

HMM and CRF are two basic sequential models in activity recognition and there are various extensions on them as listed below.

For *Hidden Semi-Markov Models* and *Semi-Markov Conditional Random Fields*, they extend the standard HMM and CRF respectively to formulate the state duration (i.e. activity duration) in the models. Activity duration can be a useful feature in activity recognition, because in practice different activities can last for different time scales. For example, having dinner usually takes more than 30 minutes while brushing teeth may only take a few minutes. HSMM is capable of encoding such activity durations in the model [106]. Different from standard HMM, HSMM formulates a joint probability of observations, states and each state's duration [114]:

$$p(y_{1:T}, \mathbf{x}_{1:T}, d_{1:T}) = \prod_{t=1}^T p(\mathbf{x}_t|y_t)p(y_t|y_{t-1}, d_{t-1})p(d_t|d_{t-1}, y_t),$$

where d_t denotes the remaining duration of a state, y denotes the state (i.e. activity), and \mathbf{x} denotes the observation feature vector. As shown in the above equation, HSMM predicts a state y_t not only based on the observations \mathbf{x}_t and the previous state y_{t-1} , but also the activity duration d_{t-1} . After EM-style parameter learning, an adapted Viterbi algorithm is employed for inference. Similar case exists for SMCRF, which formulates a conditional probability of the states and their durations given the observations:

$$p(y_{1:T}, d_{1:T}|\mathbf{x}_{1:T}) = \frac{1}{Z(\mathbf{x}_{1:T})} \prod_{t=1}^T \exp \sum_{k=1}^K \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t, d_t, d_{t-1}),$$

where $f_k(\cdot)$ denotes a feature function over the observations, states and their durations. $Z(\cdot)$ is a normalization function. As can be seen, SMCRF also takes the activity duration into account in state estimation. Its parameter learning and inference are similar to those of standard CRF model. Kasteren et al. tested HSMM and SMCRF in the real world sensor data [114]. They showed that, firstly, these semi-markov models generally perform better than standard HMM and CRF; secondly, in most cases, HSMM and SMCRF have comparable performances, but SMCRF usually takes significantly longer time in training.

For *Coupled Hidden Markov Models* and *Factorial Conditional Random Fields*, they extend the standard HMM and CRF respectively to encourage the interactions of multiple co-temporal HMM/CRF chains. In this way, they can formulate the dependency between multiple concurrent activities. CHMM model assumes that each state is conditionally dependent on all the states of component HMM chains in the previous time slice. In particular, it formulate the following joint probability for a coupled C -chain HMM [115]:

$$p(y_{1:T}^{\{C\}}, \mathbf{x}_{1:T}^{\{C\}}) = \prod_{c=1}^C \left(\prod_{t=1}^T p(\mathbf{x}_t^c|y_t^c) \prod_{d=1}^C p(y_t^c|y_{t-1}^d) \right).$$

As can be seen, the CHMM has the ability to capture the state interactions among C HMM chains by modeling $\prod_{d=1}^C p(y_t^d | y_{t-1}^d)$. Such capability can be used to predict the concurrent activities such as chatting among multiple users [112]. Similarly, FCRF can also model the interaction among different component CRF chains, but in a discriminative way:

$$p(y_{1:T}^{\{C\}} | \mathbf{x}_{1:T}^{\{C\}}) = \frac{1}{Z(\mathbf{x}_{1:T}^{\{C\}})} \prod_{c=1}^C \left(\prod_{t=1}^T \exp \sum_{k=1}^K \lambda_k f_k(y_t^c, y_{t-1}^1, \dots, y_{t-1}^C \mathbf{x}_t^c) \right),$$

where the feature function $f(\cdot)$ can describe the dependency between y_t^c and all the previous states $\{y_{t-1}^1, \dots, y_{t-1}^C\}$ even from other chains. Various feature functions can be designed according to the domain characteristics [98, 112]. The parameter learning and inference for CHMM and FCRF are adapted from the standard HMM and CRF.

For *Dynamic Bayesian Network* and *Hierarchical Conditional Random Fields*, they extend the standard HMM and CRF respectively to encode more general relations by introducing additional variables. As the models in this category are more flexible, it is hard to provide a general probabilistic framework to subsume all the cases. We give some examples using DBN and hierarchical CRF to achieve activity recognition instead. For instance, Patterson et al. extended HMM by introducing some new random variables which indicated the sensor objects' aggregate information to form a DBN [5]. Therefore, at each time slice's activity prediction, one needs to consider not only the direct sensor observations but also some aggregate features. Yin et al. designed a DBN to infer a user's goals from raw sensor signals in the indoor environment [107]. In their model, a user's goal depends on her actions, which further depend on her locations predicted from WiFi signals. All of these composes a hierarchical dynamic Bayesian network, as shown in Figure 3.1(a). Liao et al. proposed a hierarchical CRF model to extract a persons activities and significant places from traces of GPS data [113]. As shown in Figure 3.1(b), each activity is associated with some observed local evidences comprising information such as GPS location, time of day, duration and velocity; meanwhile, each place is associated with a set of activities that are performed there. A hierarchical CRF model can be employed to formulate these dependencies for learning and inference.

Some further variants of the HMM and CRF models also exist; for example, Natarajan and Nevatia extended the coupled HMM with semi-Markov assumption for activity recognition [116]. Hu and Yang employed Skip-Chain Conditional Random Fields to formulate the long-distance dependencies among states so as to model the activity interleaving property [99].

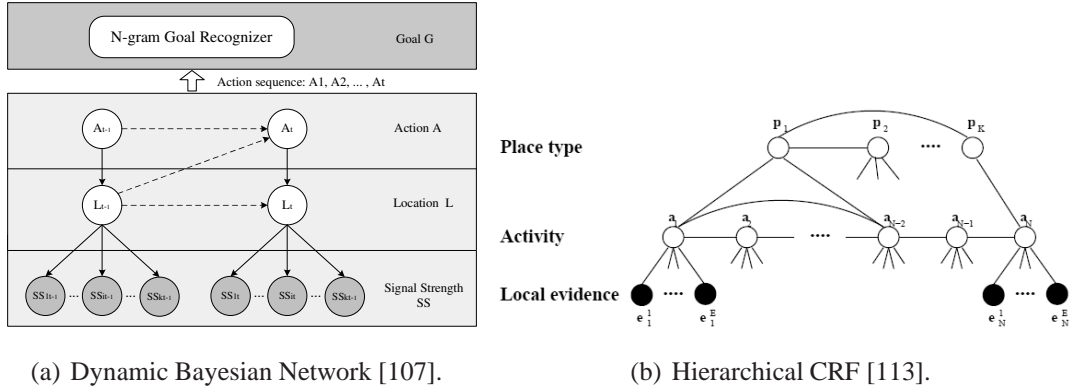


Figure 3.1: Examples of DBN and hierarchical CRF in activity recognition.

3.2 The User-dependent Data Sparsity Challenge

In general, the success of training an activity recognition model relies on having sufficient sensor data from user. However, in the real world, a single user's data are often insufficient due to the data sparsity problem. This is especially true when we are interested in obtaining a personalized activity recognition model. Consider a real-world example when we are trying to build a multi-user activity recognition system with WiFi data. Each user uses her mobile device to record WiFi signal data as she moves in the environment, and annotates the data with some activity labels such as "having class". Because the human labeling is expensive and a single user may not foresee all possible WiFi observations in such an open environment, each single user actually does not have enough annotated data to train a personalized activity recognition model for her own. Such a user-dependent data sparsity challenge is very common in mobile computing, and we will later see it exists in mobile recommendation as well.

This user-dependent data sparsity challenge is also encountered in our developing an e-health system demo system. In this e-health demo system, as shown in Figure 3.2, we are trying to utilize the ambient sensor information such as WiFi signals, GPS locations and accelerometer readings to automatically recognize a mobile phone user's activities from time to time. Then, based on the predicted activities within a time period, we provide an activity profiling function so that the user can easily perceive how much time she spent on each activity through a statistics pie chart. One intuitive application is that, we can help the users to achieve work and life balance, by tracking their work time expense. To facilitate the activity recognition module of the system, we may need the users to provide some activity annotations. However, such activity annotations, conducted in

an experience sampling way [93], require the users to annotate their activities from time to time by entering some labels on their phones. The whole process is labor intensive, and thus we observe that each project participant invited to use our demo system for around a month provides very few annotations. Such a scenario is a typical example for the user-dependent data sparsity challenge.

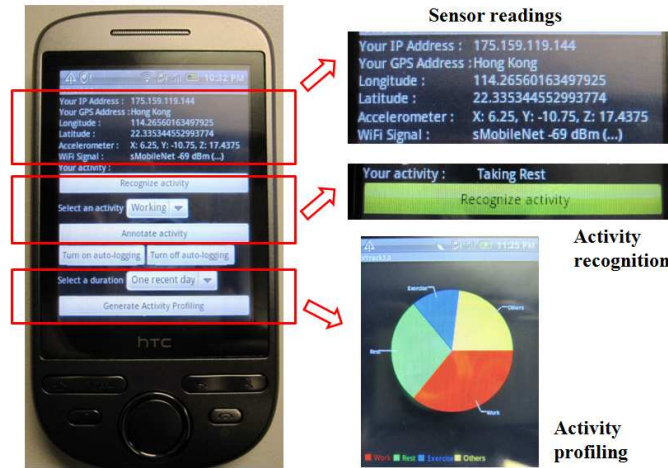


Figure 3.2: An eHealth demo based on activity recognition.

3.2.1 Related Work

In the past, to solve the data sparsity problem, people had considered using all the users' data together to train an activity recognition model. For example, Liao et al. pooled all the users' GPS trajectory data together and train a hierarchical activity model to predict some transportation routine activities [92]. Such a method may work well if there is no big difference among different users' activities. This is the case in GPS transportation activity recognition but probably not in other scenarios. For example, in WiFi-based activity recognition, a user visits the coffee shop for meal and the other just enjoys sitting in its outdoor couches to read research paper. These two users are very likely to observe similar WiFi signals, but their activities are quite personalized. There is also some work trying to model concurrent activities among multiple users. For example, Wang et al. proposed a coupled Hidden Markov Model to capture user interactions and recognize multi-user activities at the same time [117]. Lian and Hsu used a factorial Conditional Random Fields model for joint recognition of multiple concurrent chatting activities [112]. Such work requires the activities to happen at the same time by multiple users, and thus cannot handle the general activity

Problem	Algorithm	Object-level	User-level	Main drawback
User dependent	GPS activity model [92]		✓	not differentiate users
	Concurrent activity model [112, 117]		✓	require concurrency
	Common sense model [118, 119, 120]	✓		not personalized
	Sensor mapping model [91]	✓		not personalized

Table 3.1: Review on previous work of activity recognition.

recognition case. All the above algorithms try to use other users’ data for activity recognition, therefore we categorize them into “user-level” methods in Table 3.1.

Comparatively, there are also some “object-level” methods that try to enrich the sensor data from some auxiliary resources. For example, Perkowitz et al. proposed to mine the natural language descriptions of activities (e.g. “making-tea”) from the Web (such as ehow and KnowItAll [121]) as the labeled data, and translated them into probabilistic collections of object terms (e.g. “teacup”, “teabag”, etc.) [118]. Then, they used these probabilistic collections as input data to train a dynamic Bayesian network model for prediction. The follow-up work incorporated unlabeled data and additional wearable sensor data to improve the performance [120, 119]. Besides, Kasteren et al. tried to train a recognition model in a new house by using the other house’s data [91]. They figured out some sensor mappings among the sensors across the houses, so that they could transfer the auxiliary house’s data to solve the data sparsity problem. All of the above “object-level” methods need to rely on using object-usage information, which may not be available at either WiFi or GPS based activity recognition.

3.3 Collaborative Activity Recognition with Latent Aspect Model

In WiFi-based activity recognition, we propose a “user-level” method based on user-dependent aspect model for personalized predictions [122]. Rather than simply pooling multiple users’ data together, our model introduces user aspect variables to capture the user grouping information from their data. As a result, for a targeted user, the data from her similar users in the same group can also help with her personalized activity recognition. Let us define the activity data as a set of quads: $\{\langle a_i, u_i, f_i, t_i \rangle | i = 1, \dots, L\}$, where a is an activity, u is a user, and f is a feature observed at time t . In our WiFi case, a feature corresponds to a wireless access point (AP) that our mobile device

can detect in the environment. A data record quad indicates that a user u is doing an activity a at time t , and meanwhile her wireless device detects some AP f . Our goal is to build a personalized activity recognition model by utilizing these labeled data, so that with a user’s WiFi observations at some time, we can predict what she is doing. The data setting is illustrated in Figure 3.3. A matrix in the figure denotes a 2D environment and a tick indicates labeled activity data collected in that location.

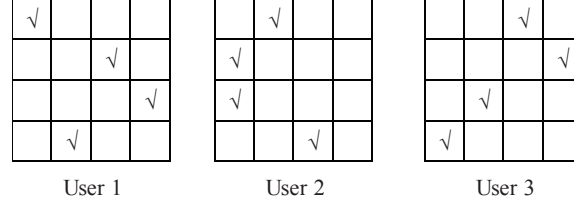


Figure 3.3: Setting for the user-dependent activity data sparsity problem.

3.3.1 Problem Formulation and Our Solution

We can extend the standard aspect model [123, 124] by introducing user aspects, as well as time aspects and feature aspects to model personalized activity recognition from time-dependent sensor data. Figure 3.4 depicts our graphical model. The shadow nodes for user variables u , time variables f , feature variables f and activity variables a are observations. The blank nodes inside the rectangle are latent aspect variables. The user latent aspects $Z_u \in \{z_u^1, z_u^2, \dots, z_u^{D_u}\}$ are discrete variables, indicating D_u user clusters. We adopt such a *user-cluster-activity* hierarchy to help the users share the knowledge. In contrast to a two-tier *user-activity* hierarchy where each user can only rely on herself to do activity recognition, our model can make the users from a same cluster to contribute together to train the recognizer from their feature and time observations. Therefore, even if some user has limited data to train an activity recognition model, she can still benefit from other similar users in the same group(s). As each user can belong to multiple user clusters at the same time with different probabilities, they actually contribute differently to each user cluster in training the recognition model and consequently get different predictions in real-time recognition.

We also introduce the latent aspects $Z_f \in \{z_f^1, z_f^2, \dots, z_f^{D_f}\}$ and $Z_t \in \{z_t^1, z_t^2, \dots, z_t^{D_t}\}$ to encode the data observations on feature and time. They are used to capture the dependency between activities and observations, considering that similar feature observations at similar time periods are likely to

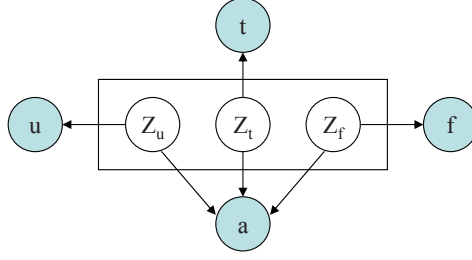


Figure 3.4: User-dependent aspect model.

imply some same activity. Note that these aspects do not necessarily rely on users. We can also take all the users' data as input, and only use them (i.e. Z_f and Z_t) to build a user-independent model for general activity recognition.

In general, our aspect model is a generative model, which uses the latent aspect variables to explain the observations. It specifies a joint probability (or data likelihood) of the observed random variables:

$$P(a, u, f, t) = \sum_{Z_u, Z_f, Z_t} P(a, u, f, t, Z_u, Z_f, Z_t), \quad (3.1)$$

where $P(a, u, f, t, Z_u, Z_f, Z_t)$ is expanded as:

$$P(a, u, f, t, Z_u, Z_f, Z_t) = P(a|Z_u, Z_f, Z_t)P(Z_u)P(u|Z_u)P(Z_f)P(f|Z_f)P(Z_t)P(t|Z_t).$$

The user variables u , feature variables f and activity variables a are all discrete in nature, so their conditional probabilities on latent aspects can be modeled easily by multi-nominal distributions. One exception is the time, which could be continuous. To formulate $P(t|Z_t)$, we discretize the time t with two possible strategies. One is “ByHour”, which segments the time into hours. The other is “ByPeriod”, which segments it into larger time periods; for example, we can define five periods, including morning (7am~11am), noon (11am~2pm), afternoon (2pm~6pm), evening (6pm~12am) and night (12am~7am). Finally, we can maximize the likelihood (or, negative loss) in Eq.(3.1) for training the personalized activity recognition model. Here, the knowledge share part is implemented by introducing Z_u for prediction, so that we can make the users from a same cluster to share their data in training the recognizer and prediction. In this instantiation, we provide a simple solution and omit the regularization terms here. We may consider to apply some hyper priors (such as a Dirichlet prior in Latent Dirichlet Allocation [125]) on the Z_u 's for regularization.

In inference, for an existing user, we want to predict her activity based on all the access point

features detected² at some time t . Therefore, our model outputs the activity a^* that has the highest likelihood:

$$a^* = \arg \max_a \prod_f P(a, u, f, t). \quad (3.2)$$

For a new user, as we do not have any of her data before, we will summarize all the users' predictions to output an activity:

$$a^* = \arg \max_a \sum_u \prod_f P(a, u, f, t). \quad (3.3)$$

Model Training

For model training, we can use the Expectation Maximization (EM) algorithm to get the maximum likelihood estimation³. At E-step, for each data example $\langle a_i, u_i, f_i, t_i \rangle$, we compute

$$P(Z_u, Z_f, Z_t | a_i, u_i, f_i, t_i) = \frac{P(a_i, u_i, f_i, t_i, Z_u, Z_f, Z_t)}{\sum_{Z_u, Z_f, Z_t} P(a_i, u_i, f_i, t_i, Z_u, Z_f, Z_t)}. \quad (3.4)$$

At M-step, given L data examples, we compute

$$P(Z_u) = \frac{1}{L} \sum_i \sum_{Z_f, Z_t} P(Z_u, Z_f, Z_t | a_i, u_i, f_i, t_i), \quad (3.5)$$

$$P(u | Z_u) = \frac{\sum_{i: u_i=u} \sum_{Z_f, Z_t} P(Z_u, Z_f, Z_t | a_i, u_i, f_i, t_i)}{L \times P(Z_u)}, \quad (3.6)$$

$$P(a | Z_u, Z_f, Z_t) = \frac{\sum_{i: a_i=a} P(Z_u, Z_f, Z_t | a_i, u_i, f_i, t_i)}{\sum_i P(Z_u, Z_f, Z_t | a_i, u_i, f_i, t_i)}. \quad (3.7)$$

Analogous to Eq.(3.5), we can get $P(Z_f)$ and $P(Z_t)$; and similarly, analogous to Eq.(3.6), we can get $P(f | Z_f)$ and $P(t | Z_t)$.

For model complexity, let's denote the number of activity as N_a , the number of users as N_u , the number of features as N_f . Together with D_u user aspects, D_f feature aspects and D_t time aspects, we totally need to maintain $N = (N_a \times D_u \times D_f \times D_t + N_u \times D_u + N_f \times D_f + N_t \times D_t + D_u + D_f + D_t)$ variables for our model. The time complexity for each EM iteration is as follows: at E-step, we need to update the probability of $P(Z_u, Z_f, Z_t | a_i, u_i, f_i, t_i)$ for each data example by summing

²We follow [107] to assume AP independence.

³Derivation details are skipped here for space, interested readers can refer to [123, 124].

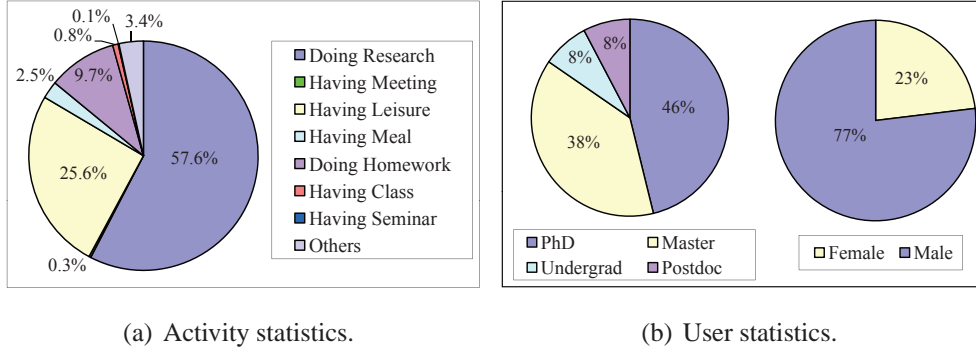


Figure 3.5: Data statistics for collaborative activity recognition.

over the latent aspects, therefore the cost is $O(L \times D_u \times D_t \times D_f)$; at M-step, we can amortize the summation over L samples as $O(L \times D_u \times D_t \times D_f)$ and after one data scan update the N variables by $O(N)$.

3.3.2 Empirical Study

In our experiments, we have 13 users collecting the WiFi data for around a month, basically in a university area. Their mobile devices sniffed the WiFi signals roughly every ten minutes when it was power on. The users annotated their WiFi data with eight possible activities from time to time, and we further preprocess the data by data segmentation and label parsing. On average, each user has around 1,150 data examples, and in total we observed 2,912 different access points (i.e. features) in this dataset. Some activity and user statistics is shown in Figure 3.5. For each user, we split the first half of her data in chronological order as training data, and the other half as test data. We measure the average accuracy among all the users at each trial, and report the average accuracy of three trials through the experiments. Without special notification, our model uses the “ByPeriod” time segmentations, and the model parameters are set as: $D_u = 3$, $D_t = 2$ and $D_f = 20$.

Activity Recognition for Existing Users In this experiment, we gather the same percent of training data from each user and fit them into our model for training. Then, we test each user and give the average accuracy. We also vary the number of training data and see the performance change. We employ three baselines for comparison: (1) “Single” baseline, which uses each single user’s data for training and maintains a personalized activity recognition model for each user; (2) “Merged” baseline, which pools all the users’ data together and trains a general activity recognition

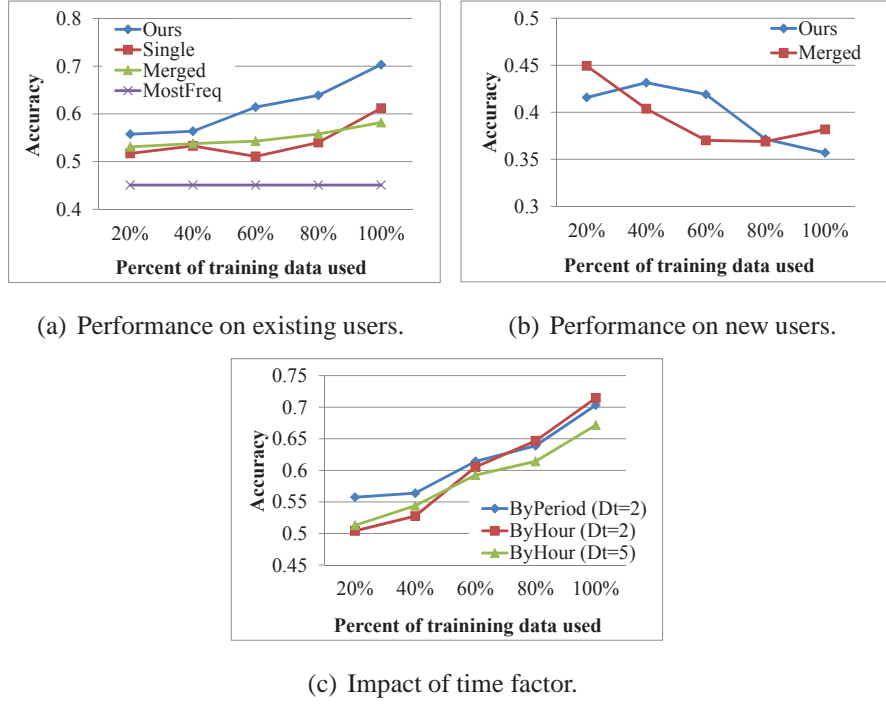


Figure 3.6: System performance for collaborative activity recognition.

model for all the users. For these two baselines, we use the sophisticated Conditional Random Fields [109, 92] as the classifier model. These two baselines also take the time information as a feature input for fair comparison; (3) “MostFreq” baseline, which uses the most frequent activity based on training data as the prediction.

