SCALING ACTIVITY DISCOVERY AND RECOGNITION TO LARGE, COMPLEX DATASETS

by

PARISA RASHIDI

A dissertation submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

WASHINGTON STATE UNIVERSITY
Department of Electrical Engineering and Computer Science

MAY 2011

To the Faculty of Washington State University:

The members of the Committee appointed to examine the dissertation of PARISA RASHIDI find it satisfactory and recommend that it be accepted.

Diane J. Cook, Ph.D., Chair

Lawrence B. Holder, Ph.D.

Behrooz Shirazi, Ph.D.

ACKNOWLEDGMENTS

A famous Japanese proverb says that "better than a thousand days of diligent study is one day with a great teacher". When I first came to Washington State University in 2006 as a new graduate student, I knew little about research. During my undergraduate years, all I was expected to do was to diligently read my textbooks and learn them by heart. But now I was expected not only to read, but also to deliberate and criticize. I was expected not only to understand the past ideas, but also to come up with new ones. But, the question was: how do I do that? Where do I start? Obviously, even if I could spend thousands of days diligently reading and delving into every corner of computer science, it was not possible all by myself. In those days, I came to know Dr. Diane Cook. I first started working with her as a Masters student in 2006. She kindly showed me the ways of doing research, patiently going through my shaky first papers, and yet still encouraging me. To this date, she has constantly helped me in every aspect of my studies. Not only she is a great advisor, but she also has a great personality. I've learnt from her how to be respectful towards others and how to be understanding. Words cannot express how much I have learnt from you. I am grateful for having you as an advisor and for all of your endless

supports during my studies. I hope we can continue working together in the future. Thank you!

During the past 5 years, other great people also have been influential in my studies. Dr. Lawrence Holder has been a great source of advice and always a great teacher. He kindly provides advice whenever asked for. Dr. David Bakken has also been a great inspiration to me. He has showed me how passionate a teacher can be in teaching. He has provided me with help and guidance in countless occasions. Dr. Shirazi is another great professor who I wish to express my gratitude for his help and advice. Last, but not the least, other professors such as Dr. Zhe Dang, Dr. Chris Hundhausen, and Dr. Ananth Kalyanaraman have been very influential in my studies.

I also would like to thank all my dear friends for providing support and friendship in all these years. I have found friendship to be the pillar of a happy and fruitful life. Without you, it couldn't have been possible to smile at life. I also would like to thank the people of CASAS lab for their help during various stages of my studies.

I especially like to thank my mom Leyli, my dad Aziz and my brother Chia for their unconditional love and support. Without your endless support, it was impossible to finish my studies abroad. At the end, I want to say how thankful I'm for having you Nima as my soul-mate during the past 7 years. You kept faith in me in times when I lost faith in myself. Thank you for being there for me!

SCALING ACTIVITY DISCOVERY AND RECOGNITION TO LARGE, COMPLEX DATASETS

Abstract

by Parisa Rashidi, Ph.D.
Washington State University
May 2011

Chair: Diane J. Cook

In the past decade, activity discovery and recognition has been studied by many researchers. However there are still many challenges to be addressed before deploying such technologies in the real world. We try to address some of those challenges in order to achieve a more scalable solution that can be used in the real world.

First, we introduce a novel data mining method called the continuous Varied order Sequence Mining method (DVSM). It is able to discover activity pattern sequences, even if those patterns are disrupted or have varied step orders. We further extend DVSM into another data mining method called the Continuous varied Order Multi threshold activity discovery method (COM). COM is able to handle issues such as rare events across time and space. Furthermore, for discovering patterns in a real time manner, we extend COM as a stream mining method called StreamCOM.

In addition to discovering activity patterns, we propose several methods for transferring discovered patterns from one setting to another. We propose methods for transferring activity models of one resident to another, activity models of a physical space to another, and activity models of multiple spaces to another. We also show a method for selecting the most promising sources when multiple sources are available.

In order to further expedite the learning process, we also propose two novel active learning methods to construct generic active learning queries. Our generic queries are shorter and more intuitive and encompass many similar cases. We show how we can achieve a higher accuracy rate with fewer queries compared to traditional active learning methods.

All of our methods have been tested on real data collected from CASAS smart apartments. In several cases, we also tested our algorithms on various other datasets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Dedication

To my mother Leyli, who never thought of failure as an option, and taught me the same.

CHAPTER 1. INTRODUCTION

A man's errors are his portals of discovery.

— *James Joyce*

With remarkable recent progress in computing power, networking, and sensor technology, we are steadily moving into the world of ubiquitous computing where technology recedes into the background of our lives. Using sensor technology combined with the power of data mining and machine learning techniques, many researchers are now working on smart environments that can respond to the needs of the residents in a context aware way [Cook and Das, 2004].

For example, researchers are recognizing that smart environments can be of great value for monitoring and tracking the daily activities of individuals with memory impairments, as the ability to consistently complete Activities of Daily Living (ADLs) is necessary to live independently at home [Reisberg et al., 2001]. The need for the development of such smart home technologies is underscored by the aging of the population, the cost of formal health care, and the importance that individuals place on remaining independent in their own homes [Rialle et al., 2008]. Today, approximately 10% of the world's population is over the age of 60, by 2050 this proportion will have more than doubled. Moreover, the greatest rate of increase is

amongst the "oldest old", people aged 85 and over [Pollack, 2005]. Technologies such as smart homes can effectively help such an aging generation to stay independently at home, while lowering costs for the families and the government.

Besides assisted living, smart homes can be quite useful for providing more comfort, security and automation for residents. Some of the efforts for realizing such smart environments have been demonstrated in actual physical testbeds such as CASAS [Rashidi and Cook, 2009b], MavHome project [Cook et al., 2003], Gator Tech Smart House [Helal et al., 2005], iDorm [Doctor et al., 2005], and Aware Home [Abowd and Mynatt, 2004]. A couple of such smart home testbeds are specified in Table 1.1.

A smart environment typically contains many highly interactive devices, as well as different types of sensors. The data collected from various sensors can be used by data mining and machine learning techniques to discover residents' frequent activity patterns and to recognize such patterns later [Heierman and Cook, 2003, Liao et al., 2005]. Recognizing residents' activities allows the smart environment to respond in a context-aware way to the users' needs [Gopalratnam and Cook, 2007, Rashidi and Cook, 2008, Yiping et al., 2006, Wren and Munguia-Tapia, 2006].