As shown in Figure 3.6(a), as the number of training data increases, our model is consistently better than the baselines. The “MostFreq” baseline’s performance keeps unchanged, as the found most frequent activity is always “doing research” as training data size increases. The baseline’s comparatively poor performance shows that a simple solution may not be adequate for this complex task. For the other two baselines, we notice that, when the training data size is small, our model’s improvement over the baselines seems small as well; but as the training data size increases, the improvement increases quickly. This is because with limited training data, the learned user clusters may not be as accurate as that with enough training data. Another interesting observation is that, when there is more training data, the “Single” baseline seems to outperform the “Merged” baseline. It implies that each user’s unique activity pattern can be better preserved by personalized activity recognition given enough training data.

We also study the impact of user latent aspects to our model, so that we can understand how

Percent of training data	User-dependent (our model)	User-independent (variant of our model)
20%	0.56 ± 0.02	0.52 ± 0.01
40%	0.57 ± 0.02	0.53 ± 0.02
60%	0.62 ± 0.03	0.58 ± 0.04
80%	0.65 ± 0.01	0.60 ± 0.02
100%	0.71 ± 0.00	0.64 ± 0.03

Table 3.2: Impact of the user latent factors [acc \pm std].

much benefit this user-dependent solution brings to personalized activity recognition. We employ a variant of our model as the baseline, named user-independent aspect model for comparison. This baseline takes all the users’ data as input. It is a simplified version of our model with the user latent factor Z_u and user variable u removed. In other words, this baseline only consists of feature and time latent aspects, without encoding any user information, and thus it is user-independent. As shown in Table 3.2, our user-dependent model consistently outperforms the baseline. It proves that, the user aspects are crucial to the personalized activity recognition task. In addition, we also tested the case when this user-independent baseline only takes a single user’s data as input for training to mimic personalized activity recognition. On average, we observed a 15% performance lift of our model over such a baseline. This shows that our model can well incorporate multiple users’ data.

Activity Recognition for New Users In this experiment, we use a leave-one-user-out strategy, with which we hold out one user only for testing and the other users for training. Then we rotate on each user and report an average accuracy in Figure 3.6(b). As there is no training data for the test user, the “Single” baseline does not work anymore. We compare our model with the “Merged” baseline which requires all the users to share a same activity recognizer. As can be seen in the figure, our model’s performance is comparable with the baseline. This is because when there is no training data for a new user, our model has no way to capture her grouping information. By summing up the prediction probabilities over the users (c.f. Eq.(3.3)), our model thus gives a comparable prediction result with the “Merged” baseline of pooling all the data together.

Impact of Time and the Model Parameters We study the impact of two time segmentation strategies: “ByHour” and “ByPeriod”. As shown in Figure 3.6(c), two strategies have comparable performances, though when the training data size is small, the “ByPeriod” strategy seems better. This may be because “ByPeriod” strategy summarizes the reasonable time segments with fewer parameters, which are favored given limited training data. We also observe that, the model

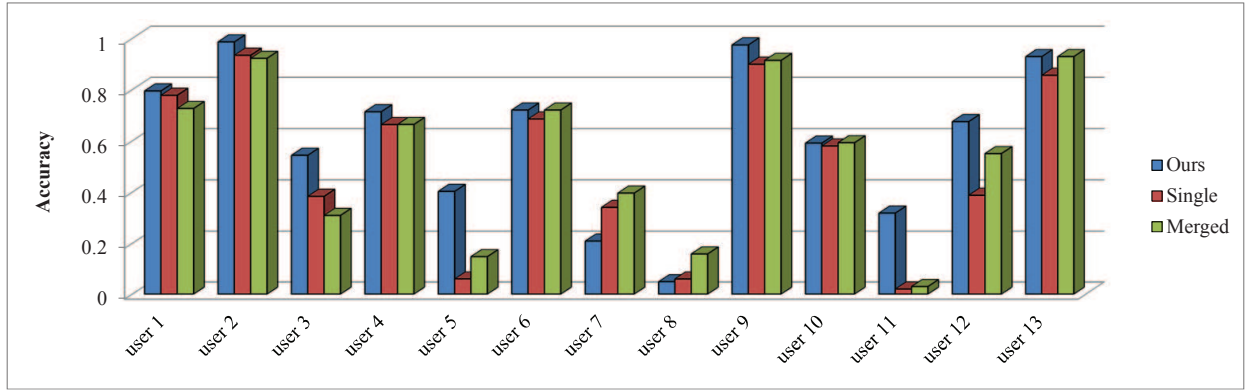


Figure 3.7: Performance on each user.

performance seems not very sensitive to the latent time aspect cardinality D_t .

We study the impact of the model parameters on user and feature group numbers. In this experiment, we use 60% of the training data. We vary the model parameter pairs (D_u, D_f) , and report the averaged accuracy in the recognition task with existing users. As shown in Table 3.3, when the user cluster numbers are reasonably small, e.g. $D_u = 3$ or 5 compared with the total user number as $N_u = 13$, the model performance is stable. But when the user cluster number is too big, e.g. $D_u = 10$, the performance could possibly drop due to inappropriate clustering. For the number of feature clusters D_f , as the number of observed APs in our data is big, we see the performances for $D_f = 20$ and $D_f = 50$ are comparable. We also observe similar patterns in recognizing new users.

<i>Parameters</i>	Accuracy
$D_u = 3, D_f = 20$	0.62 ± 0.02
$D_u = 5, D_f = 20$	0.60 ± 0.01
$D_u = 5, D_f = 50$	0.59 ± 0.02
$D_u = 10, D_f = 20$	0.57 ± 0.03

Table 3.3: Impact of the model parameters on D_u and D_f .

Discussion When our model works well and when it may not? These are the questions that we are curious about. We plot our model’s performance on each user in Figure 3.7. We used 60% of training data, and the baselines are described as above.

In the figure, we can observe our model improvements on most users over the baselines. Our model especially works well on user 3, user 5 and user 11 compared with the baselines. Let us take user 11 as an example for analysis. The “Single” baseline does not work well in this user’s case,

because in the test data, we have many access point features unseen in the training data. Therefore, the “Single” baseline, since it only uses a single user’s data for training, cannot handle well the unseen observations and give many incorrect predictions. Comparatively, our model can work better because it can benefit from other users’ data which may contain some unseen access points. Interestingly, the “Merged” baseline also does not work well in this case, though it uses other users’ data. The test data show that user 11 often went to the campus cafe area (which has cozy couches and coffee) in the evening time to do research. But the “Merged” baseline predicts the activity in the cafe area as “having meal”, as most of the users seen in the dataset annotated “having meal” there at that time. While our model correctly gives the prediction of “doing research”, because it figures out that one of his similar user (i.e. user 4) sometimes also did the same thing in the cafe. This pattern is preserved to correct the predictions as “doing research” rather than “having meal”. This case shows that, our model works when: 1) there are multiple users’ data available; and especially 2) when similar users share the similar activities given similar observations.

There are also some cases when our model may not work well, such as on user 7 and user 8. Though the reasons why our model does not work well on both users appear slightly different, there is one thing in common. That is, our model’s underlying assumption which assumes a group of similar users would do similar activities given some similar observations is violated in these two cases. For example, some of user 7’s test data indicate his “having leisure” in some office. Our model predicts the activities as “doing research” since one of his similar user usually worked in that office area. This activity inconsistency among the users leads to the performance drop. Besides, user 8 is shown to be often “having leisure” in her residence area. But in user 8’s training data, the most frequent activities that happens in her residence area are “doing research” and “others”. Her similar users seem to have consistent activity patterns in training and test data. Consequently, this behavior difference in training/test data also makes the model perform poorly. Notice that in both user 7 and user 8, the “Merged” baseline seems better, as it suffers less from such behavior differences by considering all the users equally.

3.4 The Activity-dependent Data Sparsity Challenge

Another major challenge for activity recognition is that, in the real world, users may only have a small amount of time and effort to set up an activity recognition system; otherwise, they are quite

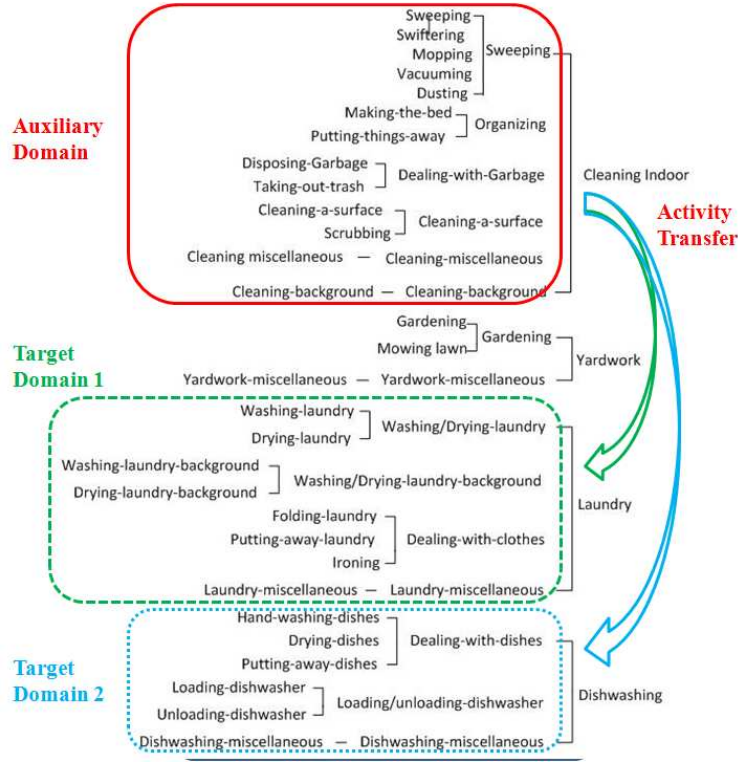


Figure 3.8: An example of CDAR on the MIT PLIA1 dataset [2].

likely to give up using such a recognition system. In order to save human effort on labeling, we try to exploit the semantic similarity among activities and use the auxiliary labeled data from a set of activities (e.g. the “Cleaning Indoor” activities in Figure 3.8) to help train a recognizer for another set of different, but related activities (e.g. “Laundry” or “Dishwashing” activities in Figure 3.8). Different from the previous learning framework, here we focus on sharing the knowledge from the label perspective rather than the model and/or data feature perspectives. We propose a cross-domain activity recognition solution, which relies on Web search to exploit the activity semantic similarity and a weighted Support Vector Machine model to consume pseudo training data generated by the similarities for target activities.

3.4.1 Related Work

Our work exploits the Web to relate two domains, as shown in Table 3.4. In the past, some previous research work had considered learning common sense knowledge from the Web (such as ehow and

Problem	Algorithm	Main drawback
Activity dependent	Common sense model [118, 119, 120]	not consider the case when auxiliary and target activities are different
	Transfer learning model [84, 126, 127, 128, 91]	

Table 3.4: Review on previous work of activity recognition.

KnowItAll [121]) to assist model training. For example, Perkowitz et al. proposed to mine the natural language descriptions of activities (*e.g.* “making-tea”) from ehow.com as labeled data, and translated them into probabilistic collections of object terms (*e.g.* “teacup”, “teabag”, etc.) [118]. Then, they used these probabilistic collections as input data to train a dynamic Bayesian network model for prediction. Wyatt et al. also proposed to mine recipes of the activities from the Web as the labeled data, but they only use this knowledge as a prior and trained a Hidden Markov Model from the unlabeled RFID sensor data [120]. Wang et al. further improved this work by utilizing personal activity data from wearable sensors [119]. They first extracted the actions from the wearable sensors, and then incorporated the actions with the object usage to finally predict the activities. Most previous approaches either exploited only object-usage information or required explicit action modeling. Few of them exploited auxiliary activity data for activity recognition.

As we aim to transfer the labeled data across domains, our work belongs to transfer learning [84, 129]. Transfer learning aims at transferring knowledge from some auxiliary domains to a target domain. In general, the data from the both domains may follow different distributions or be represented in different feature spaces. There are several main approaches to transfer learning in the past. The first approach can be referred to as the instance-transfer [130], where the training examples in an auxiliary domain are weighted for better learning in a target domain. The second approach can be referred to as the feature-representation-transfer [127], which finds a “good” feature representation that reduces difference between both domains for training. The third approach is parameter-transfer [126], which discovers shared parameters or priors between the models in both domains. The fourth approach is the relational-knowledge-transfer, which builds the mapping of relational knowledge between both domains [128]. Our work can be seen as an instance-transfer approach. However, our work is different from the most previous works, because they usually assume that the domains can have different feature spaces but share the same label space. In our work, we consider that the domains can share the same feature space (*e.g.* a same set of sensors at a home), but have different label spaces (*i.e.* different activities). We also notice that, Kasteren et al. have tried to apply transfer learning to help train an activity recognition model in a new

house [91]. But they also considered the two domains (houses) have the same label space, which is different from our work.

3.5 Cross-Domain Learning for Activity Recognition

3.5.1 Problem Formulation and Our Solution

We consider two domains that have the same set of sensors spanning a feature space but have different activity-label spaces. Specifically, we have an auxiliary domain with a set of activities $\mathcal{A}_{aux} = \{a_1, \dots, a_m\}$, and a target domain with another set of activities $\mathcal{A}_{tar} = \{a_{m+1}, \dots, a_n\}$, where $\mathcal{A}_{aux} \cap \mathcal{A}_{tar} = \emptyset$. In the auxiliary domain, we have $\mathcal{D}_{aux} = \{(\mathbf{x}_i, y_i) | i = 1, \dots, n_{aux}\}$. In the target domain, we only have unlabeled test data $\mathcal{D}_{tar}^{tst} = \{\mathbf{x}_i | i = 1, \dots, n_{tar}^{tst}\}$. Here, the auxiliary domain’s sensor data and the target domain’s sensor data share the same feature space, i.e. $\mathbf{x} \in \mathbb{R}^d$; but the two domains have different label spaces, i.e. $y_{aux} \in \mathcal{A}_{src}$ has no overlap with $y_{tar} \in \mathcal{A}_{tar}$. Note that in this problem, we omit the sequential information due to several reasons: first, when we constrain ourselves to using training data from only one subset of activities, we are already “choosing” the activities in the sequences and have damaged the sequential information contained in the original data. Second, it helps to avoid heavy overhead of using activity semantic similarities to generate pseudo sequential data as shown later.

A Cross-Domain Activity Recognition Solution

Our CDAR algorithm can be generalized into three steps. First, we learn the similarities between different activities by mining knowledge from the Web. Then, we generate some pseudo training data by using the auxiliary labeled data and these similarities. Here, “pseudo training data” is the training data with the same feature value as the auxiliary data, but interpreted as target activities’ data with some confidence based on the similarity values. Finally, we apply weighted SVM method [?] to predict the activity labels for the target test data.

Calculate Semantic Similarity among Activities from Web Data With the proliferation of the Web services, there are emerging Web pages that describe the daily activities. For example, we can easily find many web pages introducing how to make coffee. Such web pages encode the

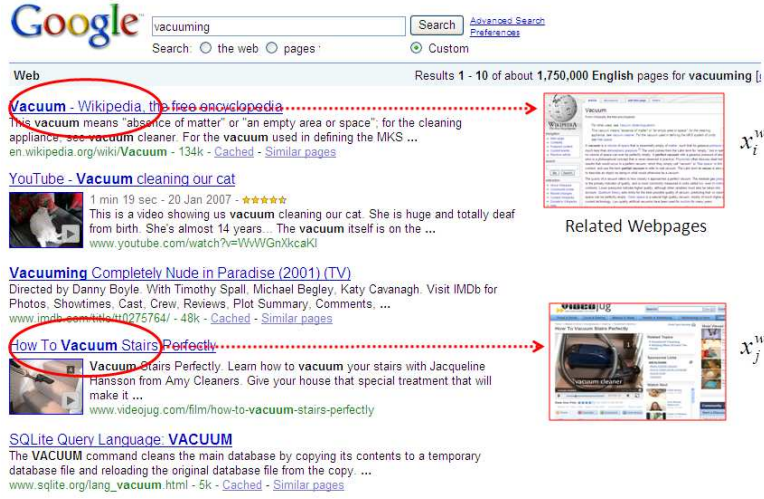


Figure 3.9: Extract Web data for activities.

human understanding to the activity semantics, which can help to measure the activity similarities. Therefore, given an activity name, we can employ Web search to extract web pages related to the activity. For example, for an activity “Vacuuming” defined in the taxonomy of Figure 3.8, we can search on Google with the query “Vacuuming” as shown in Figure 3.9, and get a list of search result Web pages. Then, for each Web page, we can construct a bag-of-words vector x_i^w , with each dimension as the term frequency-inverse document frequency (tf-idf) [131] of a word t :

$$tf-idf_{i,t} = \frac{n_{i,t}}{\sum_l n_{i,l}} \cdot \log \frac{|\{d_i\}|}{|\{d_i : t \in d_i\}|},$$

where $n_{i,t}$ is the number of occurrences of word t in document d_i . Besides, $|\{d_i\}|$ is the total number of collected documents, and $|\{d_i : t \in d_i\}|$ is number of documents where word t appears. The first term $\frac{n_{i,t}}{\sum_l n_{i,l}}$ denotes the frequency of the word t that appears in the document d_i . If the word t appears more in the document d_i , then $|\{d_i : t \in d_i\}|$ is larger, and thus the whole term is larger. The second term $\log \frac{|\{d_i\}|}{|\{d_i : t \in d_i\}|}$ denotes the inverse document frequency for the word t . If the word t , such as “the”, appears in many returned Web pages, then it is not distinctive to describe this activity and thus having a lower weight.

Therefore, for an activity a_u (e.g. “Vacuuming”), we can get a set of documents $\mathcal{D}_u^w = \{x_i^w | i = 1, \dots, m_u\}$, with each x_i^w as a tf-idf vector. Similarly, for another activity a_v (e.g. “Washing-laundry”), we can also Google it and get another set of documents $\mathcal{D}_v^w = \{z_i^w | i = 1, \dots, m_v\}$, with each z_i^w as a tf-idf vector. Then we propose to use the Maximum Mean Discrepancy (MMD) [132], which can directly measure the distribution distance without density estimation, to calculate the

similarity:

$$s_{uv} = \left\| \frac{1}{m_u} \sum_{i=1}^{m_u} \phi(x_i^w) - \frac{1}{m_v} \sum_{i=1}^{m_v} \phi(z_i^w) \right\|_{\mathcal{H}}^2 = \frac{1}{m_u^2} \|K_{uu}^w\|_1 - \frac{2}{m_u m_v} \|K_{uv}^w\|_1 + \frac{1}{m_v^2} \|K_{vv}^w\|_1, \quad (3.8)$$

where $K_{uv}^w(x_i^w, z_j^w) = \exp\left(-\frac{\|x_i^w - z_j^w\|^2}{2\sigma^2}\right)$ is the Gaussian kernel defined over the data \mathfrak{D}_u^w and \mathfrak{D}_v^w . Here, σ is the kernel width for the Gaussian kernel function. In Eq.(3.8), $\|\cdot\|_1$ is an entry-wise norm which sums up all the entries in the matrix.

Weighted SVM with Generated Pseudo-Training Data Given the learned semantic similarity for a pair of activities (a_u, a_v) with $a_u \in \mathcal{A}_{aux}$ and $a_v \in \mathcal{A}_{tar}$, we can generate some pseudo training data for the target activity a_v . Let us denote the auxiliary activity a_u 's data as $D_{aux}^u = \{(\mathbf{x}_i^u, y_i = a_u)\} \subset D_{aux}\}$. Then, the generated pseudo training data for a_v is $D_{pseudo}^{uv} = \{(\mathbf{x}_i^u, y_i = a_v, s_{uv}) | \mathbf{x}_i^u \in D_{aux}^u\}$. Here, s_{uv} is the similarity between a_u and a_v . The larger it is, the more similar the two activities are, and the more confident we are when interpreting such a training data to the target activity a_v . Finally, we have a whole pseudo-training data set as $D_{pseudo} = \{D_{pseudo}^{uv} | a_u \in \mathcal{A}_{aux}, a_v \in \mathcal{A}_{tar}\}$.

We use the weighted SVM to handle these pseudo training data with confidence values. We adopt a ‘‘one-against-one’’ approach to do multi-class classification. For each pair of two target activities a_v and $a_{v'}$, we use the corresponding pseudo training data D_{pseudo}^{uv} and $D_{pseudo}^{u'v'}$, where $a_u \neq a_{u'}$, to train a classifier:

$$\begin{aligned} \min_{\mathbf{w}^{vv'}, b^{vv'}, \xi_{vv'}, \xi_{vv'}} \quad & \frac{1}{2} \|\mathbf{w}^{vv'}\| + C_v \sum_{y=a_v} \xi_{vv'} + C_{v'} \sum_{y=a_{v'}} \xi_{vv'} \\ \text{s.t.} \quad & \mathbf{w}^{vv'} \cdot \phi(\mathbf{x}) + b^{vv'} \geq 1 - \xi_{vv'}, \quad \text{if } y = a_v, \\ & \mathbf{w}^{vv'} \cdot \phi(\mathbf{x}) + b^{vv'} \leq -1 + \xi_{vv'}, \quad \text{if } y = a_{v'}, \\ & \xi_{vv'} \geq 0. \end{aligned} \quad (3.9)$$

Here, $\phi(\cdot)$ is a feature mapping function, \mathbf{w} is the model parameter, b is the bias term, and ξ is the slack variable denoting the classification error. C_v denotes the weight for the pseudo training data D_{pseudo}^{uv} , and it is equal to the similarity value $s(u, v)$. Intuitively, if the weight C_v is higher, the pseudo training data points from D_{pseudo}^{uv} are more trusted in training the SVM model. After solving the above equation, we use a voting strategy on all the binary classification results for multi-class classification. In case that two classes have identical votes, the one with the smallest class index is simply chosen. Finally, we summarize our CDAR method in Algorithm 3.

Algorithm 3 Cross-Domain Activity Recognition Algorithm

Input: Labeled data \mathcal{D}_{aux} from an auxiliary activity set \mathcal{A}_{aux} ; test data \mathcal{D}_{tar}^{tst} from a target domain's activity set \mathcal{A}_{tar} .

Output: Labels on the test data.

begin

- 1: For each activity $a_u \in \mathcal{A}_{src}$, extract a list of Web pages from some search engine and form a Web data set \mathfrak{D}_u^w ;
- 2: For any two activities $a_u \in \mathcal{A}_{src}$ and $a_v \in \mathcal{A}_{tar}$, calculate the similarity $s(u, v)$ by using Eq.(3.8);
- 3: Generate pseudo training data D_{pseudo} from D_{aux} and the learned $\{s(u, v)\}$;
- 4: Construct a set of binary weighted SVM classifier for each pair of D_{pseudo}^{uv} and $D_{pseudo}^{u'v'}$ with $a_u \neq a_{u'}$ by using Eq.(3.9);
- 5: **return** Labels of \mathcal{D}_{tar}^{tst} with a voting strategy.

end

3.5.2 Empirical Study

In this paper, we use three real-world activity recognition datasets to validate our algorithm. Our first dataset (Amsterdam in short) ⁴ is from [133] where a dataset is recorded in the house of a 26-year-old man, living alone in a three-room apartment where 14 state-change sensors are installed. Locations of sensors include doors, cupboards, refrigerators and a toilet flush sensor. Sensors were left unattended and collected data for a period of 28 days, resulting in 2120 sensor events and 245 activity instances with seven different activities annotated: “Leave house”, “Toileting”, “Showering”, “Sleeping”, “Preparing breakfast”, “Preparing Dinner”, “Preparing Beverage”. The second dataset we use is the MIT PLIA1 dataset ⁵ [2], which was recorded on March 4, 2005 from 9AM to 1PM in the MIT PlaceLab. The dataset contains 89 different activities and was manually classified into several categories such as “Cleaning”, “Yardwork”, *etc.* The third dataset is from [5] and is provided by Intel Research Lab (Intel in short), which aims to recognize 11 routine morning activities.

The evaluation criterion we use in this paper is rather standard and simple. Since we are omitting all possible sequential information in the dataset, we just calculate the accuracy we achieve on the

⁴<http://staff.science.uva.nl/~tlmkaste/research/software.php>

⁵http://architecture.mit.edu/house_n/data/PlaceLab/PLIA1.htm

test data set, in other words, the number of correctly predicted activities divides the total number of activities.

We also briefly describe how we handle the Web pages we crawled from the search engine. Using the activity names (e.g. preparing breakfast, cleaning misc, etc.) as queries, we submit these queries to Google and then the top K results are retrieved from, where K is a parameter we will tune in the next sections to show the relationship between algorithm performance and the number of Web pages we retrieve for each query. Next, data preprocessing has been applied to the raw data where all letters in the text are converted to lower case and the words are stemmed using the Porter stemmer [134]. Furthermore, stop words are removed. The SVM implementation we used is the LIBSVM package⁶.

Algorithm Performance We report the performances of our algorithm on three datasets with different auxiliary domain’s activities and target domain’s activities. We will also describe the way we choose the activities in the auxiliary domain and how we choose our target domain for testing in detail.

In our first dataset, there are only seven activities. Therefore, we use training data from 3 activities for training (auxiliary domain), and using the remaining 4 activities for testing (target domain). All sensor events with their activities in the auxiliary domain are used as training data and the rest used as testing data. The process of selecting different activities is repeated ten times and we report the average accuracy and standard deviation we get under different parameters K . Here K is the number of top ranked Web pages we extract from the search engine results. We use a similar way of choosing activities in the auxiliary domain and the target domain in the dataset [5], while we use five activities in the auxiliary domain and the remaining six activities in the target domain. Again, the algorithm is repeated ten times to report a mean accuracy and standard derivation value. Detailed results are shown in Table 3.5 below. The row “Supervised” indicates the accuracy we achieve using SVM classifier with normal ten-fold cross validation on the target domain when we could acquire labeled data on them, in other words, the performance of SVM classifier under the traditional supervised learning setting on the target domain. Such a result could be used as a baseline and an upper-bound to understand how good the performance of our cross-domain activity recognition system is.

In the MIT PLIA1 dataset, since there are many activities included in this dataset and a taxonomy

⁶<http://www.csie.ntu.edu.tw/~cjlin/libsvm>

K	Amsterdam Acc(Std)	Intel Acc(Std)
K = 5	40.2% (21.7%)	39.8% (19.7%)
K = 10	53.7% (22.8%)	47.3% (20.2%)
K = 20	65.8% (22.1%)	58.1% (20.7%)
K = 50	66.7% (21.2%)	63.2% (23.5%)
K = 100	66.0% (22.4%)	63.1% (20.7%)
Supervised	72.3% (20.7%)	78.3% (17.6%)

Table 3.5: Algorithm performance on Amsterdam and Intel datasets.

could be built to describe these activities [100, 2]. Therefore, in MIT PLIA1 dataset, we analyze how the performance will be when we use the activities under the same category as activities in the auxiliary domain and construct training data, and then do the testing on another set of activities under the same category in the target domain.

Examples are shown in Figure 3.8, when we use activities under the node of “Cleaning Indoor” as activities in the auxiliary domain and activities under the node of “Laundry” or “Dishwashing” as activities in the target domain. Therefore, we are using all sensor events with activities “Sweeping, Swiftering, Mopping, Vacuuming, Dusting, Making the bed, Putting things away, Disposing Garbage, Taking out trash, Cleaning a surface, Scrubbing, Cleaning misc, Cleaning background” as training data and the testing data might be composing of all sensor events with activities “Washing laundry, Drying laundry, Washing laundry background, Drying laundry background, Folding laundry, Putting away laundry, Ironing, Laundry misc”.⁷

The reason for us to choose the auxiliary domain’s activity set and the target domain’s activity set in such a way is as follows: we would like to analyze the performance of our algorithm to transfer from a more “categorized” set of activities to another set of activities, which is more similar rather than chosen at random, as we had done in the previous two datasets. We report the performance of our algorithm tested on different pairs of auxiliary activity sets and target activity sets, with mean accuracies and standard deviations calculated over ten independent runs. Results are reported below in Table 3.6 with various settings of parameter K .

From Table 3.5 and Table 3.6, we can observe that our cross domain activity recognition (CDAR) algorithm could achieve comparable performance to supervised learning classifiers when evaluated on the target domain’s activities and trained on auxiliary domain’s activities. Especially, we find

⁷Some activities, though defined in the taxonomy, do not appear in the MIT PLIA1 dataset, e.g. “Mopping”.

Auxiliary	Target	K = 5 Acc (Std)	K = 10 Acc (Std)	K = 20 Acc (Std)	K = 50 Acc (Std)
Cleaning	Laundry	40.4% (21.2%)	53.4% (19.2%)	57.3% (18.7%)	58.9% (20.5%)
Cleaning	Dishwashing	38.9% (22.8%)	43.2% (18.7%)	49.3% (23.0%)	53.2% (20.7%)
Cleaning	Hygiene	42.4% (21.7%)	48.3% (22.8%)	52.4% (17.6%)	58.3% (20.7%)
Cleaning	Information / Leisure	43.1% (23.0%)	45.7% (17.9%)	52.8% (20.7%)	54.9% (22.8%)
Laundry	Cleaning	41.2% (19.2%)	52.8% (20.5%)	53.2% (20.7%)	60.2% (20.0%)
Laundry	Dishwashing	43.2% (20.2%)	53.2% (20.7%)	58.3% (20.5%)	61.2% (21.2%)
Laundry	Hygiene	49.3% (19.5%)	46.3% (19.2%)	49.5% (23.0%)	58.3% (21.4%)
Laundry	Information / Leisure	32.7% (19.2%)	40.2% (20.7%)	48.3% (20.5%)	49.2% (22.6%)
Dishwashing	Cleaning	45.3% (21.2%)	48.3% (20.5%)	53.2% (21.7%)	59.2% (19.2%)
Dishwashing	Laundry	44.8% (20.5%)	50.1% (21.7%)	54.8% (21.9%)	60.8% (22.6%)
Dishwashing	Hygiene	43.2% (20.2%)	52.7% (22.1%)	57.3% (20.7%)	59.2% (16.7%)
Dishwashing	Information / Leisure	39.3% (20.2%)	43.7% (21.7%)	48.3% (18.7%)	50.3% (19.2%)
Hygiene	Cleaning	44.7% (20.5%)	45.8% (20.5%)	49.7% (20.7%)	52.9% (19.5%)
Hygiene	Laundry	41.8% (20.7%)	43.7% (20.2%)	53.8% (23.0%)	53.7% (17.0%)
Hygiene	Dishwashing	42.9% (22.6%)	42.8% (19.2%)	47.1% (22.8%)	51.2% (18.7%)
Hygiene	Information / Leisure	30.7% (19.5%)	33.8% (20.5%)	38.3% (20.2%)	42.3% (21.2%)

Table 3.6: Algorithm performance on MIT PLIA1 dataset

that when we evaluate SVM classifier under a supervised learning scenario on the Amsterdam dataset, the accuracy is around 72%, whereas we could achieve a performance of 66% using our cross-domain activity recognition algorithm, which is very close to the upper bound.

We also observe that, with the increasing value of K , the performance generally increases. Such an observation follows our intuition, which means more information is being extracted from the top Web pages and generally the MMD value calculated is more accurate, thereby improving the performance of the algorithm overall. Here we make another special note of how the choice of value K would affect our algorithm performance. From Table 3.5 and Table 3.6, we could observe that a good value of K , around 20 or 50, would give us the optimal results. The reason is that, when K is too small, the Web page data is sparse and hence we could not learn much useful information for calculating similarity function and when K is too large, it may probably add noise into the Web page we had crawled and therefore it may degrade the overall performance of our algorithm.

Therefore, our experimental results on three datasets could validate the effectiveness of our algorithm.

Choice of Similarity Functions The Maximum Mean Discrepancy (MMD) function in our proposed approach seems to be an ad hoc choice, and it is natural for one to ask the question about whether it is chosen deliberately to improve the performance of our algorithm or whether other similarity functions would also achieve comparative performance, compared to MMD? To answer

this question, we also evaluated our algorithm using another similarity function that is also popular in calculating the similarity in information retrieval, i.e. the cosine similarity [131].

In short, the cosine similarity is defined as the cosine of the angle between two vectors, defined as:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{|A||B|}, \quad (3.10)$$

where A and B are two vectors and for text matching. Such a value is easy to calculate based on the Webp ages we crawled. In our case, we generate A and B by merging the term-frequency (tf) vectors of the documents for the two candidate activities. For example, given the K Web pages extracted for some activity a_A , we will add up all the corresponding K tf vectors to get such a vector A . Similarly, we can get the vector B for some activity a_B , so that we can compute the Eq. (3.10). Using similar approaches except the MMD function replaced with the cosine similarity function, we report our results for the Amsterdam dataset [133] and the Intel dataset [5] in Table 3.7.

K	Cosine Similarity		MMD Similarity	
	Amsterdam Acc(Std)	Intel Acc(Std)	Amsterdam Acc(Std)	Intel Acc(Std)
K = 5	48.7%(19.5%)	42.4% (20.5%)	40.2% (21.7%)	39.8% (19.7%)
K = 10	53.2%(20.7%)	45.3% (20.2%)	53.7% (22.8%)	47.3% (20.2%)
K = 20	58.3% (20.2%)	52.1% (20.7%)	65.8% (22.1%)	58.1% (20.7%)
K = 50	62.1% (19.2%)	57.3% (19.0%)	66.7% (21.2%)	63.2% (23.5%)
K = 100	65.3% (16.7%)	62.3% (21.7%)	66.0% (22.4%)	63.1% (20.7%)

Table 3.7: Algorithm performance with cosine and MMD Similarities.

In Table 3.7, the left two columns are results using cosine similarity functions and the right two columns are results using MMD functions, which is the same as results reported Table 3.5. We could see that although the results using cosine similarity functions are slightly worse than the results we report using MMD functions, it still achieves comparable performance to MMD. Therefore, we could make the conclusion that by incorporating other “meaningful” and “reasonable” similarity functions, our cross-domain activity recognition algorithm could still achieve reasonable performance.

In addition, one particular characteristic of our algorithm is that, the standard deviations of the experimental results are large. The reason is that in each round of training and testing, it is possible that we are selecting quite different sets of activities. In some rounds the activities drawn in the auxiliary domain are more similar to the activities in the target domain compared to the other rounds, thus making the prediction accuracy not as stable.

Discussion of Experiments Here we briefly discuss and summarize some characteristics of the results we report from our experiments.

- Firstly, our algorithm CDAR, could successfully transfer knowledge between auxiliary domain and target domain and thus solve the cross-domain activity recognition task. From Tables 3.5, 3.6 and 3.7, we could see that in most cases, our algorithm could achieve an accuracy of more than 60% when evaluating on the test domain activity set. Such a performance is rather promising since (i) we did not use any sequential information, which had proved to be of great help in traditional activity recognition tasks; (ii) we did not use any labeled data in the target domain, such a feature is especially effective in real world use, since the activity recognition system will be trained on a predefined set of activities and then when users use such an activity recognition system, he may perform activities outside of the predefined activity set. Therefore, such an algorithm that could perform cross-domain activity recognition and does not acquire training data in the target domain would be especially useful.
- Secondly, the number of K Web pages we extract from the search engine results would affect the algorithm performance but would be rather close to the optimal results or converge when K is larger than 50. This means that we could approximate the underlying similarity score between different activities by using around 50 related Web documents. Another advantage is that the cost of performing such a Web search would not be heavy to put into real world usage.
- Thirdly, the choice of similarity functions would not affect our algorithm performance that much as long as a reasonable similarity function that approximates the underlying similarity would be used. In our experiments we have already validated this conclusion through the usage of both MMD and cosine similarity functions and from the comparison between these two functions, we arrive at such a conclusion.
- Fourthly, whether the cross-domain activity recognition performance would be successful or not depends not only on the algorithm but on some underlying characteristics of the activity set. In Table 3.6, we could note that some activity pairs perform worse than others, for example, when we are transferring knowledge from activities under the subcategory of “Hygiene” (Auxiliary Domain) to activities under the subcategory of “Information / Leisure” (Target Domain), we could only achieve an accuracy of around 42% at the best. (Corresponding to

the last row in Table 3.6.) Such a result might be of the reason that there is a relatively larger distance between “Hygiene” activities and “Information/Leisure” activities. In contrast, we could also find that when we use activities in the “Dishwashing” subcategory as the auxiliary domain and activities in the “Cleaning” or “Laundry” subcategory as the target domain, the accuracy we achieve is much higher, suggesting that these two kind of activities have closer relationships, which also follows our intuition. Thus, one interesting topic to study is to determine when we could successfully “transfer the knowledge” between the auxiliary domain and the target domain, or, if we had known the activity set in the target domain in advance, what auxiliary domain would be best for us to use to ensure that the algorithm performance can be guaranteed? One possible choice can be using some activity ontology and shrinkage based approach to combine data from similar activities for training the classifier [135].

3.6 Summary

In this chapter, we study two data sparsity challenges in activity recognition, including limited data for each single user to train a personalized recognition model and even no (annotated) sensor data for the target activities in training. We provide two solutions that make use of similar users’ data and similar activities’ data to help reduce the calibration effort. For the user-dependent data sparsity challenge, we propose a user-dependent aspect model which formulates the user grouping information with latent user aspects. Therefore, each single user can benefit from other similar users’ data collaboratively. Our model can also use the time information and provide activity recognition to a new user. For the activity-dependent data sparsity challenge, we propose a simple yet effective cross-domain learning approach which can employ the Web knowledge to help measure similarities among the activities and thus generate some pseudo training data for the target activities. We test our models with various real-world sensor data, including campus-scaled WiFi activity data and several home setting state-change and/or RFID activity data. For the user-dependent challenge, we show an average performance lift (w.r.t. different training data sizes) of 11% over the baseline with pooling all the users’ data together and 13% over the baseline with each single user’s data, for the existing user’s activity recognition. For the activity-dependent challenge, we show that our model of not using any labeled data for the target activities can still provide reasonable activity predictions. Meanwhile, our model also has comparable performance with those traditional supervised activity recognition models of using labeled data for the target activities.

In the future, we aim to extend our work in several directions. For our collaborative activity recognition model to address the user-dependent data sparsity challenge, we are interested in exploring unlabeled sensor data and sequential information to further improve the system. We also want to extend it to online update setting for fast dealing with new observations. For our cross-domain learning model to address the activity-dependent data sparsity challenge, we aim to discuss cases where not only the activity space, but also the sensor feature spaces are different. Besides, we also plan to make better use of the sequential information in modeling. Finally, we would also investigate some intrinsic questions of “whether the activities in two domains can be transferred or not” and “if we cannot acquire training data for activities in the target domain, what activities would be best for us to use as training data in the auxiliary domain”?

Chapter 4

GPS-based Mobile Recommendation

Given user location and activity data, we are interested in building a mobile recommendation system, which is able to give both location recommendations with some activity query and activity recommendation with some location query. Such a system design is highlighted by two location-related queries in our daily life: 1) if we want to do something such as sightseeing or food-hunting in a large city like Beijing, where should we go? 2) If we want to visit a place such as the Bird's Nest in Beijing Olympic park, what can we do there? A possible mobile recommendation system for this is shown in Figure 4.1. A user can input a location, such as "Bird's Nest", as a location query for activity recommendation; then, the mobile recommendation system should show the queried location on the map and suggest a ranking list of activities. For location recommendation, the user can input an activity, such as "tourism and amusement", as an activity query; then the system can suggest a ranking list of candidate locations and display them on the map.

To build such a mobile recommendation system, for example for general recommendation, we first extract the location and activity information from mobile user data. We organize these locations and activities in some form such as a matrix. Then, as shown in Figure 4.2, we can see location recommendation for some given activity query as a ranking over the row entry values in some column, and activity recommendation as a ranking over the column entry values in some row. Each entry rating denotes the popularity of that location-activity pair. Similarly, if we try to take user into account for personalized mobile recommendation, we can model the user-location-activity data as a tensor. Therefore, the personalized location recommendation or activity recommendation happens on one facet of the constructed tensor.

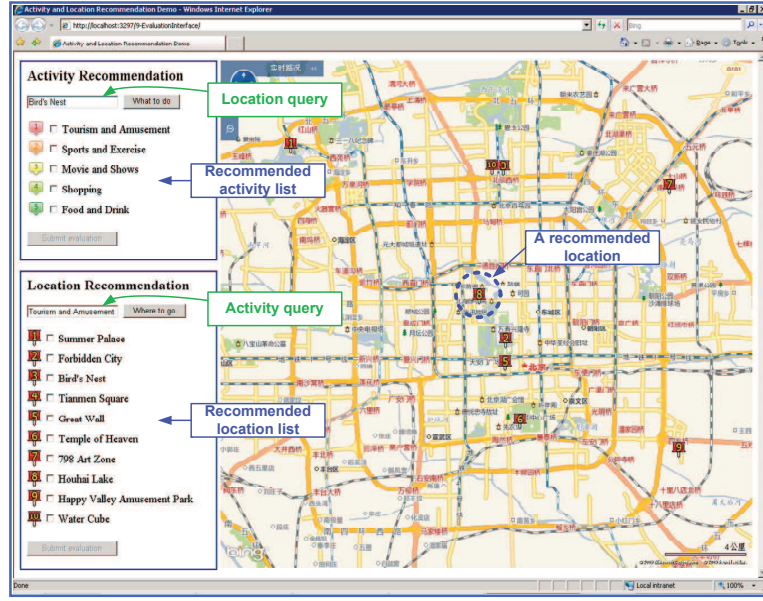


Figure 4.1: A mobile recommendation system.

	Tourism	Exhibition	Shopping
Forbidden City	5	4	2
Bird's Nest	4	3	1
Zhongguancun	1	2	6

Location recommendation

Tourism:
Forbidden City > Bird's Nest > Zhongguancun

Activity recommendation

Forbidden City:
Tourism > Exhibition > Shopping

Figure 4.2: An example of location and activity recommendation by ranking.

4.1 Background on Recommender Systems

4.1.1 Data Input and Output

Input and Output. In standard recommendation systems, there is a user-item rating matrix which indicates how a user likes an item. Denote the users as u, v and the items as i, j . The *input* data are a set of rating values $D = \{(u, i, r_{ui})\}$ that associate users and items. After learning with these rating data, given a user-item pair (u', i') , we can predict its rating $r_{u'i'}$ as an *output*. After predicting the ratings for each user to all the candidate unrated items, we can rank the items according to the ratings and give a top- K recommendation to the user. We call this process as recommendation.

Here, r_{ui} indicates the preference by user u of item i , a higher value means stronger preference. For example, in the Netflix¹ or MovieLens² movie recommendation tasks, the items are movies, and the user's ratings on a movie are integers ranging from one to five, i.e. $r_{ui} \in [1, 5]$. Assume that there are m users and n items in total. As the users usually cannot see and rate all the movies, the number of observed user ratings $|D| \ll m \times n$; in other words, the user-item matrix is incomplete. The goal of recommendation is to predict whether a user is interested in some item that she doesn't rate it before. Applicable algorithms work with the relatively few known ratings, as well as some possible additional feature information about the users and the items, to predict the missing entries of the matrix as the output.

4.1.2 Existing Models

The existing recommendation models can be generally classified into two categories: one is content-based filtering, and the other is collaborative filtering [35].

In *content-based filtering*, the user is recommended items similar to the ones the user preferred in the past. Specifically, for each user and each item, some content features are extracted; then based on these content features, one can find the most similar items that the user may be interested in for recommendation [36]. For example, in movie recommendation, in order to recommend movies to user u , the content-based filtering systems try to understand the commonalities among the movies user u has rated in the past based on the movies' actors, directors, genres, etc. Finally, only those movies that have high similarity to the users preferences would be recommended. An typical example comes from Mooney and Roy's book recommender system [36]. In their system, users are used to rate a set of training books with some discrete 1-10 ratings. Then, these ratings are further interpreted, given a user rating of 1-5 as negative and 6-10 as positive. These books, each as a collection of documents, are used to train a Naive Bayes classifier. Denote a new book as B , containing a collection of documents as $\{s_m | m = 1, \dots, S\}$. Each document s_m contains a collection of words $\{a_{mi} | i = 1, \dots, d_m\}$. Then, the posterior probability of the user being interested to that book is:

$$p(c_j|B) = \frac{p(c_j)}{p(B)} \prod_{m=1}^S \prod_{i=1}^{d_m} p(a_{mi}|c_j, s_m),$$

¹<http://www.netflixprize.com>

²<http://movielens.umn.edu>

where c_j can be positive class or negative class. A major limit for pure content-based recommendation methods is that, a user's own ratings are the only factor influencing future recommendation. This can possibly make the recommendation task hard given few ratings from each user. Therefore, people started to explore the collaborative filtering to address this problem. An example is from Balabanović and Shoham's hybrid content-based and collaborative Web page recommender system [136]. In their system, users receive Web pages both when they score highly against their own profile, and when they are rated highly by a user with a similar profile. Another example is from Popescul et al.'s research paper recommender system [137]. They provided a three-way aspect model, which presumes that users are interested in a set of latent topics which in turn generate both documents and document content information. All the examples show that, with the help of collaborative filtering, the recommendation performance can be greatly improved. Therefore, in the following, we introduce the typical collaborative filtering techniques.

In *collaborative filtering*, user or item contents are not necessary. Instead, in collaborative recommendations, the user is recommended items that people with similar preferences liked in the past. The collaborative filtering (CF) algorithms can be further classified into two categories: memory-based CF and model-based CF.

Memory-based CF Memory-based CF techniques usually use user rating data to compute similarity between users or items. Then, some neighborhood-based algorithm is employed to calculate the similarity between two users or items, produces a prediction for the user taking the weighted average of all the ratings [37, 38, 39].

There are various definitions about similarity. For example, the Pearson correlation measures the extent to which two variables linearly related with each other [138, 37]. In particular, the Pearson correlation between two users u and v is defined as:

$$s_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}},$$

where I is the item sets that both users u and v rated. \bar{r}_u denotes the average rating for user u . The other popular similarity metric is cosine similarity [35], which is defined as:

$$s_{u,v} = \frac{\sum_{i \in I} r_{u,i} r_{v,i}}{\sqrt{\sum_{i \in I} r_{u,i}^2} \sqrt{\sum_{i \in I} r_{v,i}^2}}.$$

Similar Pearson correlation or cosine similarity values can be calculated for item-item similarity by changing the summations over items to over users. Other similarity metrics also exist, we refer

readers to [35] for more details.

After having the similarity values for the users, we can predict the missing entry with user-based CF [139]. For example, given a test user u and a test item i , we can predict the rating as:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N(u)} s_{u,v} (r_{v,i} - \bar{r}_v)}{\sum_{v \in N(u)} s_{u,v}},$$

where $N(u)$ denotes the a set of top- K similar users to u . Similarly, we can get the item-based CF [38]. Wang et al. provided a unified user-based and item-based CF approach by combining the user similarities and item similarities in a probabilistic way [39].

Model-based CF Model-based CF techniques use data mining and machine learning algorithms to recognize complex patterns based on training data, and then predict the ratings for collaborative filtering tasks. Most of these techniques aim to discover the latent factors that explain the rating data. We summarize the models into two categories as follows.