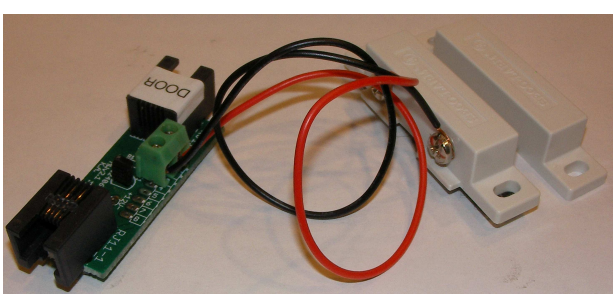
Our studies have been carried out as part of the CASAS Project [Rashidi and Cook, 2009b]. We carry out our studies using various ambient sensors. Because our study participants are uniformly reluctant to allow video data or to wear sensors, our

Project	By	Environment
CASAS	Washington State University	Home/Lab
Aware Home	Georgia Tech	Home
iSpace/iDorm	University of Essex	Home
MARC	University of Virginia	Home
Gator Tech	University of Florida	Home
PlaceLab	MIT	Home
Tiger Place	University of Missouri	Lab

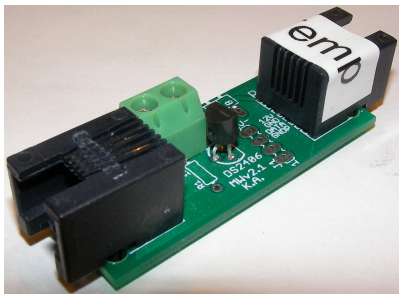
Table 1.1: Available smart home testbeds.

data collection consists of passive sensors such as motion sensors. Figure 1.1 shows a number of sensors that have been used in our studies[1]. In the following sections, we will discuss several challenges faced in activity discovery and recognition in smart homes, and we will discuss our proposed methods to address some of these challenges.

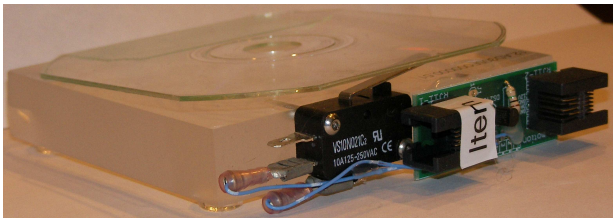
[1]Courtesy of Aaron Crandall



(a) Door sensor.



(b) Temperature sensor.



(c) Item sensor.

(d) Motion sensor.

(e) Water sensor.

Figure 1.1: Various sensors used in our experiments.

1.1 Activity Discovery and Recognition Challenges

While smart environments offer many societal benefits, they also introduce new and complex machine learning challenges. A typical home may be equipped with hundreds or thousands of sensors. Because the captured data is voluminous and rich in structure, the learning problem is a challenging one.

There is a variety of machine learning methods for discovering, modeling and recognizing activities. Though current supervised activity recognition methods can

recognize activities in some simplified settings, they still face many challenges. A few such challenges include how to avoid the time consuming step of data annotation, how to detect similar activities despite differences in the order of steps or experiencing disruptions, and how to discover activities in real time settings. The goal of this dissertation is to address these challenges by designing new machine learning and data mining methods. In an effort to address some of the those challenges, we propose several new activity discovery and recognition methods, as summarized in Table 1.2.

Method	**Addressed Challenges**
Sequential Data Mining	Data annotation, disrupted patterns
Stream Data Mining	Real time settings, disrupted patterns
Activity transfer learning	Scaling to more than a single environment
Active activity learning	Small labeled datasets

Table 1.2: Our proposed methods and the main challenges addressed by those methods.

Figure 1.2 shows the relation between our proposed methods in more detail. In the following sections, we provide a more detailed description of each one of the proposed solutions.

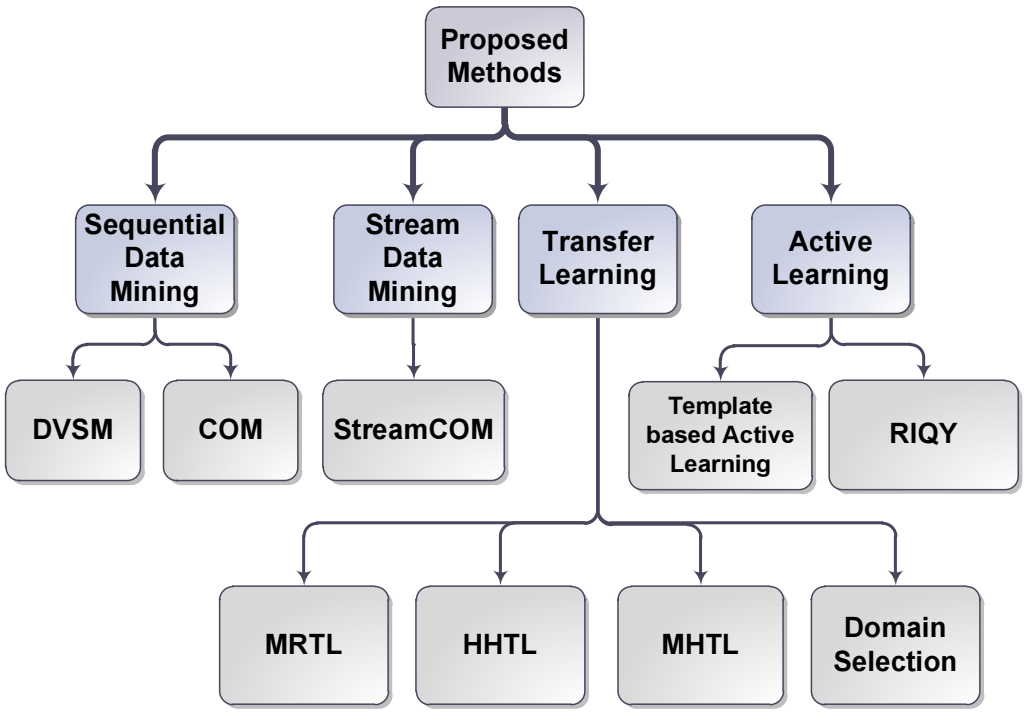


Figure 1.2: Contributions details.

1.2 Sequential Data Mining

Despite the fact that the majority of the proposed activity recognition methods in the literature are supervised [VG et al., 2008], using a supervised approach in the real world can be problematic. Supervised methods assume that we are provided with labeled training examples from a set of predefined activities (see Figure 1.3). However, the assumption of consistent pre-defined activities might not hold in reality. Due to physical, mental, cultural, and lifestyle differences not all individual perform the same

set of tasks [Wray and Laird, 2003]. Even for a specific pre-defined activity, different individuals might perform it in vastly different ways, making it impractical to rely on a list of pre-defined activities. As a result, data needs to be annotated for each individual and each task. On the other hand, annotating and hand labeling data is a very time consuming and laborious task. Despite tremendous research efforts, still very few labeled data sets are available for the community to use. This is another indication of the need for unsupervised approaches.

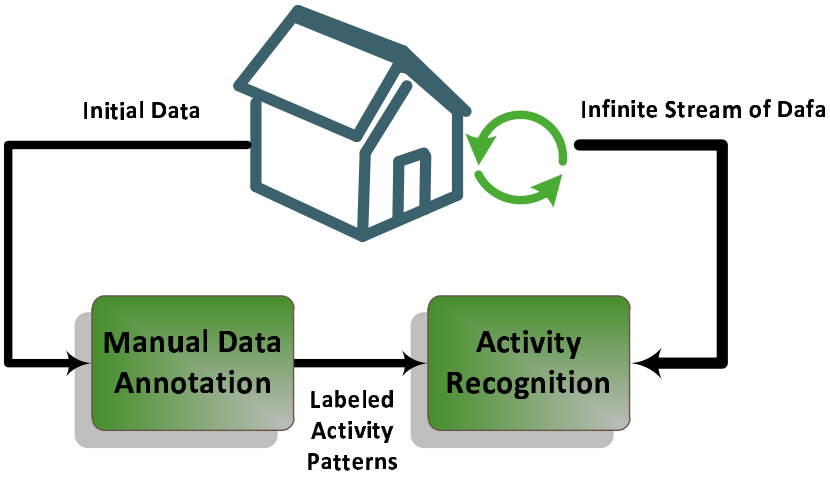


Figure 1.3: Supervised activity recognition methods require the training dataset to be "manually" labeled. Based on the labeled training dataset, a number of patterns are extracted and learned. The learned patterns are later used to recognize activities.

On the other hand, unsupervised methods require no training examples and

no labeled data, rather they automatically look for interesting patterns in the data (see Figure 1.4). Therefore unsupervised approaches seem to be more suitable for activity recognition in a normal day-to-day setting. By using unsupervised learning algorithms to discover interesting sensor event patterns, smart environment users can better understand activities that occur in the environment. These interesting patterns can be tracked over time and also used to perform trend detection and anomaly detection in the same environment. As a result, insights can be gained and behavioral baselines can be established without even relying upon labeled information and supervised learning algorithms.

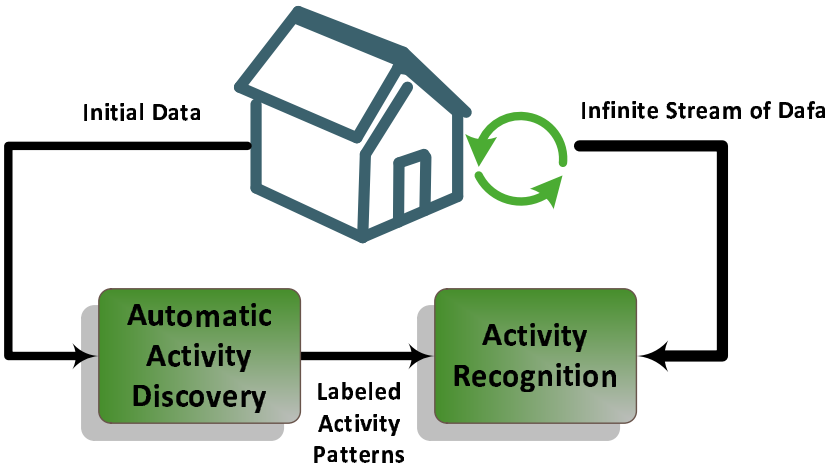


Figure 1.4: Unsupervised activity recognition methods require no labeled training dataset. They rather automatically extract activity patterns from the unlabeled training dataset, and use those discovered patterns to recognize activities.

There have been a number of unsupervised activity discovery and recognition methods introduced by researchers [Gu et al., 2009, Pei et al., 2007, Wyatt et al., 2005, Huynh and Schiele, 2006]. Most of these methods do not discover discontinuous patterns or patterns whose events may appear in varying order from occurrence to occurrence. We note that human activities are by nature erratic and order varying, thus a realistic approach to activity discovery and recognition needs to find discontinuous patterns as well as pattern variations.

We introduce two novel unsupervised activity discovery algorithms based on sequence mining methods to address the above issues. Our first method, called **D**iscontinuous **V**aried-order **S**equential **M**ining method (DVSM) is able to find frequent sequential patterns that may be discontinuous and might have variability in the ordering.

As a next step, we introduced COM, which stands for a **C**ontinuous, varied **O**rder, **M**ulti Threshold activity discovery method. COM not only discovers discontinuous patterns and their variations, but is also able to better handle real life data by dealing with the rare event problem. The rare event problem is the problem of varying frequencies for activities performed in different regions of the space. COM is able to find a higher percentage of frequent patterns, and thus achieve a higher accuracy. By pruning irrelevant patterns based on mutual information, COM retains only relevant variations of patterns, reducing the number of irrelevant variations.

1.3 Stream Activity Mining

Though using our unsupervised methods is a step forward for practical activity discovery and recognition in the real world, there are still some unanswered challenges. Our DVSM and COM algorithms do not take into account the streaming nature of data, nor the possibility that the patterns might change over time. In a real world situation, we have to deal with a potentially infinite and unbounded flow of data (see Figure 1.5). The discovered activity patterns can also change over time, and the algorithms need to detect and respond to such changes.