- *Latent semantic analysis.* In latent semantic analysis for collaborative filtering, we usually assume that the observed user ratings can be modeled as a mixture of user communities or interest groups, where users may participate in one or more groups with probabilities [140]. For example, Hofmann proposed a Gaussian probabilistic Latent Semantic Analysis (pLSA) approach. In this approach, the probability of a rating r is formulated as:

$$p(r|u, i) = \sum_z p(r|i, z)P(z|u),$$

where z 's are the hidden states (or user communities) of a hidden variable Z introduced to make user u and item i rendered conditionally independent. The possible set of states z is assumed to be finite and of size k . As the rating values are numerical, a natural way to formulate the rating probability is using a Gaussian distribution:

$$p(r|u, i) = \sum_z p(r; \mu_{i,z}, \sigma_{i,z})P(z|u),$$

where $p(r; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp[-\frac{(r-\mu)^2}{2\sigma^2}]$ is a Gaussian distribution with mean $\mu \in \mathcal{R}$ and variance $\sigma^2 \in \mathcal{R}$. Therefore, the expected rating for a user-item pair can be easily computed as

$$E[r|u, i] = \int r p(r|u, i) dr = \sum_z P(z|u) \mu_{i,z}.$$

As the pLSA model so far assumes that all users have their ratings on a common scale, but in the world, different users can have different rating mean values. It is then reasonable to further employ some user normalization by

$$p(r|u, i) = \mu_u + \sigma_u \sum_z P(z|u) p\left(\frac{r - \mu_u}{\sigma_u}; \mu_{i,z}, \sigma_{i,z}\right),$$

where μ_u denotes a user-specific rating mean value and σ_u its standard variation. By replacing the forced predictor $p(r|i, z)$ by the free predictor $p(r, i|z)$, the data log likelihood is given by $\sum_{\langle u, i, r, z \rangle} \log P(z|u) + \log p(r, i|z)$; and maximizing the log likelihood by expectation maximization leads to a train collaborative filtering model for further prediction [141]. Latent Dirichlet Allocation (LDA) can be seen as an extension of the standard pLSA, by applying some Dirichlet prior on the topic-user probability $P(z|u)$ and the term-topic probability $P(i|z)$. It is applied to various recommendation scenarios, such as tag recommendation [142] and user community recommendation [143].

- *Low-rank approximation.* In low-rank approximation, one can see the user-item ratings as an incomplete user-item matrix, and the goal to find the user and item latent factors which can be used to reconstruct the matrix. In particular, each rating $r_{u,i}$ is a dot product of two vectors $\mathbf{p}_u \in \mathcal{R}^k$ and $\mathbf{q}_i \in \mathcal{R}^k$, which correspond the low-dimensional representations of user u and item i respectively. For an item i , the elements of \mathbf{q}_i measure the extent to which the item possesses those factors. For a user u , the elements of \mathbf{p}_u measure the extent of interest the user has in those items that are high on the corresponding factors. Therefore, the dot product $\mathbf{p}_u \cdot \mathbf{q}_i^T$ measures user u 's overall interest in item i , and it approximates the rating $r_{u,i}$ as

$$r_{u,i} = \mathbf{p}_u \cdot \mathbf{q}_i^T = \sum_{j=1}^k p_{uj} q_{ij}.$$

This is closely related to singular value decomposition (SVD), which is formulated as $r_{u,i} = \sum_{j=1}^k \sigma_j p_{uj} q_{ij}$, given $\sigma_j \geq 0$ and $\mathbf{p}_j, \mathbf{q}_j$ as orthonormal vectors.

To learn the factor vectors \mathbf{p}_u and \mathbf{q}_i , the basic matrix factorization minimizes the regularized squared error on the set of known ratings [42]:

$$\min \sum_{(u,i) \in D} (r_{ui} - \mathbf{p}_u \cdot \mathbf{q}_i^T)^2 + \lambda (\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2),$$

where $\lambda > 0$ is the model parameter. When the latent factors $\mathbf{p}_u \geq 0$ and $\mathbf{q}_i \geq 0$, the above optimization becomes non-negative matrix factorization [144].

One benefit of using matrix factorization for collaborative filtering is that, it is more flexible in encoding various data aspects w.r.t. different application requirements. For example, we can consider the confidence level in matrix factorization, therefore a less important observed rating is assigned with a lower weight in measuring the prediction loss [40, 48]:

$$\min \sum_{(u,i) \in D} w_{u,i} (r_{ui} - \mathbf{p}_u \cdot \mathbf{q}_i^T)^2 + \lambda (\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2),$$

where $w_{u,i} \geq 0$ is the weight value. Besides, in order to capture the rating variations of users and items, we can add some bias terms in the first-order approximation to the ratings:

$$\hat{r}_{u,i} = \mu + b_u + b_i + \mathbf{p}_u \cdot \mathbf{q}_i^T,$$

where μ is the average rating for all the items; b_u and b_i are the deviations of user u and item i from the average. This approximation can be plugged into the above minimization framework with the model parameters including both latent factors and the bias terms [43]. One can further extend the above approximation with temporal dynamics [44]:

$$\hat{r}_{u,i}(t) = \mu + b_u(t) + b_i(t) + \mathbf{p}_u(t) \cdot \mathbf{q}_i^T,$$

so that a user-item rating can depend on time. Here, the user bias term, the item bias term and the user latent factor are all time varying, while the item latent factor remains static because the item characteristics are static in nature. Finally, when there are some additional (matrix) inputs w.r.t. users or items, one can also consider using a collective matrix factorization framework to enable the latent factor sharing across several concurrent matrix factorizations [41]. Tensor, as a multi-dimensional array, is a natural extension of the standard two-dimensional matrix. Similar factorization techniques can be applied to the applications with interactions among multiple entities. Some examples about tensor decomposition include modeling user's sequential online shopping behavior with a user-item-item tensor [45] and modeling user's tagging behavior with a user-item-tag tensor [46]. Typical learning algorithms in low-rank approximation include alternating least squares, gradient descent and stochastic gradient descent. We refer readers to [42] for the details of the comparisons among these algorithms.

It is generally shown that, model-based CF performs better than memory-based CF [48]. It is also argued that collaborative filtering has several advantages over content-based filtering [49], as it

does not depend on error-prone machine analysis of content and it is able to model complex, hard-to-represent concepts, such as taste and quality. In this thesis, we focus on using model-based collaborative filtering for mobile recommendation.

Some advanced topics are not discussed here, but worth further study. First, a major challenge in collaborative filtering is the cold start problem, when there are new users or new items. To address this challenge, there are different ways. For example, one can consider to use incremental learning [139, 47] or online learning [145] to fast update the collaborative filtering model, or use active learning to carefully select a limited number of items for query in order to obtain user’s preference [146, 147]. Second, besides the explicit rating feedback, it is also shown that the missing ratings carry useful information and thus can be used as implicit feedback to further describe the user’s preference [48, 148, 149, 150]. Third, given that in collaborative filtering domain the data volumes are usually huge, large-scale computation solutions based on either more sophisticated optimization strategy [151], or parallel computing [152, 153, 154], or distributed computing with Map/Reduce or MPI (message passing interface) [47, 155] become attractive. Fourth, as recommendation is essentially a ranking problem, it is argued that directly formulating the objective as a ranking optimization problem is better than as an exact rating prediction problem [156, 157]. Last, but not the least, given multiple heterogeneous collaborative filtering domains, transfer learning is shown to be applicable to share the knowledge among them and further improve the prediction performance [158, 159, 160].

4.2 Crowd-dependent Data Sparsity

As we can see, the mobile recommendation we considered in this thesis is based on rankings over a complete location-activity matrix or a complete user-location-activity tensor. However, in practice, the user location and activity data are so sparse because the users either cannot explore all the possible location-activity patterns or do not annotate the activity information. It is generally observed that both the constructed location-activity matrix and the user-location-activity tensor are very sparse, given this limited location history data from the crowd. The crowd-dependent sparsity problem for both general and personalized recommendation is illustrated in Figure 4.3.

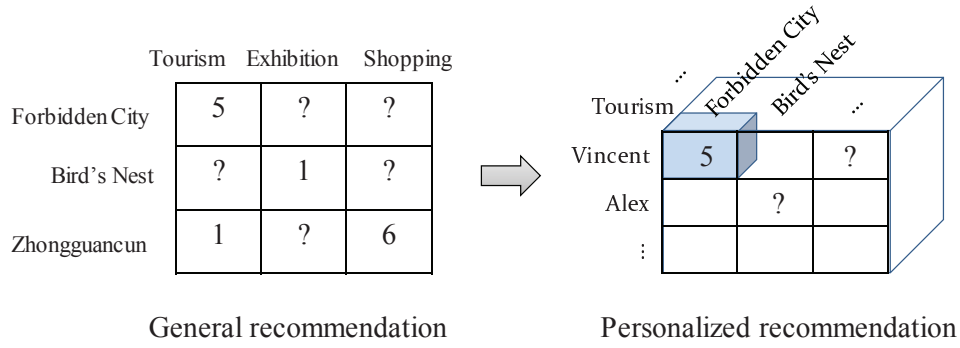


Figure 4.3: Illustration of the data sparsity challenge in mobile recommendation.

4.2.1 Related Work

Some previous work considers to use collaborative filtering (CF) to address the data sparsity problem as shown in Table 4.1. For example, Takeuchi and Sugimoto designed a CityVoyager system, which collected users' shopping histories, and relied on an item-based collaborative filtering method to recommend to a user some shops that are similar to her previously visited shops [33]. Horozov et al. designed a Geowhiz system using a user-based collaborative filtering algorithm to recommend restaurants [161]. Both of these methods are using memory-based collaborative filtering technique. There is also some other work in mobile recommendation that does not use collaborative filtering for solving data sparsity problem. For example, Zheng et al. proposed a HITS-based model is proposed to take into account a user's travel experience and the interest of a location in tourism spot recommendation, so that the locations that are really popular and also recommended by experienced users can be recommended. Park et al. proposed a Bayesian preference model to take into account both users' preferences and location contexts and calculate user's interests in some restaurants for recommendation [32]. Note that most of the above work focuses on location recommendation, while we are interested in handling various types of locations w.r.t. different activities. Besides, model-based collaborative filtering techniques are usually shown to be better than memory-based CF techniques [40, 42], but few of the previous work adopts this category of CF models. There is plentiful auxiliary information related to location, activity and user. For example, from some yellow page database, we are able to know the points of interests (POIs) on each location so that we can know the functionality of that location. We also have the common sense of what kinds of activities being strongly correlated, so that if we know some activity happens, then the other activity is likely to happen as well. Last, but not the least, we can

Algorithm	CF-MemoryBased	Non-CF	Main drawbacks
CityVoyager [33]	✓		single recommendation, no auxiliary info
Geowhiz [161]	✓		single recommendation, no auxiliary info
TravelRec [34]		✓	single recommendation, no auxiliary info
BPM [32]		✓	single recommendation, limited auxiliary info

Table 4.1: Review on previous work of mobile recommendation.

also utilize user’s social network to better give the recommendation. All of this information can be used as the auxiliary information and fit into the recommendation system for better performance. The current mobile recommendation solutions are still limited in addressing these issues.

4.3 Collective Matrix Factorization for General Recommendation

Recall that our goal is to fill the missing entries in the location-activity matrix, and we have the knowledge about the location features and the activity correlations. From the location features, we can know whether a location is similar to another location. Therefore, if we know that one location i is suitable for doing some activity a (such as “Shopping”), and another location j is similar to location i according to the features, we may infer that location j is also suitable for doing activity a . This information can help to fill the missing entries in the location-activity matrix by collaboratively considering the row-row similarities. From the activity-activity correlations, we know how likely the occurrence of one activity implies the occurrence of another activity. As a result, if we know that one activity a (e.g. “Shopping”) usually happens at some location (e.g. shopping mall), and another activity b (e.g. “Food and drink”) is closely related to activity a , then we may infer that many users also do activity b in that location. For example, many people choose to have food and drink in the shopping mall since usually there are many restaurants and bars in/near the shopping mall. Comparatively, this information helps to fill the missing entries in the location-activity matrix by collaboratively considering the column-column similarities.

We provide a collaborative location and activity filtering solution for general mobile recommendation [6], as illustrated in Figure 4.4. Given the location-activity matrix $X_{m \times n}$, we decompose it by low-rank approximation as a product of two matrices $U_{m \times k}$ and $V_{n \times k}$ (the superscript “T” for

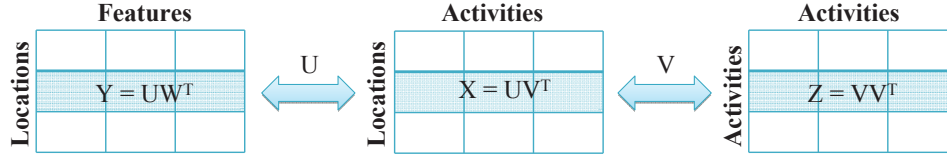


Figure 4.4: Demonstration of our model.

$V_{n \times k}^T$ denotes the matrix transpose), where $k < n$. It shares location information through sharing matrix $U_{m \times k}$ with the location-feature matrix $Y_{m \times l}$, which is decomposed as a product of matrices $U_{m \times k}$ and $W_{l \times k}$. As we can see, each location is now represented by a low-dimensional feature vector with k dimensions. Similarly, the location-activity matrix shares the activity information through sharing matrix $V_{n \times k}$ with the activity-activity matrix $Z_{n \times n}$, which is decomposed as a self product of $V_{n \times k}$. Each activity is also represented by some low-dimensional feature vector with k dimensions. In this way, we are ready to bring low-dimensional location representations and activity representations together for prediction. If two low-dimensional location representations are similar (i.e. two rows in the low-rank matrix U), by multiplying the same low-dimensional activity representation (i.e. some row in the low-rank matrix V), we have the two corresponding entry values in the location-activity matrix X to be close. Similarly, if two low-dimensional activity representations are similar (i.e. two rows in the low-rank matrix V), by multiplying the same low-dimensional location representation (i.e. some row in the low-rank matrix U), this also results in the two corresponding entry values in X being close. Note that, if one row or one column in the interested location-activity matrix is empty (i.e. no record for the corresponding location or activity), our model can still give some predictions. However, such predictions can be biased to the prior knowledge of the location features and the activity-activity correlations.

4.3.1 Problem Formulation and Our Solution

Formally, we formulate our idea as a collective matrix factorization model as:

$$\begin{aligned} \mathcal{L}(U, V, W) = & \frac{1}{2} \|I \circ (X - UV^T)\|_F^2 \\ & + \frac{\lambda_1}{2} \|Y - UW^T\|_F^2 + \frac{\lambda_2}{2} \|Z - VV^T\|_F^2 + \frac{\lambda_3}{2} (\|U\|_F^2 + \|V\|_F^2 + \|W\|_F^2) \end{aligned} \quad (4.1)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. I is an indicator matrix with its entry $I_{ij} = 0$ if X_{ij} is missing, $I_{ij} = 1$ otherwise. The operator “ \circ ” denotes the entry-wise product. Hence, this term corresponds to the loss term in our learning framework of Eq.(5.1). The sparsity in terms

Algorithm 4 CLAR Algorithm

Input: Incomplete location-activity matrix $X_{m \times n}$, location-feature matrix $Y_{m \times l}$ and activity-activity matrix $Z_{n \times n}$.

Output: Complete location-activity matrix $\hat{X}_{m \times n}$.

begin

1. $t = 1$;
2. **While** ($t < T$ and $\mathcal{L}_t - \mathcal{L}_{t+1} > \epsilon$) **do** // T is #(max iterations)
3. Get the gradients ∇_{U_t} , ∇_{V_t} and ∇_{W_t} by Eq.(4.2);
4. $\gamma = 1$
5. **While** ($\mathcal{L}(U_t - \gamma \nabla_{U_t}, V_t - \gamma \nabla_{V_t}, W_t - \gamma \nabla_{W_t}) \geq \mathcal{L}(U_t, V_t, W_t)$) **do**
6. $\gamma = \gamma/2$; // search for the maximal step size
7. $U_{t+1} = U_t - \gamma \nabla_{U_t}, V_{t+1} = V_t - \gamma \nabla_{V_t}, W_{t+1} = W_t - \gamma \nabla_{W_t}$
8. $t = t + 1$;
9. **Return** $\hat{X} = U_t V_t^T$.

end

of locations and activities can addressed by: first, enforcing similar locations/activities to have similar low-dimensional representations U and V ; second, sharing the location feature knowledge and activity correlation knowledge through the factorization on Y and Z . The last term controls the regularization over the factorized matrices to avoid overfitting.

In general, this objective function is not jointly convex to all the variables $U_{m \times k}$, $V_{n \times k}$ and $W_{l \times k}$, and we cannot get closed-form solutions for minimizing the objective function. Therefore, we turn to some numerical method such as gradient descent to get the local optimal solutions. Specifically, we find the gradients (denoted as ∇) for each variable as

$$\begin{aligned}\nabla_U \mathcal{L} &= [I \circ (UV^T - X)] V + \lambda_1(UW^T - Y)W + \lambda_3 U, \\ \nabla_V \mathcal{L} &= [I \circ (UV^T - X)]^T U + 2\lambda_2(VV^T - Z)V + \lambda_3 V, \\ \nabla_W \mathcal{L} &= \lambda_1(Y - UW^T)^T U + \lambda_3 W.\end{aligned}\tag{4.2}$$

After finding the gradients, we can use gradient descent to iteratively minimize the objective function. The details are given in Algorithm 4.

Having the complete location-activity matrix $X_{m \times n}$, for a user query of some location, we can look up the rows of $X_{m \times n}$. If this location exists in our system (i.e. the location coordinates fall in

some stay region), for example, the i -th row of $X_{m \times n}$, we rank the i -th row's values in descending order and return a list of corresponding activities for activity commendation. For example, we search "Bird's Nest" in our system and find it matches with second row (i.e. second location) of $X_{m \times n}$, then we extract the 10th row's ratings, e.g. $x = [5, 1, 3, 2, 4]$, where each entry denotes the rating for an activity. Assume that from left to right in x , the activities are "Tourism", "Exhibition", "Shopping", "Food", "Sports", then we recommend a ranking list of activities with "Tourism" > "Sports" > "Shopping" > "Food" > "Exhibition". Similarly, for location recommendation, given an activity query, we look up the columns of $X_{m \times n}$. If this activity is matched in our system, for example, the j -th column of $X_{m \times n}$, we rank the j -th column's values in descending order and return a list of the top N corresponding locations for location recommendation.

4.4 Collective Tensor Factorization for Personalized Recommendation

To enable the personalized recommendation, we model the user-location-activity relations in a tensor \mathcal{A} , so that for each user, we can perform ranking on her own location-activity matrix slice of the whole tensor [162]. As the tensor can be sparse, we exploit some auxiliary information w.r.t. each tensor entity. In particular, we have the user-user matrix $B \in \mathbb{R}^{m \times m}$ which encodes the user-user similarities in a social network. We aim to use this similarity information to uncover the like-minded users in CF. We also have a location-feature matrix $C \in \mathbb{R}^{n \times p}$, with each feature referring to the (normalized) number of POIs (point of interests, e.g. museums) in that location [34]. For activities, we have a matrix $D \in \mathbb{R}^{r \times r}$ representing the activity-activity correlations showing how likely an activity may happen if another activity happens. Beyond the tensor to model the user-location-activity, we also extract another matrix $E \in \mathbb{R}^{m \times n}$ from the GPS data to model the user-location visiting relations. This matrix could be helpful to model the case when we only know a user visited some place but have no idea what she was doing there. Consequently, our model is illustrated by Figure 4.5.

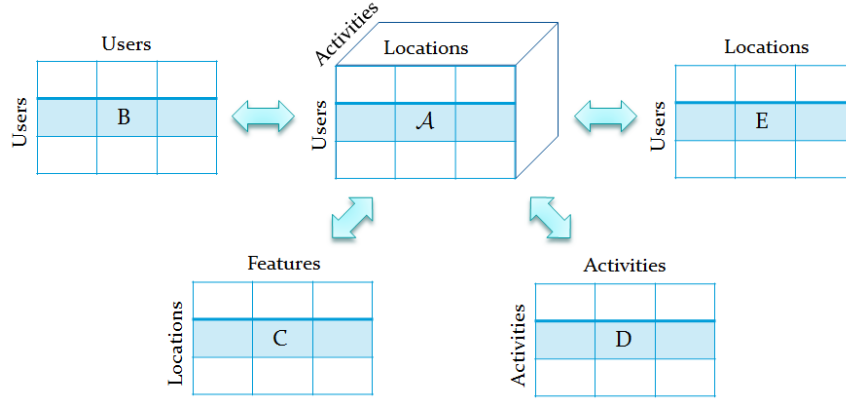


Figure 4.5: Model illustration in a tensor/matrix form.

4.4.1 Problem Formulation and Our Solution

We propose a PARAFAC-style tensor decomposition [163] framework to integrate the tensor with the additional matrices for a regularized decomposition. Specifically, our objective function is

$$\begin{aligned} \mathcal{L}(X, Y, Z, U) = & \frac{1}{2} \|\mathcal{A} - \llbracket X, Y, Z \rrbracket\|^2 \\ & + \frac{\lambda_1}{2} \text{tr}(X^T L_B X) + \frac{\lambda_2}{2} \|C - YU^T\|^2 + \frac{\lambda_3}{2} \text{tr}(Z^T L_D Z) + \frac{\lambda_4}{2} \|E - XY^T\|^2 \\ & + \frac{\lambda_5}{2} (\|X\|^2 + \|Y\|^2 + \|Z\|^2 + \|U\|^2), \end{aligned} \quad (4.3)$$

where the variables $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k] \in \mathbb{R}^{m \times k}$, $Y = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_k] \in \mathbb{R}^{n \times k}$ and $Z = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k] \in \mathbb{R}^{r \times k}$. $\llbracket X, Y, Z \rrbracket = \sum_{i=1}^k \mathbf{x}_i \circ \mathbf{y}_i \circ \mathbf{z}_i$, where “ \circ ” denotes the outer product. Another variable $U \in \mathbb{R}^{p \times k}$. L_B is the Laplacian matrix of B , defined as $L_B = Q - B$ with Q being a diagonal matrix whose diagonal entries $Q_{ii} = \sum_j B_{ij}$. $\text{tr}(\cdot)$ denotes the trace of a matrix. L_D is the Laplacian matrix of D . $\|\cdot\|$ denotes the Forbenius norm. $\lambda_1 \sim \lambda_5$ are model parameters. In (4.3), the first term decomposes the user-location-activity tensor \mathcal{A} as an outer-product of three low-dimensional representations w.r.t. each entity (i.e. X for the users, Y for the locations and Z for the activities). It measures the loss on prediction. The second term poses a regularization term on the users, forcing the low-dimensional representations of two users as close as possible if they are similar as suggested by the additional information. The third term borrows the similar idea with collective matrix factorization [41], by sharing the low-dimensional location representation Y with the tensor decomposition. The fourth term is a regularization term similar to the second term, forcing the low-dimensional representations of two activities as close as possible w.r.t. their correlations. The fifth term shares the low-dimensional user representations X and

location representations Y with the tensor decomposition. Similar to the general recommendation object function, these four terms imply the knowledge share with user similarities, location visit frequency preference, location features and activity correlations. The last term is a regularization term so as to avoid overfitting.