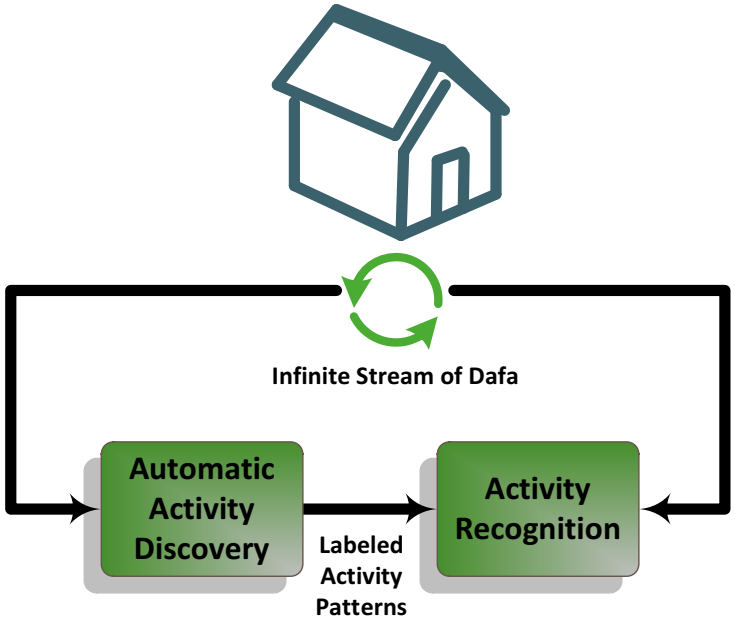


Figure 1.5: Stream mining methods for activity recognition can automatically detect patterns in data "over time", rather than as a one time initial step.

In order to address this issue, we adopt the tilted-time window approach [Giannella et al., 2003] to discover activity patterns over time. The tilted-time window approach finds frequent itemsets using a set of tilted-time windows, such that the frequency of the item is kept at a finer level for recent time frames, and at a coarser level for older time frames. We extended our COM method into a streaming version based on using the tilted-time window. Our proposed method, called StreamCOM, allows us to find discontinuous varied-order patterns in "streaming" sensor data "over time". StreamCOM represents the first reported stream mining method for discovering human activity patterns in sensor data over time.

1.4 Activity Transfer Learning

Previously we mentioned how to address some of the activity recognition challenges using unsupervised methods. Unfortunately a problem with all unsupervised approaches is that these methods usually require a large amount of unlabeled data in order to be able to find interesting patterns[2]. Collecting a large amount of data results in a lengthy data collection phase and a prolonged deployment process.

As a remedy, we can leverage the recognition and discovery process by using

[2]The same applies to semi-supervised methods which assume that at least we have access to a large pool of unlabeled data, besides a few labeled data points.

knowledge of discovered activities in previous spaces. Traditionally, each environmental situation has been treated as a separate context in which to perform learning. This means that providing labeled data and training the learning algorithm must occur anew with each new environment, and the new environment will not benefit from learning performed in other similar environments.

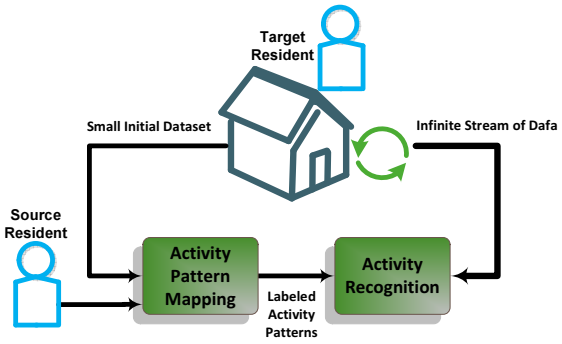
Here we propose several methods for transferring the knowledge of learned activities from a source space to a new target space.

Our first method transfers activity models from one resident to another resident in the same physical space (see Figure 1.6a). We call this method **M**ulti **R**esident **T**ransfer **L**earning (MRTL).

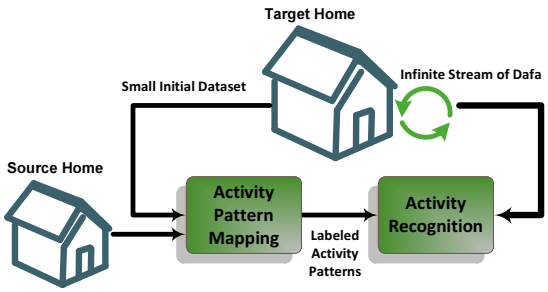
The next method transfers activities from one source physical space to another target physical space where the residents, the space layouts, and the sensors can be different (see Figure 1.6b). We call this method the **H**ome to **H**ome **T**ransfer **L**earning method (HHTL).

The third method transfers activities from multiple source physical spaces to a new target physical space (see Figure 1.6c). We call this method **M**ulti **H**ome **T**ransfer **L**earning (MHTL).

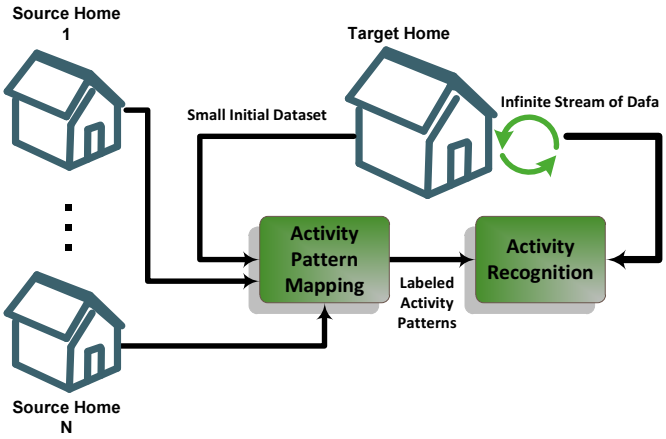
Finally the last method selects the best subset of source domains from multiple available domains (see Figure 1.6d). It provides a method for measuring the difference between activities performed in two different spaces. It should be noted that except



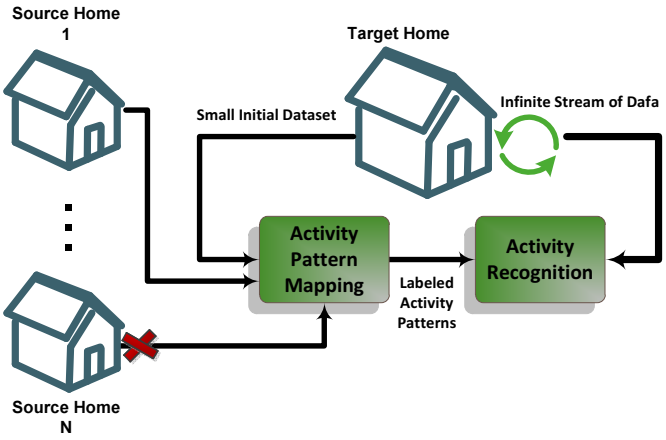
(a) Multi resident transfer learning method.



(b) Home to home transfer learning method.



(c) Multi home transfer learning method.



(d) Domain selection.

Figure 1.6: Activity transfer learning methods.

for the MRTL method, we assume that the space layouts, the residents, and the sensors can be different between source and target.

1.5 Active Activity Learning

There are many occasions where we have access to a small labeled dataset, as well as a pool of unlabeled instances. In such cases, we can take advantage of combining supervised methods or activity transfer methods with active learning [Lewis and Gale, 1994]. Active learning allows us to tailor the activity recognition process to the target space by querying the user for a few activity labels that we are most uncertain about (see Figure 1.7).

This approach of asking for only a few labeled instances is in contrast to supervised methods which require the whole dataset to be annotated beforehand, without imposing any criteria for choosing the data instances.

We propose two novel active learning methods. In contrast to traditional active learning methods, our methods do not query for the label of a specific instance. Rather we build generic and more intuitive queries by aggregating many similar cases into a single short query. Our first method uses a heuristic feature selection method. We call our second method RIQY, standing for a **R**ule **I**nduced active learning **Q**uer**Y** method, as it employs rule induction for building more generic queries. Both of

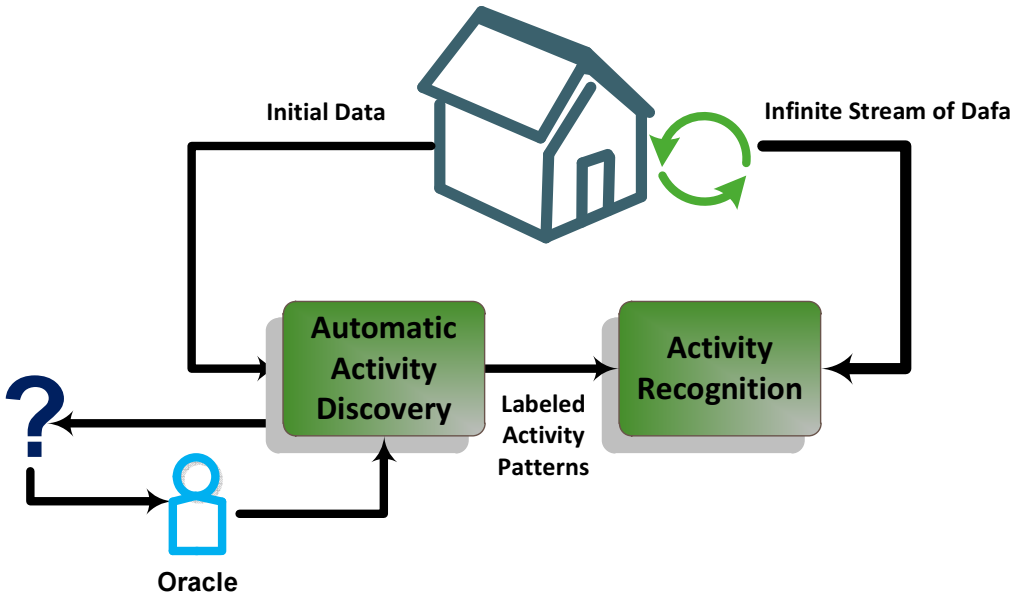


Figure 1.7: Active learning method allows the system to ask for the labels of activities about which it is most uncertain.

our methods are able to achieve higher accuracy rates with fewer queries. We also compare our active learning methods to a traditional uncertainty sampling approach [Lewis and Gale, 1994].

CHAPTER 2. RELATED WORK

Be as a tower firmly set; Shakes not its top for any blast that blows.

— Dante Alighieri

In recent years, many approaches have been proposed for activity discovery and recognition in different settings. In the following subsections, first we will provide a general overview of activity recognition methods. Next, we will review supervised activity recognition methods in more detail. Then unsupervised methods and in particular stream mining methods are explained. This discussion is followed by a literature overview of transfer learning methods and active learning methods.

2.1 Activity Recognition

Activity recognition is the problem of recognizing human activities from low level sensor data. The activity recognition algorithm is usually given a sequence of sensor readings. The algorithm also might be provided with annotated sensor events, where each sensor event is labeled with its corresponding activity label. The sensor readings (the sensor events) are collected from various types of sensors, such as ambient sensors, wearable sensors and more recently from personal devices and

Timestamp	Sensor ID	Label
7/17/2009 09:52:25	$M004$	Personal Hygiene
7/17/2009 09:56:55	$M030$	Personal Hygiene
7/17/2009 14:12:20	$M015$	None

Table 2.1: Example sensor data. Here $M004$, $M030$ and $M015$ denote sensor IDs.

cell phones. The learning algorithm usually preprocesses data to convert it to a more high level format. Table 2.1 shows several example sensor readings from our testbeds.

As depicted in Table 2.1, each sensor event can be part of a labeled activity, such as the first and the second sensor events. It also can have no activity labels, such as the third sensor event. The goal of an activity recognition algorithm is to predict the label of a sensor event or a sequence of sensor events, e.g. "Personal Hygiene" for $M004 - M030$ sequence.

In the past decade, researchers have proposed many different approaches for activity discovery and recognition. Activity discovery and recognition has also been explored in various settings. Methods have been proposed for recognizing nurses' activities in hospitals [Sánchez et al., 2008], recognizing quality inspection activities during car production [Ogris et al., 2008], monitoring elderly adults' activities [Cook and Schmitter-Edgecombe, 2009], studying athletic performance [Ahmadi et al., 2006,

Michahelles and Schiele, 2005], gait analysis in diseases such as Parkinson [Salarian et al., 2004, Pentland, 2004, Lorincz et al., 2009], emergency response [Lorincz et al., 2004] and monitoring unattended patients [Curtis et al., 2008].

Activity discovery and recognition approaches not only differ according to the deployed environment, but also with respect to the type of activity data that is collected, the model that is used for learning the activity patterns, and the method that is used for annotating the sample data. In the following subsections, we will explain each aspect in more details.