In general, there is no closed form solution for Eq.(4.3), so we turn to numerical methods, such as gradient descent [41], to solve the problem. By taking the derivatives over the objective function, we have

$$\begin{aligned}\nabla_X \mathcal{L} &= -A^{(1)}(Z * Y) + X [(Z^T Z) \odot (Y^T Y)] + \lambda_1 L_B X + \lambda_4 (XY^T - E)Y + \lambda_5 X, \\ \nabla_Y \mathcal{L} &= -A^{(2)}(Z * X) + Y [(Z^T Z) \odot (X^T X)] + \lambda_2 (YU^T - C)U + \lambda_4 (XY^T - E)^T X + \lambda_5 Y, \\ \nabla_Z \mathcal{L} &= -A^{(3)}(Y * X) + Z [(Y^T Y) \odot (X^T X)] + \lambda_3 L_D Z + \lambda_5 Z, \\ \nabla_U \mathcal{L} &= \lambda_2 (YU^T - C)^T Y + \lambda_5 U,\end{aligned}\tag{4.4}$$

where $A^{(i)}$ denotes the mode- i tensor unfolding with $A^{(1)} \in \mathbb{R}^{m \times nr}$, $A^{(2)} \in \mathbb{R}^{n \times mr}$, $A^{(3)} \in \mathbb{R}^{r \times mn}$. In particular, a tensor entry $a_{i_1 i_2 i_3}$ has a corresponding position (i_n, j) in each mode's unfolding: for mode-1, $j = i_2 + (i_3 - 1)n$; for mode-2, $j = i_1 + (i_3 - 1)m$; for mode-3, $j = i_1 + (i_2 - 1)m$. Besides, “ $*$ ” denotes the Khatri-Rao product: for two matrices $V = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_J] \in \mathbb{R}^{I \times J}$ and $W = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_J] \in \mathbb{R}^{T \times J}$, their Khatri-Rao product is defined as $V * W = [\mathbf{v}_1 \otimes \mathbf{w}_1, \mathbf{v}_2 \otimes \mathbf{w}_2, \dots, \mathbf{v}_J \otimes \mathbf{w}_J] \in \mathbb{R}^{IT \times J}$, where “ \otimes ” denotes the Kronecker product. “ \odot ” denotes the Hadamard product (or, entry-wise product).

Given the incomplete tensor and additional information matrices, our goal is to complete the tensor for output. As shown in Algorithm 5, we use an iterative algorithm to solve the problem. In each iteration, we calculate the gradients for the objective function \mathcal{L} according to (4.4), and then get the updated objective function until it reaches the minimum. In the algorithm, T is the maximal number of iterations and γ is the step size. We set $T = 1000$ and $\gamma = 0.0001$ in our experiments. Finally, after the iteration stops, we reconstruct the user-location-activity tensor $\hat{\mathcal{A}}$ by using the low-dimensional representations X, Y, Z . To do recommendations, for an existing user u ($1 \leq u \leq m$), we have her location-activity matrix as $G \in \mathbb{R}^{n \times r}$ with $G(l, a) = \hat{\mathcal{A}}(u, l, a)$ for $1 \leq l \leq n, 1 \leq a \leq r$. Then we are ready for recommendations: given a location input l , we rank the l -th row of G in a decreasing order, and recommend to user u some top activities to do there. Similarly, given an activity input a , we rank the a -th column of G in a decreasing order, and recommend to user u some top locations to go. We can also recommend to a new user in a similar way, except that we construct such a location-activity matrix by summing up the tensor along its user dimension:

Algorithm 5 The UCLAF Algorithm

Input: An incomplete location-activity-user tensor \mathcal{A} and four additional information matrices B, C, D, E .

Output: The complete tensor for \mathcal{A} , denoted by $\hat{\mathcal{A}}$.

begin

```
1:  $t = 1$ ;  
2: while  $t < T$  and  $\mathcal{L}_t > \mathcal{L}_{t+1}$  do  
3:   Get the gradients  $\nabla_X, \nabla_Y, \nabla_Z$  and  $\nabla_U$  by Eq.(4.4);  
4:    $X_{t+1} = X_t - \gamma \nabla_X, Y_{t+1} = Y_t - \gamma \nabla_Y, Z_{t+1} = Z_t - \gamma \nabla_Z, U_{t+1} = U_t - \gamma \nabla_U$ ;  
5:   if  $\mathcal{L}_t < \mathcal{L}_{t+1}$  then break; end if  
6:    $t = t + 1$ ;  
7: end while  
8: return  $\hat{\mathcal{A}} = \llbracket X_t, Y_t, Z_t \rrbracket$ .
```

end

$G' \in \mathbb{R}^{n \times r}$ with $G'(l, a) = \sum_{u=1}^m \hat{\mathcal{A}}(u, l, a)$.

Let us consider the algorithm's complexity. At each iteration, to get the gradient ∇_X , we spend $O(mnr + m^2)$ w.r.t. the input data dimensions m, n, r . Similarly, for ∇_Y , we have $O(mnr)$; for ∇_Z , we have $O(mnr)$; for ∇_U , we have $O(n)$. Thus in total, for getting the gradients, we spend $O(mnr + m^2 + mnr + mnr + n) = O(mnr + m^2)$. To calculate the objective function, we spend $O(mnr + m^2 + r^2)$. As the maximal number of iterations is constant, the total complexity of this algorithm is $O(T \times (mnr + m^2 + r^2)) = O(mnr + m^2 + r^2)$, showing that our algorithm is efficient.

4.5 Empirical Study

In our experiments, we got data from 162 users (61 females and 101 males, and more statistics is shown in Figure 4.6) who carried GPS devices to record their outdoor trajectories from April 2007 to Oct. 2009.

Figure 4.7(a) shows the GPS devices used to collect data, which are comprised of stand-alone GPS receivers and GPS phones. In general, the sampling rate for GPS devices is set as two seconds.

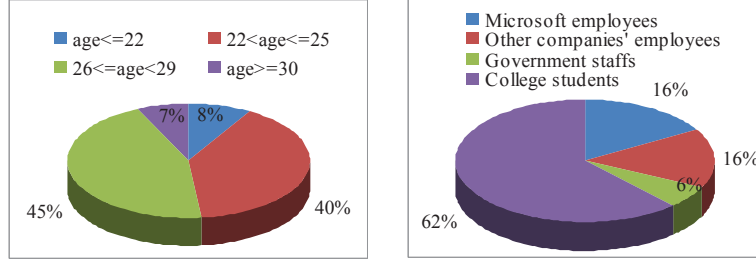


Figure 4.6: GPS user statistics.

The GPS logs were collected in China, as well as a few cities in the United States, South Korea, and Japan. As most parts of the logs were generated in Beijing, and for easier evaluation of our system, we extracted the logs from Beijing for our experiments. After this data preprocessing, we obtain a dataset having 12,765 GPS trajectories with a total of over 3,980,320 GPS points and a total trajectory length of over 139,310 kilometers. We have a total of 530 comments. To make sure that we recommend useful locations and activities, we also remove some GPS points for work and home. The data distribution in Beijing is shown in Figure 4.7(b).

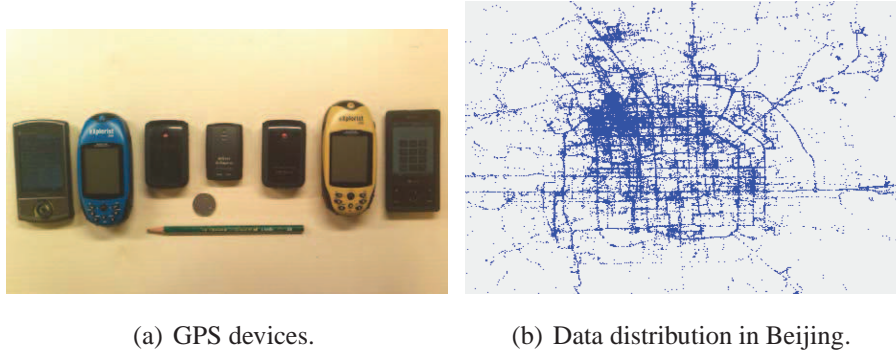


Figure 4.7: GPS devices and data distribution.

We defined five activities to recommend from the GPS data. As shown in Table 4.2, these 5 activities basically cover people's daily routines. Therefore, $n = 5$ for the matrices X and Z in Eq.(4.1).

Parameter Selection

- **Data processing parameters.** In this experiment, to obtain stay points from raw GPS data, we follow our previous work [34] to set T_{thresh} as 20 minutes and D_{thresh} as 200 meters

Table 4.2: Activities that we used in the experiments.

Activities	Descriptions
Food and drink	Dinning/drinking at restaurants/bars, etc.
Shopping	Supermarkets, department stores, etc.
Movie and shows	Movie/shows in theaters and exhibition in museums, etc.
Sports and exercise	Doing exercises at stadiums, parks, etc.
Tourism and amusement	Tourism, amusement park, etc.

for stay point detection. To extract stay regions, we tentatively set the stay region size at $d = 300$, and we study its impact later.

- **Model parameters.** Our model has three parameters: λ_1 , λ_2 and λ_3 , where λ_1 and λ_2 control the contributions of location features and activity correlations respectively, and we study their impacts later. λ_3 controls the regularization term, and we set it as 10 through all our experiments. As our model is based on low-rank approximation, we set the rank $k = 3$ for Eq.(4.1).

Evaluation Methodology

To evaluate our general recommendation system, we first invited five subjects who are familiar with Beijing, to individually use our system and provide the feedback for a user study. For activity recommendation, we asked the subjects to evaluate the top five recommended activities on the top twenty popular locations according to the GPS logs; and for location recommendation, we asked them to evaluate the top ten recommended locations. Our system’s user interface is shown in Figure 4.1, where users can provide ratings to the recommended locations/activities.

Table 4.3: Rating criteria for locations and activities.

Rating	Explanation
3	I’d like to visit this location / do this activity
2	I’d like to visit this location if passing by / do this activity if time spared
1	I have no feeling to visit this location / do this activity
0	This location does not deserve visit / this activity is not suitable to do there.

In Table 4.3, we list the rating criteria. To get the ground truths for evaluation, we aggregate all the subjects’ feedback to get an ideal ranking list. As our recommendations are based on ranking

results, we employ normalized discounted cumulative gain (nDCG) [131] to measure our retrieved location/activity list. nDCG is commonly used in information retrieval to measure the search engine’s performance. A higher nDCG value to a list of search results means that, the highly relevant items appearing earlier (with higher ranks) in the result list. In particular, $nDCG[p]$, or referred as $nDCG@p$, measures the relevance of top p results:

$$nDCG[p] = \frac{DCG[p]}{IDCG[p]}, \quad DCG[p] = rel_1 + \sum_{i=2}^p \frac{rel_i}{\log_2 i},$$

where $IDCG[p]$ is the $DCG[p]$ value of ideal ranking list. rel_i is a relevance value. nDCG ranges from 0 to 1. The higher nDCG is, the better a ranking result list is. For example, given a ranking list of 4 items with relevance as $\langle 1, 3, 0, 2 \rangle$, the $nDCG@4$ is

$$nDCG[4] = \frac{1 + 3/\log_2 2 + 0 + 2/\log_2 4}{3 + 2/\log_2 2 + 1/\log_2 3} = 0.89.$$

To make our evaluation more thorough, we also employ an objective evaluation methodology to further evaluate our (both general and personalized) recommendation systems. Specifically, at each trial, we randomly split some percentage (e.g. 50%) of the tensor/matrix data for training and hold out the other for testing. Then, we employ two metrics; one is RMSE (root mean square error) to measure the tensor/matrix reconstruction loss on a hold-out test data. For RMSE, the smaller, the better. The other metric is nDCG, as described above, to measure the ranking results based on the reconstructed tensor/matrix from training data. For nDCG, the larger, the better. Finally, we run the testing five times to generate the mean values and standard deviations of the results.

Experiments with Our General Recommendation System

In this section, we first study the performances of our general location and activity recommendation system under different model parameters. Then, we employ two baselines for comparison under several settings. Finally, we give some more insights to our CLAR model.

System Performance - Impact of the Location Feature Information The parameter λ_1 controls the contribution of the location feature information to the objective function (4.1). To study the impact of this information, we vary the value of λ_1 and plot our model’s performances with the average nDCG values over all the subjects in Figure 4.8. In this study, we fix $\lambda_2 = 200$, which is in the magnitude of division of the location-feature matrix size by the activity-activity matrix size.

In this way, we make sure that both location features and activity correlations can contribute to the objective function. We use $nDCG@5$ to evaluate the results for activity recommendation and $nDCG@10$ for location recommendation in this paper if without specific references.

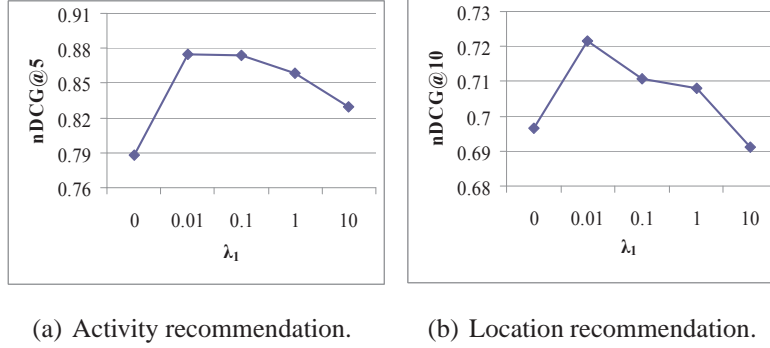


Figure 4.8: Impact of the location features, w.r.t. λ_1 .

As shown in Figure 4.8(a), the model’s performance first increases and later decreases as λ_1 increases. This is because when λ_1 is too small, the model cannot fully utilize the information from the location features to understand the location functionalities and the connections among the locations. When λ_1 is too large, the location feature information dominates the objective function (4.1), thus overwhelming the activity information from the location-activity matrix X and activity-activity matrix Z . It may cause some problem; for example, given two locations i and j with similar location features, if location i has no rating for “food and drink” but has ratings for “movie and shows”, according to location similarity, we cannot recommend to users to try “food and drink” at location j although usually there are some nice restaurants near the movie theaters. In Figure 4.8(b), we observe a similar pattern for location recommendation. When the location-activity ratings are not well inferred with too small or too large λ_1 , the recommended location list also may miss some interesting places. Note that, when $\lambda_1 = 0$, the model equals to only exploiting one additional information source, i.e. the activity-activity correlation, to help the recommendation. As the performance at $\lambda_1 = 0$ is lower than that at $\lambda_1 > 0$ (e.g. $\lambda_1 = 0.1$), we demonstrate the benefit of using location features.

System Performance - Impact of the Activity Correlation Information We also study the impact of parameter λ_2 , which controls the contribution of the activity correlation information to the objective function. We vary the value of λ_2 and plot our model’s performances in Figure 4.9. In this study, we fix $\lambda_1 = 0.1$ according to the previous study in Figure 4.8.

As shown in Figure 4.9(a), similarly we observe the model’s performance first increasing and

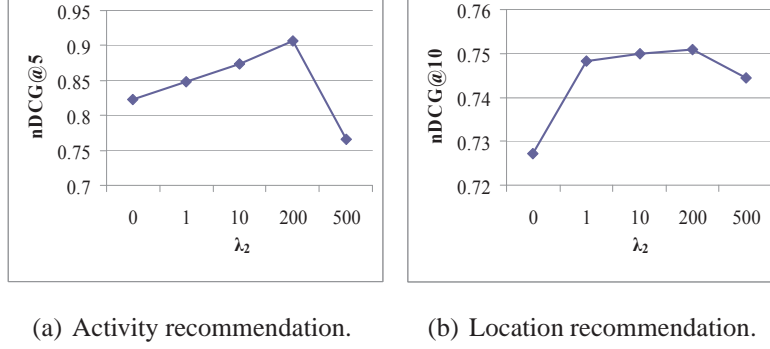


Figure 4.9: Impact of the activity correlations, w.r.t. λ_2 .

later decreasing as λ_2 increases. When λ_2 is too small, the activity correlation information cannot contribute much to the objective function. When λ_2 is too large, the activity correlations can dominate the objective function, so that for a location it recommends the activities mostly based on the correlation values while not fully considering whether such a location is suitable for some activity. For example, if a location has some ratings for “food and drink”, then with a too large λ_2 , the model also recommends the user to see “movie and shows” without carefully considering whether this location has a theater or not. In Figure 4.9(b), we can observe a similar pattern for location recommendation. Note that, when $\lambda_2 = 0$, the model equals to only exploiting one additional information source for location features. As the performance at $\lambda_2 = 0$ is lower than that at $\lambda_2 > 0$ (e.g. $\lambda_2 = 200$), we show the benefit of exploiting activity correlations.

Investigation into Our System - Comparison with Baselines We employ two baselines: single collaborative filtering (SCF) and unifying collaborative filtering (UCF). In SCF, we only use the incomplete location-activity matrix as input for collaborative filtering. We employ the popular low-rank matrix factorization approach to accomplish such a collaborative filtering task [40]. In particular, SCF aims to solve a singular value decomposition problem by $\min J(U, V) = \|I \circ (X - UV^T)\|_F^2$, where X denotes the incomplete location-activity matrix, U and V are the low-rank matrices, I is the indicator matrix the same as Eq.(4.1). It can be seen that this optimization problem equals the case when our objective function (4.1) has both λ_1 and λ_2 as zeros. We employ this baseline to show that with limited number of comments (and thus sparse in location-activity matrix), the recommendation results may not be satisfying. So we can validate our motivation to use additional information sources to help improve the recommendations. We also follow [39] to provide a solution, UCF, which can use the additional information sources for unifying collaborative filtering. In UCF, for each missing entry in the location-activity matrix, it

extracts a set of top N similar locations and top N similar activities, and then uses the ratings for these users over these items in a probabilistic way to calculate a value for the missing entry. After all the missing entries are filled in the location-activity matrix, similar ranking strategy with our system can be used to output the location and activity ranking list for recommendations. We use this baseline to testify the effectiveness of our model over other collaborative filtering methods given the same inputs.

Table 4.4: Comparisons under different p -values for $nDCG@p$.

	Activity Recommendation		Location Recommendation		
	p=3	p=5	p=5	p=8	p=10
CLAR	0.83 ± 0.04	0.91 ± 0.03	0.84 ± 0.06	0.84 ± 0.04	0.86 ± 0.04
UCF	0.72 ± 0.06	0.87 ± 0.03	0.76 ± 0.03	0.74 ± 0.03	0.75 ± 0.03
SCF	0.70 ± 0.07	0.84 ± 0.05	0.63 ± 0.08	0.62 ± 0.07	0.63 ± 0.06

In Table 4.5, we report the performances of our model and other two baselines for both activity and location recommendations. We vary the p -values for $nDCG@p$ to extensively evaluate the systems' performances. The entry value in Table 4.5 denotes the mean and standard deviation of the $nDCG$ values. As shown in the table, our model CLAR consistently outperforms the two baselines under different measurements. We also conduct the t-test over the results and find our results are significantly better than the baselines' results (one-tailed test $p_1 < 0.01$, two-tailed test $p_2 < 0.01$) in both location and activity recommendations. Both our CLAR and UCF can outperform SCF due to using more information. Besides, our CLAR can outperform UCF because in UCF, the information flow is in a single direction from location features and activity correlations while our CLAR enables the information flow in both directions. In other words, in UCF, the location similarities and activity similarities are learned from the location features and activity correlations; then they are passed to the location-activity matrix for collaborative filtering. This collaborative filtering does not have further feedback to the location-features and activity correlations. So, if the similarities learned from this additional information are not accurate, there is no second chance to refine them. In contrast, in our CLAR, we put the location-activity matrix and the two pieces of additional information together in an objective function for optimization, so that we can get the feedback from the matrix factorization in location-activity matrix to the location-feature matrix and activity-activity correlation matrix. In this way, our CLAR can have bi-directional information flows and thus outperform UCF.

Investigation into Our System - Impact of the Stay Region Size We study the impact of stay region size in our recommendation. As discussed above, in recommendations, we may prefer smaller stay region size so that the users can easily find what they want in the recommended location. Therefore, we vary the stay region size by varying the region width d from 200 to at most 500 ($d = 500$ means that the stay region size is 500×500 square meters). As shown in Table 4, as the stay region size increases, the number of stay regions extracted by grid-based clustering (shown in Algorithm ??) decreases. Our CLAR model consistently outperforms the two baselines UCF and SCF. We also conduct the t-test and find our results are better than the baselines' results (one-tailed test $p_1 < 0.05$, two-tailed test $p_2 < 0.05$) in both location and activity recommendations. When $d = 300$, our CLAR works the best, showing that a too small region may make the extracted stay regions' location features insufficient to represent the location functionalities and a too large region may lead to difficulty in finding points of interest from a big area.

Table 4.5: Impact of the stay region size.

	Activity Recommendation			Location Recommendation		
	CLAR	UCF	SCF	CLAR	UCF	SCF
$d = 200$	0.86 ± 0.02	0.85 ± 0.02	0.83 ± 0.02	0.82 ± 0.03	0.72 ± 0.03	0.58 ± 0.05
$d = 300$	0.91 ± 0.03	0.87 ± 0.03	0.84 ± 0.05	0.86 ± 0.04	0.75 ± 0.03	0.63 ± 0.06
$d = 500$	0.86 ± 0.01	0.81 ± 0.03	0.83 ± 0.02	0.86 ± 0.03	0.74 ± 0.03	0.67 ± 0.02

Investigation into Our System - Impact of the User Number As GPS devices become more popular, we have more and more users and accumulate GPS data on the Web as time goes by. We study the impact of user numbers to see whether our system can handle the data well.

Table 4.6: Impact of the user number.

	#(stay point)	Running Time (ms)	Activity Recommend.	Location Recommend.
#user=50	3895	5780.15	0.84 ± 0.04	0.75 ± 0.03
#user=100	8039	10828.45	0.88 ± 0.03	0.89 ± 0.02
#user=162	12656	15053.6	0.90 ± 0.03	0.91 ± 0.03

As the user number increases, the GPS data size increases and thus the number of stay points also increases (though it might not be the case when the user number is sufficiently large). As shown in Table 5, the running time for our CLAR model is almost linear to the number of stay points. This is because the computational complexity of our CLAR model is linear to the number of stay points. Consider the algorithm for our CLAR model in Algorithm 4. Given the input matrices $X_{m \times n}$, $Y_{m \times l}$, $Z_{n \times n}$ and their low-rank factorized matrices $U_{m \times k}$, $V_{n \times k}$, $W_{l \times k}$, we have the computational

complexity of evaluating the objective function (4.1) is: $m \times k \times n + m \times k \times l + n \times k \times n + (m \times n + n \times k + l \times k)$, which is $O(m)$ since n, l and k are much smaller than m (e.g. in our case with 162 users, $m = 12656, n = 5, l = 13, k = 3$). Similarly, we can have the computational complexity for the gradients as $O(m)$. As our algorithm has an iteration limit and in practice converges fast (in less than 300 iterations), the whole computational complexity for our model is linear to the number of stay points $O(m)$. Hence, our model can be quite efficient. From Table 4.6, we also observe that as the user number increases, there are more GPS data and thus we can keep improving the system’s performance.

Discussions - Impact of the Location Types to Activity Recommendation Is our system doing equally well on activity recommendation for different types of locations? We summarize the experimental results for the setting with $d = 300$ and $\#(\text{user})=162$ to answer this question in Figure 4.10(a). As is seen from the figure, for the 20 most popular locations, our system works best on the locations that are in the “food and sports area”, and worst on the locations that are in the “shopping and movie area” (here we aggregate the user evaluations and pick the top two activities as the location types). This is because “food and drink” happens more often in our daily lives; and it is also more likely there would be many restaurant POIs in the location feature for predicting this activity. For “sports”, the location features can capture the location functionality by detecting parks and stadiums. For “tourism”, there are more comments from the GPS users, so that the prediction on such areas can be comparatively accurate. For the “shopping and movies”, the activity recommendation results are not as good as the other areas, because there are fewer comments from the GPS users on these activities and thus fewer ratings in collaborative filtering. Besides, such areas are usually also suitable for food searches and sometimes tourism, so that they may be overwhelmed by the recommendations for food and tourism.