2.1.1 Activity Data

Activity data in smart environments can be captured via different mediums depending on the target environment and also target activities. Activity data at home can be collected using ambient sensors such as infrared motion sensors to track the motion of residents around home [Rashidi and Cook, 2009b]. Additional ambient sensors such as temperature sensors, pressure sensors, contact switch sensors, water sensors and smart powermeters can in addition provide other type of context information. For recognizing residents' interaction with the key objects, some researchers have used object sensors such as RFID tags that can be placed on key items [Tapia et al., 2004b, Philipose et al., 2004]. Another type of sensor that is used to recognize

activities, is wearable sensors such as accelerometers [Maurer et al., 2006, Yin et al., 2008]. Also more recent research uses mobile phones as carryable sensors [Zheng et al., 2010]. The advantage of wearable and mobile sensors is their ability to capture activity data in both indoor and outdoor settings. Their main disadvantage is their obtrusiveness, because user is required to carry the sensor all the time. Besides ambient sensors and wearable sensors, surveillance cameras and other types of image capturing devices such as thermographic camera have been used for activity recognition [Stauffer and Grimson, 2000]. However users are usually reluctant to using cameras for capturing activity data due to their privacy concerns [B.K. Hensel and Courtney, 2006]. Another limitation for using cameras is that activity recognition methods based on video processing techniques can be computationally expensive.

2.1.2 Activity Models

The number of machine learning models that have been used for activity recognition varies almost as greatly as the types of sensor data that have been used. Naive Bayes classifiers have been used with promising results for activity recognition [Brdiczka et al., 2005, Tapia et al., 2004b, van Kasteren and Krose, 2007]. Other researchers have employed other methods such as decision trees, Markov models, dynamic Bayes networks, and conditional random fields [Cook and Schmitter-

Edgecombe, 2009, Liao et al., 2005, Philipose et al., 2004, Sánchez et al., 2008, Maurer et al., 2006]. There also have been a number of works in the activity discovery area using unsupervised methods [Gu et al., 2009, Wyatt et al., 2005, Huynh and Schiele, 2006]. We will provide a more in-depth overview of these methods in Section 2.2 and Section 2.3.

2.1.3 Activity Annotation

Another aspect of activity recognition is the method that is used to annotate the sample training data. Most of the researchers have published results of experiments in which the participants are required to manually note each activity they perform at the time they perform it [Liao et al., 2005, Tapia et al., 2004b, Philipose et al., 2004]. In other cases, the experimenters told the participants which specific activities should be performed, so the correct activity labels were identified before the sensor data was even collected [Cook and Schmitter-Edgecombe, 2009, Gu et al., 2009, Maurer et al., 2006]. In one case, the experimenter manually inspected the raw sensor data in order to annotate it with a corresponding activity label [Wren and Munguia-Tapia, 2006]. Researchers have also used methods such as experience sampling [Tapia et al., 2004a], where subjects carry a personal digital assistant (PDA) as self-report devices.

2.2 Supervised Methods

There have been a number of supervised methods for activity recognition. Supervised methods assume that we have access to labeled data. In other words, the learning algorithm is presented with a number of training examples, each labeled with its corresponding activity label. In the case of activity recognition, the supervised learning algorithm learns a mapping from a sequence of sensor events to a corresponding activity label, using pre-labeled sequences as training data.

The simplest type of classifiers used for activity recognition is the Naive Bayes classifier. Naive Bayes classifiers have been used with promising results for activity recognition [Brdiczka et al., 2005, Tapia et al., 2004b, van Kasteren and Krose, 2007]. Naive Bayes classifiers identify the activity that corresponds with the greatest probability to the set of sensor values that were observed. Despite the fact that these classifiers assume conditional independence of the features, the classifiers yield good accuracy when large amounts of sample data are provided.

Other researchers, including Maurer et al. [Maurer et al., 2006], have employed decision trees to learn logical descriptions of the activities. This approach offers the advantage of generating rules that are understandable by the user, but it is often brittle when high precision numeric data is collected. Also a number of researchers have used other methods such as support vector machines [Brdiczka et al., 2009] and

back propagation neural networks [Favela et al., 2007].

The most popular type of activity recognition method is based on using Markov chains and hidden Markov models [Rabiner, 1990]. A Markov Model is a statistical model of a dynamic system. It models the system using a finite set of states, each of which is associated with a multidimensional probability distribution over a set of parameters. The system is assumed to have a Markovian property, i.e. the current state depends on only a finite history of previous states. Earlier approaches [Cook and Schmitter-Edgecombe, 2009] use Markov chains to recognize activities from sensor event traces that were segmented into non-overlapping sequences. A separate Markov model is learned for each activity and the model that best supports a new sequence of events is selected as the activity label for the sequence.

Similarly, a hidden Markov model (HMM) is a statistical model that is assumed to be Markovian. However, hidden Markov models take the assumption that the underlying data is generated by a stochastic unobservable process. HMMs traditionally perform well in cases where temporal patterns need to be recognized, such as human activity recognition [Hu et al., 2009, Cook and Schmitter-Edgecombe, 2009, Sánchez et al., 2008]. Researchers also have used variations of hidden Markov models for human activity recognition, such as the hidden semi-Markov model [Duong et al., 2005], relational Markov networks [Liao et al., 2005], and the hierarchical hidden Markov model [Kawanaka et al., 2006].

Hidden Markov models are a special case of more general dynamic Bayesian networks (DBNs) [Ghahramani, 1998]. A DBNs explicitly represents conditional independencies in the variables, therefore it allows for a more efficient and accurate inference and learning procedure. Brand et al. [Brand et al., 1997] have introduced a simple DBN extension of HMMs called "the coupled hidden Markov model", which is used for recognition of simultaneous human actions. They also proposed a multi-layer version to learn the model of office activity [Oliver et al., 2002]. Quantitative temporal Bayesian networks (QTBNs) as another type of DBN have been used by Colbry et al. [Colbry, 2002] for monitoring the steps of a plan from sensor observations.

Another more flexible alterative to HMM is conditional random fields [Lafferty et al., 2001]. As discriminative models, CRF try to find the conditional probabilities instead of the joint probabilities. CRFs also have been used by a number of researchers for activity recognition [Vail et al., 2007, Sminchisescu et al., 2005, Sukthankar and Sycara, 2006].

2.3 Unsupervised Methods

Unlike supervised methods, unsupervised methods do not require any labeled data. Instead, they try to automatically find interesting patterns in unlabeled data. Activity discovery methods also as unsupervised methods try to find interesting ac-

tivity patterns in sensor data.