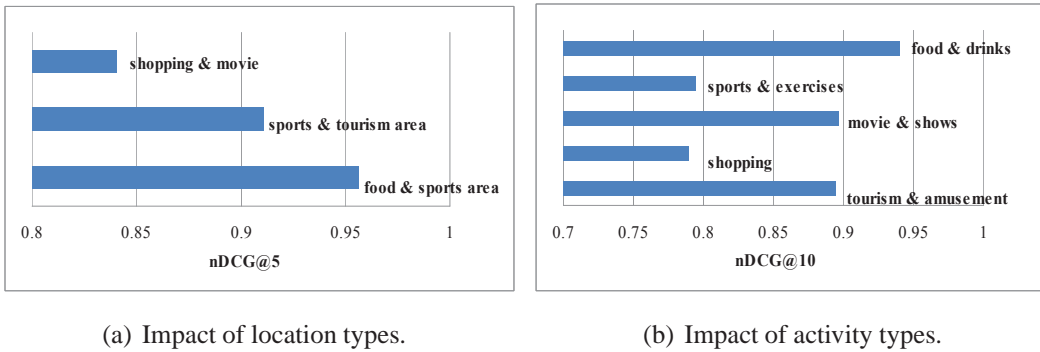


Figure 4.10: Impact of the location/activity types.

Discussions - Impact of the Activity Types to Location Recommendation As expected, for “food and drinks” which happens more often, and “tourism and amusement” which has more user comments, the recommendation results are quite satisfying. For “movies and shows”, the results are still good. For “shopping”, the performance is worse due to less user comments in modeling the location-activity matrix for collaborative filtering. For “sports and exercise”, the performance is also worse than other activities; yet an interesting observation is that, in Figure 4.10(b), our system usually performs well in the “sports” areas. Is there something wrong? By analyzing the data, we find that this is reasonable; because in our system the locations with more comments are more likely to be recommended (i.e. the higher ratings on other activities can propagate to the activity “sports”), but most of these locations are related to food hunting and tourism which are loosely connected with “sports”. As a result, the location recommendation for the activity “sports” is worse than the others.

Discussions - Prediction for New Locations and Activities Our system is based on some GPS data which is limited in size. Therefore, there could be some locations that we do not see in the existing GPS dataset. Similarly, we also only define 5 main activities, what if the user wants to get recommendations for some more-detailed activities, such as “Thai food” instead of food in general? One possible solution could be relying on the data accumulation on the Web. As GPS devices become more popular, more and more GPS data related to more detailed activities in people’s daily life becomes available. Once we have these data, we can keep updating our system. Since our model’s computational complexity is linear to the number of GPS stay points (i.e. the data size), such updates could be easy. Another possible solution is to get such location-activity information from the Web. As there are blogs describing such information (e.g. travel logs), we may mine such knowledge from the Web to enhance our system. However, considering that the blog content can be quite noisy, it is not clear how much it helps. We may leave this as future study.

Experiments with Our Personalized Recommendation System

In this section, we evaluate our personalized recommendation system by comparing with our general recommendation system which does not distinguish between users.

System Performance - Comparison with Baselines We employ five baselines for comparison: user-based CF (UCF), location-based CF (LCF), activity-based CF (ACF), unifying user-location-

activity CF (ULA), high-order singular value decomposition (HOSVD). The first three baselines (i.e. UCF, LCF and ACF) are memory-based methods, adapted from [37] for tensor CF and taking only tensor as input. The fourth baseline, ULA, is also a memory-based method, adapted from [39] to take both the tensor and the additional matrices into consideration. The fifth baseline, HOSVD, is a model-based method used in [164] to model the user-item-tag relations for tag recommendation. It only takes the tensor information as input.

In particular, for UCF, we consider CF on each user-location matrix w.r.t. each activity independently. On each matrix, we follow [37] and use Pearson correlation as the user similarity weights. We find the top N similar users for some target user (with missing entries) and then compute their weighted average to predict the missing entry. Similarly, we have LCF and ACF by considering CF on each location-activity matrix w.r.t. each user individually. In the experiments, we set $N = 4$, since we find that the prediction results do not depend on N significantly.

In ULA, for each missing entry in the tensor, we extract a set of top N_u similar users, top N_l similar locations and top N_a similar activities, and then use the ratings from all these users on the corresponding locations and activities, in a weighted manner to calculate the entry value. Specifically, we adopt the idea of [39] and design the prediction function as

$$\hat{\mathcal{A}}_{i,j,k} = \frac{\sum_{u \in R_i} S_{u,i} \mathcal{A}_{u,j,k}}{4 \sum_u S_{u,i}} + \frac{\sum_{l \in R_j} S_{l,j} \mathcal{A}_{i,l,k}}{4 \sum_l S_{l,j}} + \frac{\sum_{a \in R_k} S_{a,k} \mathcal{A}_{i,j,a}}{4 \sum_a S_{a,k}} + \frac{\sum_{u \in R_i, l \in R_j, a \in R_k} S_{u,l,a} \mathcal{A}_{u,l,a}}{4 \sum_{u,l,a} S_{u,l,a}},$$

where $S_{u,i}$ is the similarity for users i and u learned from the user-user matrix B ; $S_{l,j}$ is the similarity for locations j and l learned from the location-feature matrix C and the user-location matrix E by equally combining the cosine similarities calculated from each; $S_{a,k}$ is the similarity for activities k and a learned from activity-activity matrix D ; $S_{u,l,a}$ is the similarity between $\mathcal{A}_{i,j,k}$ and $\mathcal{A}_{u,l,a}$ for some u, l, a belong to the neighboring sets R_i, R_j, R_k of user i , location j and activity k , respectively. It is designed as

$$S_{u,l,a} = 1 / \sqrt{(1/S_{u,i})^2 + (1/S_{l,j})^2 + (1/S_{a,k})^2}.$$

In the experiments, similar to the previous cases, we also set $N_u = N_l = N_a = 4$.

We report the comparison results in Table 4.7. To have these results, we randomly split 50% of the tensor data for training and hold out the other 50% for testing. For all the comparisons here, we run for five times to generate the mean values and standard deviations of the results. In our model, we set the model parameters $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \lambda_5 = 0.1$ and the low dimension $k = 4$. We will study the impact of the parameters in the next section. For HOSVD, we preserved

30% of the information in the original tensor for dimensionality reduction, as suggested in the experiments of [164]. For evaluation, we employ two metrics; one is RMSE (root mean square error) to measure the tensor reconstruction loss on the hold-out test data. For RMSE, the smaller, the better. The other metric is nDCG (normalized discounted cumulative gain) [34] to measure the ranking results of our retrieved location/activity list. For location recommendation to some user given some activity query, we first rank the locations that were held out in the test data for current user and activity. Then, we compare this ranking with the optimal ranking as suggested by the test data to generate the nDCG value. Finally, we take the average the nDCG values over all the users and activities to output the results in the “ nDCG_{loc} ” column at Table 4.7. Similarly, we have the results for activity recommendation in the “ nDCG_{act} ” column. For nDCG, the larger, the better.

	RMSE	nDCG_{loc}	nDCG_{act}
UCF	0.027 ± 0.006	0.297 ± 0.024	0.807 ± 0.007
LCF	0.009 ± 0.000	0.532 ± 0.021	0.614 ± 0.019
ACF	0.022 ± 0.005	0.408 ± 0.012	0.785 ± 0.006
ULA	0.015 ± 0.003	0.291 ± 0.022	0.799 ± 0.012
HOSVD	0.006 ± 0.001	0.390 ± 0.021	0.913 ± 0.004
UCLAF	0.006 ± 0.001	0.599 ± 0.036	0.959 ± 0.009

Table 4.7: Comparison with baselines, by “mean \pm std”.

As we can see from the table, our model consistently outperforms the other baselines, showing the effectiveness of our modeling with tensor and incorporating additional information for collaborative location and activity recommendations. We also note that the nDCG for activity recommendation is usually higher than that for location recommendation. This is because the number of activities is much smaller than that of locations for ranking, especially when we measure the rankings over some part of the activities/locations on the test data. Besides, we find that ULA does not necessarily deliver better results. This implies that our designed prediction function does not model the data characteristics perfectly, thus encouraging us to study more sophisticated memory-based methods for comparison. Also note that, the nDCG values shown in the table are not necessarily close to our previous results in [6], as they are tested with different datasets and experimental conditions.

System Performance - Personalized vs. General Recommendation We evaluated our personalized recommendation system and compared it with our general recommendation system. The results are give in Table 4.8. As can be seen, the personalized recommendation results are con-

sistently better than the general recommendation results on all three evaluation metrics. Here, the model parameter α is set as 0.01. We investigate the impact of this model parameter later. Also remember that the $nDCG_{loc}$ and $nDCG_{act}$ are measured by ranking all the existing entries in the test data, rather than just on top K locations or activities (which is used in the user study based evaluation considering the human effort).

	$nDCG_{loc}$	$nDCG_{act}$	$RMSE$
General Recommendation	0.59 ± 0.03	0.93 ± 0.01	0.06 ± 0.00
Personalized Recommendation	0.62 ± 0.03	0.95 ± 0.00	0.01 ± 0.00

Table 4.8: Personalized vs. General Recommendation

Investigation into Our System - Impact of the User Number To evaluate the impact of the user number on our personalized recommendation system, we randomly picked a fixed set of 25 users as testing users, and varied the number of users in training. The results are shown in Figure 4.11. In general, as the number of (training) user increases, the performances in terms of $nDCG_{loc}$ and $nDCG_{act}$ increase. We notice that, there is some minor decrease when the user number is either too small or too big. A possible reason for this may be overfitting. When the user number is small, the increase of user number may bring more data and thus improves the performance. However, as the user number keeps increasing, the model’s unknown variable number also increases and the data may gradually become insufficient until there are enough users again. Therefore in this case, the performance could decrease. We also notice that, compared with $nDCG$ metrics, RMSE does not change that much as the user number changes. This may be because the RMSE is based on the square sum of the differences between the reconstructed tensor and the test data tensor, and it is less sensitive than ranking used in $nDCG$ given some CF model.

Impact of the Model Parameters We first study the impact of $\lambda_1 \sim \lambda_5$. Recall the objective function in Eq.(4.1), where each λ_i ($i = 1, \dots, 4$) controls the contribution for each additional information. We vary each λ_i from 0 to 10, and report the nDCG values for location recommendation in Figure 4.12(a) and activity recommendation in Figure 4.12(b). As can be seen from both plots, in general, using the additional information (when λ_i ’s are larger than 0) could be better than not using it (when λ_i ’s equal to 0). Besides, our model is not sensitive to most of these parameters. This implies that, the information in the additional matrices of this dataset could be limited, and meanwhile, our model is robust. We also notice that, as λ_2 increases, the nDCG for location recommendation decreases quickly. This could be because the location-feature matrix is noisy in

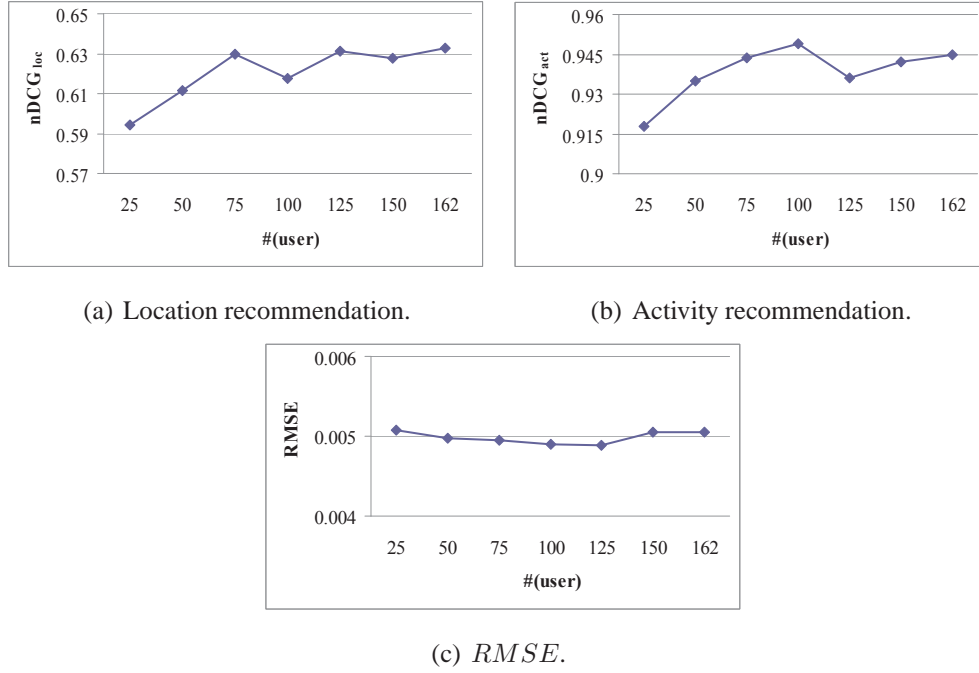


Figure 4.11: Impact of the user number.

data extraction from the POI database, as the λ_2 increases, the impact of the noise becomes bigger. As this location-feature matrix is not directly related to activities, we don't observe similar performance decrease in activity recommendation.

In Figure 4.12(c), we vary the low dimension k from one to five (as the minimal dimension in the tensor is five, i.e. the number of activities), and report the $nDCG$ results. As can be seen, our model's performances, for both location recommendation and activity recommendation, are insensitive to the change of k . Due to space limit, we didn't report the results under RMSE for all the parameters (i.e. λ_i 's and k), but we observed the similar patterns from experiments.

4.6 Summary

In this chapter, we study how to mine knowledge from real-world GPS data and retrieve relevant mobile information to answer two typical questions in our daily life. The first question is, if we want to do something, where shall we go? This question corresponds to location recommendation. The second question is, if we visit some place, what can we do there? This question corresponds

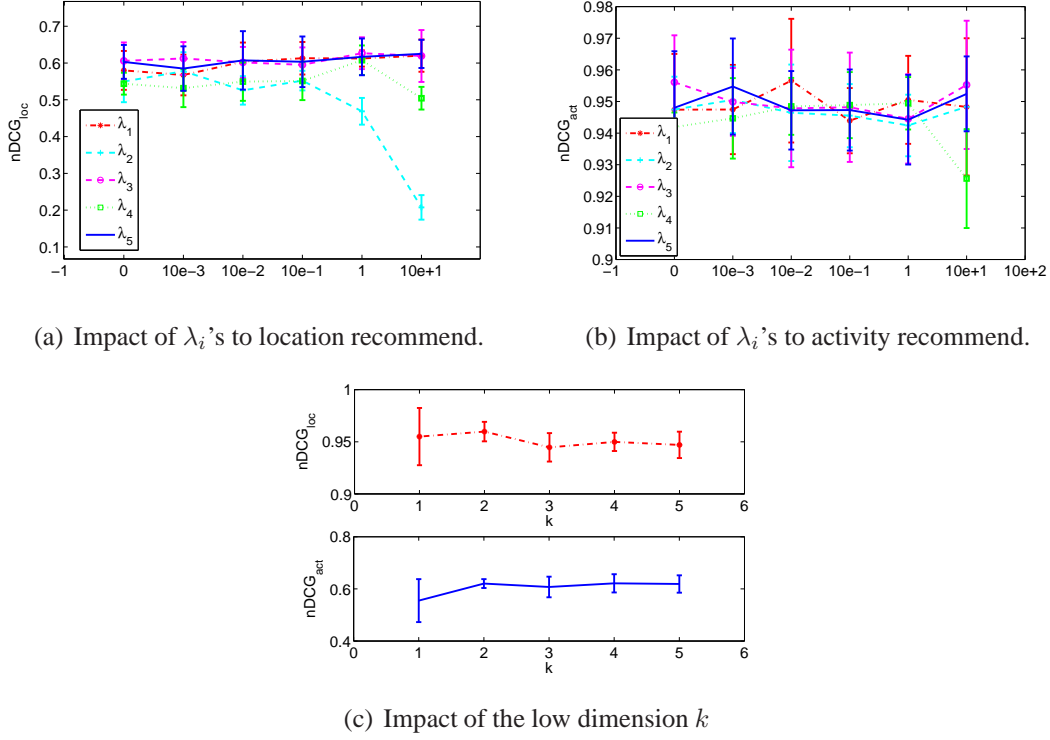


Figure 4.12: Impact of model parameters.

to activity recommendation. We show that these two questions are inherently related, as they can be seen as a ranking problem over a location-activity rating matrix. Because the location-activity matrix is very sparse in practice, we proposed to exploit other information, including the location features and the activity-activity correlations from various information sources, to address the data sparsity challenge. We provided a collaborative location and activity filtering approach based on collective matrix factorization to take these information sources as inputs and train a location and activity recommender. Both PC and hand-held device users can access our recommender through the Web to get recommendations for better trip planning, etc. We also further extend our general collaborative location and activity recommendation system to personalized case by encoding the user dimension in modeling. We extensively evaluated our system on a real-world, large GPS dataset. Our general recommendation system has 7% improvement on activity recommendation and over 20% improvement on location recommendation over the simple baseline without exploiting any additional information. And the corresponding personalized recommendation system is shown to further improve the recommendation performance.

In the future, we will consider more external information, such as user social network so that the

experiences from friend (similar) users can contribute more in retrieving recommendation results. We are also interested in incorporating the time or sequence information of the trajectories to provide more constraints in the recommendations.

Chapter 5

A Unified Learning Framework

In this chapter, we are interested in designing a unified learning framework for our behavior recognition problems. We wish the designed unified learning framework to be able to cover all our existing solutions and also be flexible to encode the possible inter-task relationships. Therefore, we first discuss the inter-task relationships among location estimation, activity recognition and mobile recommendation. Then, we summarize our ideas of using transfer learning to incorporate auxiliary data to address the data sparsity challenges, and put forward a unified learning framework. We finally show how to instantiate the learning framework to each previous chapter’s solution.

5.1 Relationships among Location Estimation, Activity Recognition and Mobile Recommendation

Let us revisit the hierarchy of our three interested problems in sensor-based behavior recognition in Figure 5.1. our work is focused on getting the location and activity information from the sensor information, and then using it for mobile recommendation. The information flow is one-way: from sensor to locations and activities, then to mobile recommendation. We can also foresee more relationships among location estimation, activity recognition and mobile recommendation. For example, location estimation can be useful to activity recognition. First, location can be one type of activity, indicating that some user visited some place. This is also well known as “check-in” activity in many location-based services such as Foursquare and Facebook places. Second, location

can also provide some context for activity recognition, therefore knowing the location information can help to infer the activity. Liao et al. analyzed the GPS location information to predict the user's outdoor activities such as working or visiting friends [50]. On the contrary, knowing the activity information can also help to understand the user's location. For example, if we know a user is having a class, then we can expect the user is very likely to be in the school. Besides, mobile recommendation can also have some feedback to location estimation and activity recognition. Informative mobile recommendation can lead the mobile users to explore new interesting locations and activities. Therefore, it helps to accumulate more sensor data for location estimation and activity recognition. This closes the loop of using sensor data to predict a user's locations, activities, and finally interests.

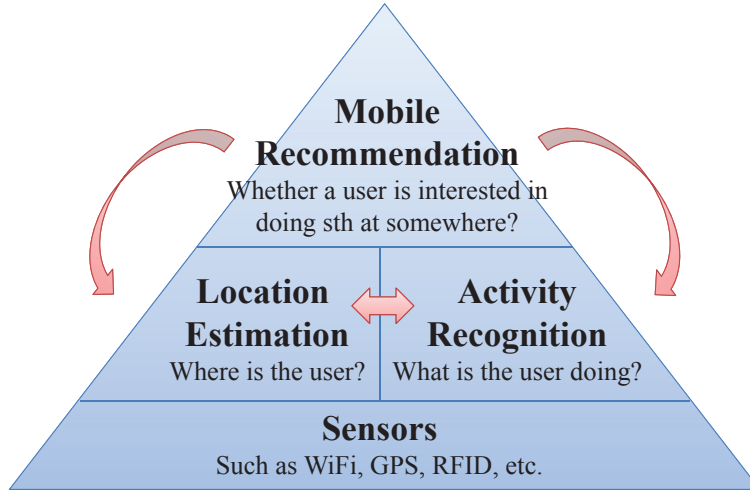


Figure 5.1: Sensor-based behavior recognition revisited.

The Unified Learning Framework As the key to address the data sparsity problems in mobile computing becomes how to discover those informative auxiliary data from different contexts and how to further well incorporate them through some well-designed knowledge share in learning, we may put forward a possible learning framework to summarize all our solutions:

$$\min \underbrace{\mathcal{L}(D_{tar}; \theta_{tar}) + \alpha \cdot \mathcal{L}(D_{aux}; \theta_{aux})}_{loss} - \underbrace{\beta \cdot S(\mathcal{D}_{tar}, \mathcal{D}_{aux}; \theta_{tar}, \theta_{aux})}_{knowledge\ share} + \underbrace{\gamma \cdot \Omega(\theta_{tar}, \theta_{aux})}_{regularization}, \quad (5.1)$$

where D denotes the data, and θ denotes the model. The first term measures the loss on training tasks, which can be in form of classification errors [16], regression errors [11], negative likelihood [61]. In the context of location estimation, this loss can be localization errors. In the context of

activity recognition, this loss can be the recognition errors. In the context of mobile recommendation, this loss can be the prediction errors on the training data. In general, we want to minimize this loss term so as to get accurate localization/recognition/recommendation models. The second term enforces the knowledge share between target data and auxiliary data. In different problems, such knowledge share terms can be in various forms. For example, in location estimation, we can require the different devices or time periods to share similar model parameters. In activity recognition, we can exploit the user clusters so that making similar users to share the data together for recognition. In mobile recommendation, except exploiting the user clusters, we can also exploit the location clusters, together with some other meta information such as location features, activity-activity correlations, for prediction. In general, we want to maximize this knowledge share term so that we can maximally utilize the information from these auxiliary data. The third term is a regularization term to prevent overfitting. Real numbers α , β and γ are the trade-off parameters. We show how to instantiate this learning framework to our each previous chapter's solution in details later.

Our above unified learning framework is also flexible to encode the inter-task relationships as we discussed before. Let us consider the data D , which can be easily instantiated to cover different types of inputs for different behavior recognition tasks. If we want to encode the relationships between location estimation and activity recognition, we can, for example, further utilize the location data as part of the inputs D_{tar} and D_{aux} for activity recognition. Similarly, we can also use the activity data as part of the inputs D_{tar} and D_{aux} for location estimation. In addition, we can take the mobile recommendation results as D_{tar} and D_{aux} for either location estimation or activity recognition.

5.2 Instantiation of The Unified Learning Framework

We need to exploit different instantiations of this learning framework for different data sparsity problems in our sensor-based behavior recognition.

Instantiation to Location Estimation in Chapter 2: When the target domain data D_{tar} and the auxiliary domain data D_{aux} are both location data, then our unified learning framework corresponds to learning with limited data in location estimation.

- *Device-dependent data sparsity in location estimation.* Our latent multi-task learning solution in Eq.(2.1) essentially instantiates the learning framework in Eq.(5.1):

$$\begin{aligned}
& \min J(\mathbf{w}_0, \mathbf{v}_t, \xi_{it}, \xi_{it}^*, b, \varphi_t) \\
& = \underbrace{\sum_{t=1}^T \pi_t \sum_{i=1}^{n_t} (\xi_{it} + \xi_{it}^*)}_{\text{loss}} + \underbrace{\frac{\lambda_1}{T} \sum_{t=1}^T \|\mathbf{v}_t\|^2}_{\text{knowledge share}} + \underbrace{\lambda_2 \|\mathbf{w}_0\|^2 + \frac{\lambda_3}{T} \sum_{t=1}^T \Omega(\varphi_t)}_{\text{regularization}} \\
& s.t. \quad \begin{cases} y_{it} - (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) - b \leq \varepsilon + \xi_{it} \\ (\mathbf{w}_0 + \mathbf{v}_t) \cdot \varphi_t(\mathbf{x}_{it}) + b - y_{it} \leq \varepsilon + \xi_{it}^* \\ \xi_{it}, \xi_{it}^* \geq 0 \end{cases}
\end{aligned}$$

As described in Section 2.3, the *loss* term measures the localization errors. The *knowledge share* term regularizes the dissimilarity among the task hypotheses in the latent feature space $\varphi_t(\mathbf{x})$. The *regularization* term maximizes the margins of the learned classifiers and constrains the complexity of mapping function φ_t .

- *Time-dependent data sparsity in location estimation.* Our transferred Hidden Markov Model in Section 2.4 can be seen as an instantiation of the learning framework as:

$$\begin{aligned}
& \min \underbrace{[P(T_{tar}|\theta_{tar}) + P(T_{aux}|\theta_{aux})]}_{\text{loss}} \\
& \quad - \underbrace{[P(D_{tar}^{\notin L}|D_{tar}^{\in L}, \theta_{tar}, \alpha) + P(D_{aux}^{\notin L}|D_{aux}^{\in L}, \theta_{aux}, \alpha)]}_{\text{knowledge share}} + \underbrace{\gamma \cdot \Omega(\alpha)}_{\text{regularization}} \tag{5.2}
\end{aligned}$$

where in the *loss* term, $P(T|\theta)$ measures the data likelihood of traces T given some Hidden Markov Model θ . $D_{aux}^{\in L}$ is the data collected from the reference points L at the auxiliary time period, and $D_{tar}^{\in L}$ is the data at the target time period. $D_{aux}^{\notin L}$ and $D_{tar}^{\notin L}$ are the data from those non-reference-point locations. In the *knowledge share* term, $P(D^{\notin L}|D^{\in L}, \theta, \alpha)$ measures the likelihood of data $D^{\notin L}$, which can be predicted by data $D^{\in L}$ using some shared signal correlation model α . As defined in Eq.(2.6), α is a multiple linear regression model, which can be shown to absorb the *regularization* term $\Omega(\alpha)$ naturally.

Because we cannot afford keeping going through the whole environment to collect data as time changes, we do not have $D_{tar}^{\notin L}$. Therefore, rather than directly solving Eq.(5.2), we employ a three-stage approach as shown in Section 2.4. First, we learn the signal correlation model α by applying regression analysis at the auxiliary time period 0, and thus we can get an estimation of the model parameter θ_{aux} . Then, we rebuild the radio map at the target time

period t (i.e. target time period) using some up-to-date signal data $D_{tar}^{\in L}$. In this way, we can get some estimation of the new model parameter θ_{tar} based on θ_{aux} . Finally, we use EM on unlabeled trace data T_{tar} at time t to update the model θ_{tar} .

Instantiation to Activity Recognition in Chapter 3: When the target domain data D_{tar} and the auxiliary domain data D_{aux} are both activity data, then our unified learning framework corresponds to learning with limited data in activity recognition.

- *User-dependent data sparsity in activity recognition.* Our proposed collaborative activity recognition model, as shown in Eq.(3.1), aims to maximize the data likelihood by introducing latent user factors as

$$\max P(a, u, f, t) = \underbrace{\sum_{Z_u, Z_f, Z_t} P(a, u, f, t, Z_u, Z_f, Z_t)}_{(negative) \text{ loss}}$$

where the *knowledge share* is achieved by modeling the user groups Z_u :

$$P(a, u, f, t, Z_u, Z_f, Z_t) = \underbrace{P(a|Z_u, Z_f, Z_t)P(Z_u)P(u|Z_u)}_{\text{knowledge share}} P(Z_f)P(f|Z_f)P(Z_t)P(t|Z_t).$$

- *Activity-dependent data sparsity in activity recognition.* Our cross-domain activity recognition model tries to borrow the sensor data from auxiliary activities, and generate some weighted pseudo training data for the target activities. Therefore, the data terms in Eq.(3.9) correspond to the *knowledge share*, as shown below:

$$\begin{aligned} \min_{\mathbf{w}^{vv'}, b^{vv'}, \xi_{vv'}} & \underbrace{\frac{1}{2} \|\mathbf{w}^{vv'}\|}_{\text{regularization}} + C_v \underbrace{\sum_{y=a_v} \xi_{vv'}}_{\text{loss}} + C_{v'} \sum_{y=a_{v'}} \xi_{vv'} \\ s.t. & \underbrace{\mathbf{w}^{vv'} \cdot \phi(\mathbf{x}) + b^{vv'}}_{\text{knowledge share}} \geq 1 - \xi_{vv'}, \quad \text{if } y = a_v, \\ & \underbrace{\mathbf{w}^{vv'} \cdot \phi(\mathbf{x}) + b^{vv'}}_{\text{knowledge share}} \leq -1 + \xi_{vv'}, \quad \text{if } y = a_{v'}, \\ & \xi_{vv'} \geq 0. \end{aligned}$$

The *loss* term and the *regularization* term are similar to the above-mentioned latent multi-task learning solution.

Instantiation to Mobile Recommendation in Chapter 4: When the target domain data D_{tar} and the auxiliary domain data D_{aux} are both the mobile users' location and activity history data, then our unified learning framework corresponds to learning with limited data in mobile recommendation.

- *Crowd-dependent data sparsity in mobile recommendation.* In mobile recommendation, we have the general recommendation model as an instantiation of the framework as:

$$\begin{aligned} \mathcal{L}(U, V, W) = & \underbrace{\frac{1}{2} \|I \circ (X - UV^T)\|_F^2}_{loss} \\ & + \underbrace{\frac{\lambda_1}{2} \|Y - UW^T\|_F^2 + \frac{\lambda_2}{2} \|Z - VV^T\|_F^2}_{(negative) \text{ knowledge share}} + \underbrace{\frac{\lambda_3}{2} (\|U\|_F^2 + \|V\|_F^2 + \|W\|_F^2)}_{regularization}, \end{aligned}$$

where the *loss* term measures the matrix reconstruction error, the *knowledge share* term transfers the information from the location features and the activity correlations, and the *regularization* term adds constraints on the model parameters. Similarly, we can have the framework instantiation as a personalized mobile recommendation model:

$$\begin{aligned} \mathcal{L}(X, Y, Z, U) = & \underbrace{\frac{1}{2} \|\mathcal{A} - \llbracket X, Y, Z \rrbracket\|^2}_{loss} \\ & + \underbrace{\frac{\lambda_1}{2} \text{tr}(X^T L_B X) + \frac{\lambda_2}{2} \|C - YU^T\|^2 + \frac{\lambda_3}{2} \text{tr}(Z^T L_D Z) + \frac{\lambda_4}{2} \|E - XY^T\|^2}_{(negative) \text{ knowledge share}} \\ & + \underbrace{\frac{\lambda_5}{2} (\|X\|^2 + \|Y\|^2 + \|Z\|^2 + \|U\|^2)}_{regularization}. \end{aligned}$$

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we study the sensor-based human behavior recognition, which is an important topic in ubiquitous computing and artificial intelligence. In human behavior recognition, we are interested in three problems, the WiFi-based location estimation, the sensor-based activity recognition and the GPS-based mobile recommendation. For each problem, we review the background knowledge and related work; then we point out several real-world data sparsity challenges in solving the problem.

In particular, for WiFi-based location estimation, we address the context-dependent data sparsity challenges w.r.t. varying signal distributions over devices and time periods. We developed two transfer learning solutions that can address the data distribution differences in knowledge share. For the first solution, we propose a latent multi-task learning algorithm which can make use of the auxiliary mobile device's data to help training the localization model in a target device. For the second solution, we propose a transferred hidden Markov model which can make use of the out-of-date signal data to help update the localization model from time to time. We test our solutions with real-world data sets, and show that we can greatly reduce the labeled data required for the new device or new time period, while preserving the localization performance.

For sensor-based activity recognition, we study the user-dependent data sparsity challenge indi-

cating that each user has very limited activity data for training, and the activity-dependent data sparsity challenge indicating that some activities may have very limited or even no data for training. We address the user-dependent data sparsity challenge by proposing a collaborative activity recognition algorithm. The algorithm considers transfer learning with cross user task spaces. With our algorithm, each user can benefit from other similar users' data and meanwhile maintain a personalized activity recognizer. We test this algorithm with some real-world WiFi activity data set, and show to outperform those baselines that either pool all the users' data together or treat each user independently. We address the activity-dependent data sparsity challenge by developing a cross-domain activity recognition algorithm. It considers transfer learning with cross label spaces. Our algorithm uses the Web to measure the semantic similarities among activities, and thus generates a set of weighted pseudo training data for the activities without data to train the classifier. We test our algorithm with several real-world sensor-based activity recognition data sets, and show to be able to obtain reasonable recognition performances even when there are no data for some activities that we want to recognize.

For GPS-based mobile recommendation, we show to parse the trajectory data for mobile recommendation. To address the crowd-dependent data sparsity challenge when there are limited user-location-activity annotations, we develop two collaborative filtering models, which are able to formulate the interactions among users, locations and activities, as well as their auxiliary information. In general mobile recommendation, we pool the crowd's history data together and construct a location-activity matrix indicating the popularity levels of locations and activities. Due to the sparseness, we apply collective matrix factorization to predict the missing entries in the matrix. We use the auxiliary information such as location features and activity correlations to help obtain better latent location factors and latent activity factors. In personalized mobile recommendation, we model the user-location-activity tensor and design a collective tensor decomposition solution to predict the missing entries. In addition to the existing auxiliary information used in general recommendation, we also exploit some new information such as user-user similarities and location popularity in prediction. Both of our proposed mobile recommendation algorithms consider transfer learning with collaborative filtering from multiple information sources. We test our algorithms in a real-world GPS data set collected for more than 2 years, and show to be able to give reasonable recommendation results for both general and personalized cases.

6.2 Future Work

In addition to the possible future work we have mentioned in each chapter, we will continue to explore some other possible research opportunities along the following directions:

- *Calibration-free localization.* Our transfer learning solutions have shown to help reduce the human labeling cost for the new context such as device or time period. However, we still require some labeled data from the existing context; in other words, we still need the user labeling in order to build some basic localization model. How to totally get rid of the human labeling in order to build a localization system is still an open problem. One possible solution is to utilize the location topology constraints as well as the user motion constraints. In localization, as the high-dimensional signal space essentially corresponds to a low-dimensional location embedding, it is possible for us to align this embedding space with the location topology obtained from a floor plan by considering some constraints. Another interesting idea is to model user motion as these constraints like Ferris et al.’s work [61]. If we can use these topology information and user movement constraints, we are able to provide totally calibration-free localization solutions. Besides, fusing the signal propagation model and the learning-based model for localization is also promising [76], as it can benefit from knowing the locations of access points and using the physical signal propagation properties.
- *Social sensor for activity recognition and mobile recommendation.* Nowadays the physical world where user’s physical location and activity behaviors happen, is basically disjoint with the virtual world where user’s online browsing behaviors happen. Their causal relation links are somehow missing. This gap gives us more room to think about how to completely understand a user’s behavior. First, “sensor” as the ambient information detection media, can be either in the form of hardware sensors installed in the physical environment, or in the form of online media such as social media, Web log, etc. All this information, as long as it is related to physical/local behaviors, can be used for mobile intelligence. Social media such as Facebook, Twitter, Foursquare, become very popular recently because of their power in expressing the ambient awareness. These social media give the people a platform to share their ideas, feelings and most importantly, their activities. We can think of these social media as social “sensors”, which bring rich context and content information for user activity recognition. It is a great opportunity for us to use these self-generated data to understand where the users visit [165], what they are doing [166, 167], what they like [168] and how they influ-

ence each other [169, 170]. Introducing the Web data can greatly relieve the data sparsity in both activity recognition and mobile recommendation, but it also brings the large-scale data computation problem. Some cloud computing solution could be an option. Besides, as the social media data are noisy, more careful data cleaning and information retrieval are vital.

Bibliography

- [1] J. J. Pan and Q. Yang, “Co-localization from labeled and unlabeled data using graph laplacian,” in *Proc. of the 20th International Joint Conference on Artificial Intelligence, IJCAI '07*, pp. 2166–2171, 2007.
- [2] S. S. Intille, K. Larson, E. M. Tapia, J. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson, “Using a live-in laboratory for ubiquitous computing research,” in *Proc. of the 4th International Conference on Pervasive Computing, Pervasive '06*, pp. 349–365, 2006.
- [3] Y. Zheng, X. Xie, and W.-Y. Ma, “Geolife: A collaborative social networking service among user, location and trajectory,” in *IEEE Database Engineering Bulletin*, 2010.
- [4] J. J. Pan, *Learning-based Localization in Wireless and Sensor Networks*. PhD thesis, The Hong Kong University of Science and Technology, 2007.
- [5] D. J. Patterson, D. Fox, H. A. Kautz, and M. Philipose, “Fine-grained activity recognition by aggregating abstract object usage,” in *Proc. of the 9th IEEE International Symposium on Wearable Computers, ISWC '05*, pp. 44–51, 2005.
- [6] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang, “Collaborative location and activity recommendations with gps history data,” in *Proc. of the 19th International World Wide Web Conference, WWW '10*, (New York, NY, USA), ACM, 2010.
- [7] N. D. Lane, D. Lymberopoulos, F. Zhao, and A. T. Campbell, “Hapori: context-based local search for mobile phones using community behavioral modeling and similarity,” in *Proc. of the 12th ACM International Conference on Ubiquitous Computing, UbiComp '10*, (New York, NY, USA), pp. 109–118, ACM, 2010.

- [8] N. Eagle and A. Pentland, “Eigenbehaviors: Identifying structure in routine,” in *Behavioral Ecology and Sociobiology*, pp. 1057–1066, May 2009.
- [9] P. Bahl and V. N. Padmanabhan, “RADAR: An in-building RF-based user location and tracking system,” in *Proc. of IEEE International Conference on Computer Communications, INFOCOM ’00*, pp. 775–784, 2000.
- [10] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, D. S. Wallach, and G. Marceau, “Robotics-based location sensing using wireless ethernet,” in *Proc. of the 8th Annual International Conference on Mobile Computing and Networking, MobiCom ’02*, (New York, NY, USA), pp. 227–238, 2002.
- [11] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, “Landmarc: indoor location sensing using active rfid,” in *Proc. of IEEE International Conference on Pervasive Computing and Communications, PerCom ’03*, (Dallas, TX, USA), 2003.
- [12] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. E. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. N. Schilit, “Place lab: Device positioning using radio beacons in the wild,” in *Proc. of the 3rd International Conference on Pervasive Computing, Pervasive ’05*, pp. 116–133, 2005.
- [13] A. Savvides, C.-C. Han, and M. B. Strivastava, “Dynamic fine-grained localization in ad-hoc networks of sensors,” in *Proc. of the 7th Annual International Conference on Mobile Computing and Networking, MobiCom ’01*, (New York, NY, USA), pp. 166–179, ACM, 2001.
- [14] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, “Range-free localization schemes for large scale sensor networks,” in *Proc. of the 9th Annual International Conference on Mobile Computing and Networking, MobiCom ’03*, (New York, NY, USA), pp. 81–95, ACM, 2003.
- [15] A. Goldsmith, *Wireless Communications*. Cambridge: Cambridge University Press, 2005.
- [16] P. Krishnan, A. S. Krishnakumar, W.-H. Ju, C. Mallows, and S. Ganu, “A system for lease: Location estimation assisted by stationery emitters for indoor rf wireless networks,” in *Proc. of IEEE International Conference on Computer Communications, INFOCOM ’04*, 2004.

- [17] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello, “Bayesian filtering for location estimation,” in *Proc. of IEEE Pervasive Computing and Communication*, vol. 02 of *PerCom '03*, pp. 24–33, 2003.
- [18] T. Roos, P. Myllymäki, and H. Tirri, “A statistical modeling approach to location estimation,” *IEEE Transactions on Mobile Computing*, vol. 1, pp. 59–69, January 2002.
- [19] M. A. Youssef, A. Agrawala, and A. U. Shankar, “Wlan location determination via clustering and probability distributions,” in *Proc. of the First IEEE International Conference on Pervasive Computing and Communications*, PERCOM '03, (Washington, DC, USA), pp. 143–, IEEE Computer Society, 2003.
- [20] J. J. Pan, Q. Yang, H. Chang, and D.-Y. Yeung, “A manifold regularization approach to calibration reduction for sensor-network based tracking,” in *Proc. of the 21st National Conference on Artificial Intelligence*, AAAI '06, (Boston, United States), pp. 988–993, 2006.
- [21] M. E. Pollack, L. E. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos, “Autominder: an intelligent cognitive orthotic system for people with memory impairment,” *Robotics and Autonomous Systems*, vol. 44, no. 3-4, pp. 273–282, 2003.
- [22] J. Yin, Q. Yang, and J. J. Pan, “Sensor-based abnormal human-activity detection,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 17–31, 2007.
- [23] F. Albinali, S. Intille, W. Haskell, and M. Rosenberger, “Using wearable activity type detection to improve physical activity energy expenditure estimation,” in *Proc. of the 12th ACM International Conference on Ubiquitous computing*, UbiComp '10, (New York, NY, USA), pp. 311–320, ACM, 2010.
- [24] H. Kautz, *A Formal Theory of Plan Recognition*. PhD thesis, University of Rochester, 1987.
- [25] H. H. Bui, “A general model for online probabilistic plan recognition,” in *Proc. of the 18th International Joint Conference on Artificial Intelligence*, IJCAI '03, pp. 1309–1318, 2003.
- [26] M. R. Hodges and M. E. Pollack, “An ‘object-use fingerprint’: The use of electronic sensors for human identification,” in *Proc. of the 9th International Conference on Ubiquitous Computing*, UbiComp '07, pp. 289–303, 2007.

- [27] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, and L. Wixson, “A system for video surveillance and monitoring,” Tech. Rep. CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, 5 2000.
- [28] G. Medioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia, “Event detection and analysis from video streams,” *IEEE Transaction Pattern Analysis and Machine Intelligence*, vol. 23, pp. 873–889, August 2001.
- [29] S. Hongeng, R. Nevatia, and F. Bremond, “Video-based event recognition: activity representation and probabilistic recognition methods,” *Computer Vision Image Understanding*, vol. 96, pp. 129–162, November 2004.
- [30] H. Zhong, J. Shi, and M. Visontai, “Detecting unusual activity in video,” in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition, CVPR ’04*, (Washington, DC, USA), pp. 819–826, IEEE Computer Society, 2004.
- [31] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *Proc. of the 14th International Conference on Computer Communications and Networks*, (Washington, DC, USA), pp. 65–72, IEEE Computer Society, 2005.
- [32] M.-H. Park, J.-H. Hong, and S.-B. Cho, “Location-based recommendation system using bayesian user’s preference model in mobile devices,” in *Proc. of Ubiquitous Intelligence and Computing, UIC ’07*, pp. 1130–1139, 2007.
- [33] Y. Takeuchi and M. Sugimoto, “Cityvoyager: an outdoor recommendation system based on user location history,” in *Proc. of Ubiquitous Intelligence and Computing, UIC ’06*, pp. 625–636, 2006.
- [34] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, “Mining interesting locations and travel sequences from gps trajectories,” in *Proc. of the 18th International Conference on World Wide Web (WWW’09)*, pp. 791–800, 2009.
- [35] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” *IEEE Transaction on Knowledge and Data Engineering*, vol. 17, pp. 734–749, June 2005.

- [36] R. J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in *Proc. of the 5th ACM Conference on Digital Libraries*, (New York, NY, USA), pp. 195–204, ACM, 2000.
- [37] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering,” in *Proc. of the 22nd Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR ’99, pp. 230–237, ACM, 1999.
- [38] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in *Proc. of the 10th International Conference on World Wide Web*, WWW ’01, (New York, NY, USA), pp. 285–295, ACM, 2001.
- [39] J. Wang, A. P. de Vries, and M. J. T. Reinders, “Unifying user-based and item-based collaborative filtering approaches by similarity fusion,” in *Proc. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’06, pp. 501–508, 2006.
- [40] N. Srebro and T. Jaakkola, “Weighted low-rank approximations,” in *Proc. of the 21th International Conference on Machine Learning*, ICML ’03, pp. 720–727, 2003.
- [41] A. P. Singh and G. J. Gordon, “Relational learning via collective matrix factorization,” in *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’08, pp. 650–658, 2008.
- [42] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, pp. 30–37, August 2009.
- [43] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” in *Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’08, (New York, NY, USA), pp. 426–434, ACM, 2008.
- [44] Y. Koren, “Collaborative filtering with temporal dynamics,” in *Proc. of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’09, (New York, NY, USA), pp. 447–456, ACM, 2009.

- [45] S. Rendle and S.-T. Lars, “Pairwise interaction tensor factorization for personalized tag recommendation,” in *Proc. of the third ACM International Conference on Web search and data mining*, WSDM ’10, (New York, NY, USA), pp. 81–90, ACM, 2010.
- [46] S. Rendle, C. Freudenthaler, and S.-T. Lars, “Factorizing personalized markov chains for next-basket recommendation,” in *Proc. of the 19th International Conference on World wide web*, WWW ’10, (New York, NY, USA), pp. 811–820, ACM, 2010.
- [47] A. S. Das, M. Datar, A. Garg, and S. Rajaram, “Google news personalization: scalable online collaborative filtering,” in *Proc. of the 16th International Conference on World Wide Web*, WWW ’07, (New York, NY, USA), pp. 271–280, ACM, 2007.
- [48] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” in *Proc. of the 8th IEEE International Conference on Data Mining*, ICDM ’08, (Washington, DC, USA), pp. 263–272, IEEE Computer Society, 2008.
- [49] J. L. Herlocker, J. A. Konstan, and J. Riedl, “Explaining collaborative filtering recommendations,” in *Proc. of the ACM Conference on Computer Supported Cooperative Work*, CSCW ’00, (New York, NY, USA), pp. 241–250, ACM, 2000.
- [50] L. Liao, D. Fox, and H. A. Kautz, “Location-based activity recognition using relational markov networks,” in *Proc. of the 20th International Joint Conference on Artificial Intelligence*, IJCAI ’05, pp. 773–778, 2005.
- [51] S. N. Patel, K. N. Truong, and G. D. Abowd, “Powerline positioning: A practical sub-room-level indoor location system for domestic use,” in *Proc. of the 8th International Conference on Ubiquitous Computing*, UbiComp ’06, pp. 441–458, Springer, 2006.
- [52] C. Drane, M. Macnaughtan, and C. Scott, “Positioning gsm telephones,” *IEEE Communications Magazine*, vol. 36, pp. 46–54, 1998.
- [53] O. Veljo, V. Alex, L. Anthony, and de Lara Eyal, “Accurate gsm indoor localization,” in *Proc. of the 7th International Conference on Ubiquitous Computing*, UbiComp ’05, pp. 141–158, 2005.
- [54] D. Hähnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, “Mapping and localization with rfid technology,” in *Proc. of the IEEE International Conference on Robotics and Automation*, 2004.