A number of activity discovery methods have been proposed by different researchers. Gu et al. [Gu et al., 2009] have used the notion of emerging patterns to look for frequent sensor sequences that can be associated with each activity as an aid for recognition. Heierman et al. [Heierman and Cook, 2003] propose a method for episode discovery from activity data. Mahajan et al. [Mahajan et al., 2004] use a finite state machine network that learns in an unsupervised mode the usual patterns of activities in a scene over long periods of time. In the recognition phase, the usual activities are accepted as normal and deviant activity patterns are flagged as abnormal. Schiele et al. [Schiele, 2006] propose a method for detecting activity structure using low dimensional Eigenspaces. Vahdatpour et al. [Vahdatpour et al., 2009] address the problem of activity and event discovery in multi dimensional time series data by proposing a method for locating multi dimensional motifs in time series. Similarly Zhao [Zhao et al., 2010] propose a framework for discovering activity motif features for CRF-based classification. Huynh et al. [Huynh and Schiele, 2006] use a combination of discriminative and generative methods to achieve less supervision for activity recognition. Barger et al. [Barger et al., 2005] use mixture models to develop a probabilistic model of behavioral patterns. Dimitrov et al. [Dimitrov et al., 2010] utilize background domain knowledge about the user activities and the environment in combination with probabilistic reasoning methods in order to build possible expla-

nation of the observed stream of sensor events. Previously, we also have proposed a method for simultaneous discovery of frequent and periodic patterns from activity data [Rashidi and Cook, 2009b].

Also a number of methods have been proposed based on mining activities' definitions from the web. Perkowitz et al. [Perkowitz et al., 2004] mine definitions of activities in an unsupervised manner from the web. Similarly Wyatt et al. [Wyatt et al., 2005] view activity data as a stream of natural language terms. Activity models are then considered as mappings from such terms to activity names, and are extracted from text corpora such as the web. Palmes et al. [Palmes et al., 2010] also mine the web to extract the most relevant objects according to their normalized usage frequency. They develop an algorithm for activity recognition and two algorithms for activity segmentation with linear time complexities.

2.3.1 Sequence Mining

A sequence s is a set of ordered items denoted by $\langle s_1, s_2, \cdots, s_n \rangle$. In our activity recognition problems, a sequence is ordered by its timestamps. The task of discovering all the frequent sequences can be quite challenging due to its exponential search space.

Over the past decade, a number of sequence mining methods have been proposed. The first sequence mining algorithm called GSP was introduced by Agrawal

and Srikant [Agrawal and Srikant, 1995]. The algorithm was based on the Apriori approach and the generation-pruning principle [Agrawal et al., 1993]. GSP makes several passes over the database to count the support of each itemset and to generate candidates. Next, it prunes the itemsets with a support count below the minimum support. Another similar approach is PSP which uses a prefix-based tree and again is based on the generation-pruning approach [Masseglia et al., 1998].

SPADE is another algorithm which only needs three passes over the database in order to discover sequential patterns [Zaki, 2001]. Ayres et al. [Ayres et al., 2002] proposed an algorithm called SPAM which unlike previous approaches uses a vertical bitmap representation of database. Wang et al. [Wang and Han, 2004] propose BIDE, an efficient algorithm for mining frequent closed sequences without candidate maintenance. Pei et al. [Pei et al., 2007] propose a constraint-based sequential pattern mining method. Finally, FREESPAN [Han et al., 2000] and PREFIXSPAN [Pei et al., 2001] are among the first algorithms to consider a projection method for mining sequential patterns.

2.3.2 Stream Mining

As mentioned in Section 2.3.1, sequential pattern mining has been studied for more than a decade [Agrawal and Srikant, 1995] and many methods have been pro-

posed for finding sequential patterns in data [Agrawal and Srikant, 1995, Pei et al., 2001, Wang and Han, 2004, Masseglia et al., 1998]. Compared to the classic problem of mining sequential patterns from a static database, mining sequential patterns over data streams is far more challenging. In a data stream, new elements are generated continuously and no blocking operation can be performed on the data. Despite being more challenging, with the rapid emergence of new application domains over the past few years the stream mining problem has also been studied in a wide range of different application domains. A few such application domains include network traffic analysis, fraud detection, Web click stream mining, power consumption measurements and trend learning [Garofalakis et al., 2002].

For finding patterns in a data stream, approximation and using a relaxed support threshold is a key concept [Chang and Lee, 2003, Manku and Motwani, 2002]. The first approach was introduced by Manku et al. [Manku and Motwani, 2002] based on using a landmark model and calculating the count of the patterns from the start of the stream. Later Li et al. [fu Li et al., 2004] proposed methods for incorporating the discovered patterns into a prefix tree. They also designed methods for regression-based stream mining algorithms [Teng et al., 2003]. More recent approaches have introduced methods for managing the history of items over time [Giannella et al., 2003, Teng et al., 2003]. The main idea is that one usually is more interested in recent changes in more detail, while older changes are preferred in coarser granularity

in long term.

There also have been several methods for finding sequential patterns over data streams, including the SPEED algorithm [Raïssi et al., 2005], methods for finding approximate sequential patterns in Web usage [Marascu and Masseglia, 2006], a data cubing algorithm [Han et al., 2005] and mining multidimensional sequential patterns over data streams [Raïssi and Plantevit, 2008]. All of these approaches consider data to be in a transactional format. However, input data stream in a smart environment is a continuous flow of unbounded data.The sensor data has no boundaries separating different activities or episodes from each, and it is just a continuous stream of sensor events over time. To deal with this problem, we extend the DVSM method to group together the co-occurring events into varied-order discontinuous activity patterns.

2.4 Transfer Learning

The process of exploiting the knowledge gained in one problem and applying the learned knowledge to a different but related problem is called transfer learning [Caruana, 1997, Raina et al., 2006]. It is a hallmark of human intelligence, and has been vastly studied in the literature [Pan and Yang, 2010]. Researchers have studied transfer learning in different computational settings such as reinforcement learning [Asadi and Huber, 2007], genetic algorithms [Taylor et al., 2007], neural networks

[Thrun, 1996], Bayesian models [Roy and Kaelbling, 2007] and many other methods [Pan and Yang, 2010].