- [55] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, “Practical robust localization over large-scale 802.11 wireless networks,” in *Proc of the 8th Annual International Conference on Mobile Computing and Networking*, MobiCom ’02, (Philadelphia, PA).
- [56] E. Brassart, C. Pegard, and M. Mouaddib, “Localization using infrared beacons,” *Robotica*, vol. 18, no. 2, pp. 153–161, 2000.
- [57] R. Want, A. Hopper, V. F. ao, and J. Gibbons, “The active badge location system,” *ACM Transactions on Information Systems*, vol. 10, no. 1, pp. 91–102, 1992.
- [58] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, “The anatomy of a context-aware application,” *Wireless Network*, vol. 8, no. 2/3, pp. 187–197, 2002.
- [59] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, “The cricket location-support system,” in *Proc. of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom ’00, (New York, NY, USA), pp. 32–43, ACM, 2000.
- [60] M. J. Skibniewski and W.-S. Jang, “Localization technique for automated tracking of construction materials utilizing combined rf and ultrasound sensor interfaces,” in *Proc. of the ASCE International Workshop on Computing in Civil Engineering*, (Pittsburgh, Pennsylvania), July 2007.
- [61] B. Ferris, D. Fox, and N. Lawrence, “Wifi-slam using gaussian process latent variable models,” in *Proc. of International Joint Conferences on Artificial Intelligence*, IJCAI ’07, pp. 2480–2485, 2007.
- [62] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, “Estimating the absolute position of a mobile robot using position probability grids,” in *Proc. of the 13th National Conference on Artificial Intelligence*, AAAI ’96, pp. 896–901, AAAI Press, 1996.
- [63] X. Chai and Q. Yang, “Reducing the calibration effort for location estimation using unlabeled samples,” in *Proc. of IEEE International Conference on Pervasive Computing and Communications*, PerCom ’05, pp. 95–104, 2005.
- [64] L. Liao, D. Fox, J. Hightower, H. Kautz, and D. Schulz, “Voronoi tracking: Location estimation using sparse and noisy sensor data,” in *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

- [65] A. Varshavsky, M. Y. Chen, E. de Lara, J. Froehlich, D. Haehnel, J. Hightower, A. LaMarca, F. Potter, T. Sohn, K. Tang, and I. Smith, “Are gsm phones the solution for localization?,” in *Proc. of the 7th IEEE Workshop on Mobile Computing Systems & Applications*, (Washington, DC, USA), pp. 20–28, IEEE Computer Society, 2006.
- [66] E. P. Stuntebeck, S. N. Patel, T. Robertson, M. S. Reynolds, and G. D. Abowd, “Wideband powerline positioning for indoor localization,” in *Proc. of the 10th International Conference on Ubiquitous Computing*, UbiComp ’08, (New York, NY, USA), pp. 94–103, ACM, 2008.
- [67] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, “Monte carlo localization: Efficient position estimation for mobile robots,” in *Proc. of the National Conference of Artificial Intelligence*, AAAI ’99, pp. 343–349, 1999.
- [68] M. Y. Chen, T. Sohn, D. Chmelev, D. Hähnel, J. Hightower, J. Hughes, A. LaMarca, F. Potter, I. E. Smith, and A. Varshavsky, “Practical metropolitan-scale positioning for gsm phones,” in *Proc. of the 8th International Conference on Ubiquitous Computing*, UbiComp ’06, pp. 225–242, 2006.
- [69] X. Nguyen, M. I. Jordan, and B. Sinopoli, “A kernel-based learning approach to ad hoc sensor network localization,” *ACM Transaction on Sensor Network*, vol. 1, no. 1, pp. 134–152, 2005.
- [70] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz, “Localization from mere connectivity,” in *Proc. of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing*, MobiHoc ’03, (New York, NY, USA), pp. 201–212, ACM, 2003.
- [71] N. Patwari and A. O. Hero, “Adaptive neighborhoods for manifold learning-based sensor localization,” in *Proc. of the IEEE 6th Workshop on Signal Processing Advances in Wireless Communications*, pp. 1098–1102, 2005.
- [72] J. J. Pan, J. T. Kwok, Q. Yang, and Y. Chen, “Accurate and low-cost location estimation using kernels,” in *Proc. of the 19th International Joint Conference on Artificial Intelligence*, IJCAI ’05, pp. 1366–1371, 2005.
- [73] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, and J. Sievänen, “A probabilistic approach to wlan user location estimation,” *International Journal of Wireless Information Networks*, vol. 9, pp. 155–164, July 2002.

- [74] M. A. Youssef, A. Agrawala, and A. U. Shankar, “Wlan location determination via clustering and probability distributions,” in *Proc. of IEEE International Conference on Pervasive Computing and Communications*, PerCom ’03, (Washington, DC, USA), p. 143, IEEE Computer Society, 2003.
- [75] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking,” in *IEEE Transactions of Signal Processing*, vol. 50, pp. 174–188, February 2002.
- [76] J. Letchner, D. Fox, and A. LaMarca, “Large-scale localization from wireless signal strength,” in *Proc. of the 20th National Conference on Artificial Intelligence*, AAAI ’05, pp. 15–20, 2005.
- [77] R. Pan, J. Zhao, V. W. Zheng, J. J. Pan, D. Shen, S. J. Pan, and Q. Yang, “Domain-constrained semi-supervised mining of tracking models in sensor networks,” in *Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’07, (New York, NY, USA), pp. 1023–1027, 2007.
- [78] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann, “Gpps: A gaussian process positioning system for cellular networks,” in *Advances in Neural Information Processing Systems*, NIPS ’04, (Cambridge, MA), 2004.
- [79] B. Ferris, D. Häehnel, and D. Fox, “Gaussian processes for signal strength-based location estimation,” *Robotics: Science and Systems*, August 2006.
- [80] J. M. Wang, D. J. Fleet, and A. Hertzmann, “Gaussian process dynamical models,” in *Advances in Neural Information Processing Systems*, NIPS ’05, (Vancouver, Canada), pp. 1441–1448, 2005.
- [81] M. B. Kjaergaard and C. V. Munk, “Hyperbolic location fingerprinting: A calibration-free solution for handling differences in signal strength (concise contribution),” in *Proc. of the 6th Annual IEEE International Conference on Pervasive Computing and Communications*, PerCom ’08, (Washington, DC, USA), pp. 110–116, IEEE Computer Society, 2008.
- [82] J. Yin, Q. Yang, and L. M. Ni, “Adaptive temporal radio maps for indoor location estimation,” in *Proc. of the 3rd IEEE International Conference on Pervasive Computing and Communications*, PerCom ’05, pp. 85–94, 2005.

- [83] S. J. Pan, J. T. Kwok, Q. Yang, and J. J. Pan, “Adaptive localization in a dynamic wifi environment through multi-view learning,” in *Proc. of the 22d National Conference on Artificial Intelligence, AAAI ’07*, (Vancouver, Canada), pp. 1108–1113, 2007.
- [84] R. Caruana, “Multitask learning,” in *Machine Learning*, vol. 28(1), pp. 41–75, 1997.
- [85] T. Evgeniou and M. Pontil, “Regularized multi-task learning,” in *Proc. of the tenth ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD’04)*, pp. 109–117, ACM, 2004.
- [86] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.
- [87] J. C. Bezdek and R. J. Hathaway, “Convergence of alternating optimization,” *Neural, Parallel & Scientific Computations*, vol. 11, pp. 351–368, December 2003.
- [88] J. Bilmes, “A gentle tutorial on the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models,” Tech. Rep. ICSI-TR-97-021, University of Berkeley, 1997.
- [89] N. Lesh and O. Etzioni, “A sound and fast goal recognizer,” in *Proc. of the 14th International Joint Conference on Artificial Intelligence, IJCAI ’95*, (San Francisco, CA, USA), pp. 1704–1710, Morgan Kaufmann Publishers Inc., 1995.
- [90] E. M. Tapia, S. S. Intille, and K. Larson, “Activity recognition in the home using simple and ubiquitous sensors,” in *Pervasive Computing*, pp. 158–175, 2004.
- [91] T. van Kasteren, G. Englebienne, and B. Kröse, “Recognizing activities in multiple contexts using transfer learning,” in *Proc. of the AAAI Fall 2008 Symposium: AI in Eldercare*, (Washington DC, USA), 2008.
- [92] L. Liao, D. Fox, and H. A. Kautz, “Learning and inferring transportation routines,” in *Artificial Intelligence*, pp. 311–331, 2007.
- [93] L. Bao and S. S. Intille, “Activity recognition from user-annotated acceleration data,” in *Proc. of the 2nd International Conference of Pervasive Computing, Pervasive ’04*, pp. 1–17, 2004.

- [94] M. Csikszentmihalyi and R. Larson, “Validity and reliability of the experience sampling method,” *The Journal of Nervous and Mental Disease*, vol. 175(9), 1987.
- [95] K. R. Wentzel, “Motivation and achievement in adolescence: a multiple goals perspective,” in *Student Perceptions in the Classroom: Causes and Consequences* (D. H. Schunk and J. L. Meece, eds.), pp. 287–306, Erlbaum, Hillsdale, 1992.
- [96] A. J. Elliot and J. M. Harackiewicz, “Approach and avoidance achievement goals and intrinsic motivation: A mediational analysis,” *Journal of Personality and Social Psychology*, vol. 70, no. 3, pp. 461–475, 1996.
- [97] X. Chai and Q. Yang, “Multiple-goal recognition from low-level signals,” in *Proc. of the 20th National Conference on Artificial Intelligence, AAAI ’05*, pp. 3–8, 2005.
- [98] H. yu Wu, C. chun Lian, and J. Y. jen Hsu, “Joint recognition of multiple concurrent activities using factorial conditional random fields,” in *Proc. of the AAAI Workshop on Plan, Activity and Intent Recognition*, 2007.
- [99] D. H. Hu and Q. Yang, “Cigar: Concurrent and interleaving goal and activity recognition,” in *Proc. of the 23rd National Conference on Artificial Intelligence, AAAI ’08*, 2008.
- [100] D. H. Hu, S. J. Pan, V. W. Zheng, N. N. Liu, and Q. Yang, “Real world activity recognition with multiple goals,” in *Proc. of the Tenth International Conference on Ubiquitous Computing, UbiComp ’08*, pp. 30–39, 2008.
- [101] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. S. Intille, “A long-term evaluation of sensing modalities for activity recognition,” in *Proc. of the 9th International Conference on Ubiquitous Computing, UbiComp ’07*, pp. 483–500, 2007.
- [102] A. Bulling, J. A. Ward, H. Gellersen, and G. Tröster, “Eye movement analysis for activity recognition,” in *Proc. of the 11th International Conference on Ubiquitous Computing, Ubicomp ’09*, (New York, NY, USA), pp. 41–50, ACM, 2009.
- [103] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, “Activity recognition based on semi-supervised learning,” in *Proc. of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, RTCSA ’07*, (Washington, DC, USA), pp. 469–475, IEEE Computer Society, 2007.

- [104] T. Huynh, M. Fritz, , and B. Schiele, “Discovery of activity patterns using topic models,” in *Proc of the 10th International Conference on Ubiquitous Computing, UbiComp '08*, 2008.
- [105] K. Farrahi and D. Gatica-Perez, “What did you do today?: discovering daily routines from large-scale mobile data,” in *Proc. of the 16th ACM International Conference on Multimedia, ACM MM '08*, pp. 849–852, 2008.
- [106] T. Duong, D. Phung, H. Bui, and S. Venkatesh, “Efficient duration and hierarchical modeling for human activity recognition,” *Artificial Intelligence*, vol. 173, pp. 830–856, May 2009.
- [107] J. Yin, X. Chai, and Q. Yang, “High-level goal recognition in a wireless lan,” in *Proc. of the 19th National Conference on Artificial Intelligence, AAAI '04*, pp. 578–584, 2004.
- [108] M. Brand, N. Oliver, and A. Pentland, “Coupled hidden markov models for complex action recognition,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR '97*, (Los Alamitos, CA, USA), IEEE Computer Society, 1997.
- [109] D. L. Vail, M. M. Veloso, and J. D. Lafferty, “Conditional random fields for activity recognition,” in *Proc. of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, (AAMAS '07)*, pp. 1–8, 2007.
- [110] S. Sarawagi and W. W. Cohen, “Semi-markov conditional random fields for information extraction,” in *Advances in Neural Information Processing Systems, NIPS '04*, pp. 1185–1192, 2004.
- [111] T. T. Truyen, D. Q. Phung, H. H. Bui, and S. Venkatesh, “Hierarchical semi-markov conditional random fields for recursive sequential data,” in *Advances in Neural Information Processing Systems, NIPS '08*, pp. 1657–1664, 2008.
- [112] C.-C. Lian and J. Y.-j. Hsu, “Probabilistic models for concurrent chatting activity recognition,” in *Proc. of the 21st International Joint Conference on Artificial Intelligence, IJCAI '09*, 2009.
- [113] L. Liao, D. Fox, and H. A. Kautz, “Extracting places and activities from gps traces using hierarchical conditional random fields,” *International Journal of Robotics Research*, vol. 26, no. 1, pp. 119–134, 2007.

- [114] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse, “Activity recognition using semi-markov models on real world smart home datasets,” *Journal of Ambient Intelligence and Smart Environments*, vol. 2, pp. 311–325, August 2010.
- [115] M. Brand, “Coupled hidden markov models for modeling interacting processes,” tech. rep., MIT, 1997.
- [116] P. Natarajan and R. Nevatia, “Coupled hidden semi markov models for activity recognition,” in *Proc. of the IEEE Workshop on Motion and Video Computing*, (Washington, DC, USA), pp. 10–17, IEEE Computer Society, 2007.
- [117] L. Wang, T. Gu, X. Tao, and J. Lu, “Sensor-based human activity recognition in a multi-user scenario,” in *Ambient Intelligence*, vol. 5859, pp. 78–87, 2009.
- [118] M. Perkowitz, M. Philipose, K. Fishkin, and D. J. Patterson, “Mining models of human activities from the web,” in *Proc. of the 13th International Conference on World Wide Web*, WWW ’04, pp. 573–582, 2004.
- [119] S. Wang, W. Pentney, and A. maria Popescu, “Common sense based joint training of human activity recognizers,” in *Proc. of the 20th International Joint Conference on Artificial Intelligence*, IJCAI ’07, pp. 2237–2242, 2007.
- [120] D. Wyatt, M. Philipose, and T. Choudhury, “Unsupervised activity recognition using automatically mined common sense,” in *Proc. of The 20th National Conference on Artificial Intelligence*, AAI ’05, pp. 21–27, 2005.
- [121] O. Etzioni, M. J. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, “Methods for domain-independent information extraction from the web: An experimental comparison,” in *Proc. of the 19th National Conference on Artificial Intelligence*, AAI ’04, pp. 391–398, 2004.
- [122] V. W. Zheng and Q. Yang, “User-dependent latent aspect model for collaborative activity recognition,” in *Proc. of the 22nd International Joint Conference of Artificial Intelligence*, IJCAI ’11, pp. 2085–2090, 2011.
- [123] T. Hofmann and J. Puzicha, “Latent class models for collaborative filtering,” in *Proc. of the 16th International Joint Conference on Artificial Intelligence*, IJCAI ’99, pp. 688–693, 1999.

- [124] L. Si and R. Jin, “Flexible mixture model for collaborative filtering,” in *Proc. of the 20th International Conference on Machine Learning, ICML ’03*, pp. 704–711, 2003.
- [125] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, March 2003.
- [126] T. Evgeniou and M. Pontil, “Regularized multi-task learning,” in *Proc. of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’04*, (Seattle, Washington, USA), pp. 109–117, ACM, August 2004.
- [127] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, “Self-taught learning: Transfer learning from unlabeled data,” in *Proc. of the 24th International Conference on Machine Learning, ICML ’07*, (Corvalis, Oregon, USA), pp. 759–766, June 2007.
- [128] L. Mihalkova, T. Huynh, and R. J. Mooney, “Mapping and revising markov logic networks for transfer learning,” in *Proc. of the 22nd AAAI Conference on Artificial Intelligence, AAAI ’07*, pp. 608–614, 2007.
- [129] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, October 2010.
- [130] M. Sugiyama, S. Nakajima, H. Kashima, P. V. Buenau, and M. Kawanabe, “Direct importance estimation with model selection and its application to covariate shift adaptation,” in *Advances in Neural Information Processing Systems, NIPS ’08*, MIT Press, 2008.
- [131] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [132] K. Borgwardt, A. Gretton, M. Rasch, H. Kriegel, B. Schölkopf, and A. Smola, “Integrating structured biological data by kernel maximum mean discrepancy,” in *Proc. of the 14th International Conference on Intelligent Systems for Molecular Biology*, pp. 49–57, 2006.
- [133] T. van Kasteren, A. K. Noulas, G. Englebienne, and B. J. A. Kröse, “Accurate activity recognition in a home setting,” in *Proc. of the Tenth International Conference on Ubiquitous Computing, UbiComp ’08*, pp. 1–9, 2008.
- [134] M. Porter, “An algorithm for suffix stripping,” *Program*, vol. 14, no. 3, pp. 130–137, 1980.

- [135] E. M. Tapia, T. Choudhury, and M. Philipose, “Building reliable activity models using hierarchical shrinkage and mined ontology,” in *Proc. of the 4th International Conference on Pervasive Computing*, Pervasive ’06, pp. 17–32, 2006.
- [136] M. Balabanović and Y. Shoham, “Fab: content-based, collaborative recommendation,” *Communications of the ACM*, vol. 40, pp. 66–72, March 1997.
- [137] A. Popescul, L. H. Ungar, D. M. Pennock, and S. Lawrence, “Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments,” in *Proc. of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI ’01, (San Francisco, CA, USA), pp. 437–444, Morgan Kaufmann Publishers Inc., 2001.
- [138] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: an open architecture for collaborative filtering of netnews,” in *Proc. of the ACM Conference on Computer Supported Cooperative Work*, CSCW ’94, (New York, NY, USA), pp. 175–186, ACM, 1994.
- [139] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proc. of the 14th Conference on Uncertainty in artificial intelligence*, UAI ’98, pp. 43–52, Morgan Kaufmann, 1998.
- [140] T. Hofmann, “Collaborative filtering via gaussian probabilistic latent semantic analysis,” in *Proc. of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR ’03, (New York, NY, USA), pp. 259–266, ACM, 2003.
- [141] T. Hofmann, “Latent semantic models for collaborative filtering,” *ACM Transaction Information System*, vol. 22, pp. 89–115, January 2004.
- [142] R. Krestel, P. Fankhauser, and W. Nejdl, “Latent dirichlet allocation for tag recommendation,” in *Proc. of the 3rd ACM Conference on Recommender Systems*, RecSys ’09, (New York, NY, USA), pp. 61–68, ACM, 2009.
- [143] W.-Y. Chen, J.-C. Chu, J. Luan, H. Bai, Y. Wang, and E. Y. Chang, “Collaborative filtering for orkut communities: discovery of user latent behavior,” in *Proc. of the 18th International Conference on World wide web*, WWW ’09, (New York, NY, USA), pp. 681–690, ACM, 2009.

- [144] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Advances in Neural Information Processing Systems*, NIPS ’00, pp. 556–562, 2000.
- [145] D. Agarwal, B.-C. Chen, and P. Elango, “Fast online learning through offline initialization for time-sensitive recommendation,” in *Proc. of the 16th ACM SIGKDD International Conference on Knowledge discovery and data mining*, KDD ’10, (New York, NY, USA), pp. 703–712, ACM, 2010.
- [146] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, “Probabilistic memory-based collaborative filtering,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 56–69, 2004.
- [147] R. Jin and L. Si, “A bayesian approach toward active learning for collaborative filtering,” in *Proc. of the 20th Conference on Uncertainty in artificial intelligence*, UAI ’04, (Arlington, Virginia, United States), pp. 278–285, AUAI Press, 2004.
- [148] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang, “One-class collaborative filtering,” in *Proc. of the 8th IEEE International Conference on Data Mining*, ICDM ’08, (Washington, DC, USA), pp. 502–511, IEEE Computer Society, 2008.
- [149] B. M. Marlin and R. S. Zemel, “Collaborative prediction and ranking with non-random missing data,” in *Proc. of the third ACM Conference on Recommender systems*, RecSys ’09, (New York, NY, USA), pp. 5–12, ACM, 2009.
- [150] H. Steck, “Training and testing of recommender systems on data missing not at random,” in *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’10, (New York, NY, USA), pp. 713–722, ACM, 2010.
- [151] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, “Fast nonparametric matrix factorization for large-scale collaborative filtering,” in *Proc. of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’09, (New York, NY, USA), pp. 211–218, ACM, 2009.
- [152] S. A. Robila and L. G. Maciak, “A parallel unmixing algorithm for hyperspectral images,” in *Intelligent Robots and Computer Vision*, 2006.
- [153] K. Kanjani, “Parallel non negative matrix factorization for document clustering,” tech. rep., Texas A & M University, May 2007.

- [154] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, “Large-scale parallel collaborative filtering for the netflix prize,” in *Proc. of the 4th International Conference on Algorithmic Aspects in Information and Management*, AAIM ’08, (Berlin, Heidelberg), pp. 337–348, Springer-Verlag, 2008.
- [155] C. Liu, H.-c. Yang, J. Fan, L.-W. He, and Y.-M. Wang, “Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce,” in *Proc. of the 19th International Conference on World wide web*, WWW ’10, (New York, NY, USA), pp. 681–690, ACM, 2010.
- [156] N. N. Liu and Q. Yang, “Eigenrank: a ranking-oriented approach to collaborative filtering,” in *Proc. of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’08, (New York, NY, USA), pp. 83–90, ACM, 2008.
- [157] N. N. Liu, M. Zhao, and Q. Yang, “Probabilistic latent preference analysis for collaborative filtering,” in *Proceeding of the 18th ACM Conference on Information and knowledge management*, CIKM ’09, (New York, NY, USA), pp. 759–766, ACM, 2009.
- [158] B. Li, Q. Yang, and X. Xue, “Transfer learning for collaborative filtering via a rating-matrix generative model,” in *Proc. of the 26th Annual International Conference on Machine Learning*, ICML ’09, (New York, NY, USA), pp. 617–624, ACM, 2009.
- [159] B. Cao, N. N. Liu, and Q. Yang, “Transfer learning for collective link prediction in multiple heterogenous domains,” in *Proc. of the 27th International Conference on Machine Learning*, ICML ’10, pp. 159–166, 2010.
- [160] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, “Transfer learning in collaborative filtering for sparsity reduction,” in *Proc. of the 24th AAAI Conference on Artificial Intelligence*, AAAI ’10, 2010.
- [161] T. Horozov, N. Narasimhan, and V. Vasudevan, “Using location for personalized poi recommendations in mobile environments,” in *Proc. of the International Symposium on Applications on Internet*, pp. 124–129, 2006.
- [162] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang, “Collaborative filtering meets mobile recommendation: A user-centered approach,” in *Proc. of the 24th AAAI Conference on Artificial Intelligence*, AAAI ’10, (Atlanta, Georgia, USA), pp. 236–241, 2010.

- [163] A. Cichocki, R. Zdunek, A. H. Phan, and S. ichi Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multiway Data Analysis and Blind Source Separation*. Wiley, 2009.
- [164] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, “Tag recommendations based on tensor dimensionality reduction,” in *Proc. of the ACM Conference on Recommender Systems*, RecSys ’08, pp. 43–50, 2008.
- [165] Z. Cheng, J. Caverlee, and K. Lee, “You are where you tweet: a content-based approach to geo-locating twitter users,” in *Proc. of the 19th ACM International Conference on Information and knowledge management*, CIKM ’10, (New York, NY, USA), pp. 759–768, ACM, 2010.
- [166] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in *Proc. of the 19th International Conference on World Wide Web*, WWW ’10, (New York, NY, USA), pp. 851–860, ACM, 2010.
- [167] S. Wu, J. M. Hofman, W. A. Mason, and D. J. Watts, “Who says what to whom on twitter,” in *Proc. of the 20th International Conference on World wide web*, WWW ’11, (New York, NY, USA), pp. 705–714, ACM, 2011.
- [168] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, “You are who you know: inferring user profiles in online social networks,” in *Proc. of the third ACM International Conference on Web Search and Data Mining*, WSDM ’10, (New York, NY, USA), pp. 251–260, ACM, 2010.
- [169] M. Gomez Rodriguez, J. Leskovec, and A. Krause, “Inferring networks of diffusion and influence,” in *Proc. of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’10, (New York, NY, USA), pp. 1019–1028, ACM, 2010.
- [170] D. M. Romero, B. Meeder, and J. Kleinberg, “Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter,” in *Proc. of the 20th International Conference on World Wide Web*, WWW ’11, (New York, NY, USA), pp. 695–704, ACM, 2011.