Though transfer learning has been vastly studied in the literature, it has been applied to activity discovery and recognition in very few cases. Zhang et al. [Zheng et al., 2009] have developed a model for mapping different types of activities to each other (e.g. sweeping to cleaning) by learning a similarity function via a Web search. Kasteren et. al [van Kasteren et al., 2008] describe a simple method for transferring the transition probabilities of Markov models for two different spaces. They only transfer the transition probabilities, and most other activity features such as the activity's structure and related temporal features is ignored, as they assume the structure of HMMs is pre-defined.

Transfer learning methods can be categorized into several subcategories depending on the availability of data in the source and target space. Of particular interest for us is the domain adaptation category which is discussed in the following subsection.

2.4.1 Domain Adaptation

Domain adaptation is a type of transfer learning which assumes that we have access to labeled data in a source space, but no labeled data is available in a target space. The objective of domain adaptation is to develop learning algorithms that

can be easily transferred from one domain to another. For example in the case of sentiment analysis, one might use reviews in the Amazon.com books category in order to classify users' reviews in its DVD category. In our scenario, domain adaptation is appealing because it allows for a practical activity recognition solution in smart environments.

Domain adaptation has been recently studied by many researchers, and various methods have been proposed. Most of the domain adaptation methods have been proposed in the context of natural language processing problems. A number of such methods include the structural correspondence learning (SCL) by Blitzer et al. [Blitzer et al., 2006], feature replication (FR) by Daumé III et al. [Daumé, 2007], and adaptive support vector machines (A-SVM) by Yang et al. [Yang et al., 2007].

Domain adaptation has also been generalized to multiple source domains, and several multiple source domain adaptation methods have been proposed. Crammer et al. [Crammer et al., 2008] proposed a method by assuming that the distributions of multiple sources are the same and label change is only due to the noise. Luo et al. [Luo et al., 2008] proposed to maximize the prediction consensus from multiple sources. Mansour et al. [Mansour et al., 2009], propose a method of multiple source domain adaptation by assuming that the target function is a mixture model. Duan et al. [Duan et al., 2009] showed a method based on smoothness assumption using support vector machines, called Domain Adaptation Machine (DAM). They assume

that in addition to the unlabeled data, one also has access to limited labeled data from the target domain.

In our proposed methods, we assume that we only have access to limited "unlabeled" target data, to achieve a fast practical solution in real world. This is in contrast to the above proposed methods which assume that we have access to ample amounts of unlabeled data from target domain. We also assume that the source distributions are different. One can imagine that activities performed in different environments by different residents will not have the same distribution.

2.5 Active Learning

A variety of active learning approaches have been proposed during the past decade [Settles, 2009, Tomanek and Olsson, 2009]. The simplest form of active learning is the uncertainty sampling method which was introduced by Lewis and Gale [Lewis and Gale, 1994]. Uncertainty sampling first labels all the unlabeled instances using a classifier. Then it chooses the most uncertain instance and asks the oracle for its label. Other researchers have extended this method to multi-class classification problems by using uncertainty measures such as the entropy measure [Hwa, 2004]. It should be noted that the uncertainty sampling method can select the outliers, as it ignores the underlying distribution and the outliers as isolated points can be highly

uncertain.

Another type of active learning method is the committee based active learning method. It constructs a committee of classifiers and selects an unlabeled example that causes the maximum disagreement among the classifiers [Seung et al., 1992, Freund et al., 1997, McCallum and Nigam, 1998]. Similar to the committee based method is the co-testing method which trains two classifiers on two uncorrelated views of data [Muslea et al., 2000]. Version space reduction methods also discard areas of the hypothesis space that have no direct effect on the error rate [Tong and Koller, 2000].

Other query strategy approaches include maximum expected gradient length [Settles and Craven, 2008], maximum expected error reduction [Roy and McCallum, 2001], and maximum variance reduction [Zhang and Oles, 2000].

More recently, density weighted methods have been proposed [Xu et al., 2003, Nguyen and Smeulders, 2004, Xu et al., 2007, Settles and Craven, 2008]. Density weighted methods balance the uncertainty of a sample with its representativeness according to the underlying data distribution. These methods ask for the label of an uncertain instance which is also similar to the other unlabeled instances. As density weighted methods sample from the maximal density regions, they perform well with minimally labeled data. They are also better able to deal with the outlier problem [Donmez et al., 2007]. Settles and Craven [Settles and Craven, 2008] have shown that if the proximities are pre-computed and cached, then the computational complexity

of those methods is no different than the traditional active learning methods.

There are a number of active learning frameworks that resemble our generic query approach. Batch mode active learning [Brinker, 2003, Guo and Schuurmans, 2008] generates queries in groups instead of a single instance at a time. However batch mode active learning still requires the oracle to label "all" of the instances in a batch and does not reduce the set of similar queries to a generic query.

The multiple instance active learning method also groups several instances together [Dietterich et al., 1997, Settles and Craven, 2008]. The group is labeled negative if the oracle labels all the instances in the group as negative, but it is labeled positive if at least one of the instances is marked positive. Again the oracle has to label all the instances instead of a generic query.

Druck et al [Druck et al., 2008] proposed a feature labeling method where a single feature is queried for its label. For example, in a baseball vs. hockey text classification problem, the presence of the word "puck" is a strong indicator of hockey. As this method is selecting one feature at a time, it can be said that it is a specific case of our generic query method.

Du et al. [Du and Ling, 2010] also proposed a method for grouping several instances together. It does not take into account the actual data distribution as it groups synthetic data points together. It randomly generates a fixed number of synthetic data points around an informative instance selected by an uncertainty sam-

pling method. As pointed out by others [Lang, 1995], synthetic data points might not exactly reflect the actual data distribution, and might result in adding the outliers.

CHAPTER 3. SEQUENCE MINING

All truths are easy to understand once they are discovered; the point is to discover them.

— Galileo Galilei

In this chapter, we will explore the use of unsupervised methods for activity discovery. The advantage of unsupervised methods over the conventional supervised methods is that the unsupervised methods do not need any labeled data, instead they can automatically discover activity patterns in sensor data.

In the following subsections, first we will provide a brief introduction. Then we will explain two of our proposed sequential data mining methods called DVSM [Rashidi et al., 2010] and COM [Rashidi and Cook, 2010c]. We also present the results of our experiments based on DVSM and COM in the following sections.

3.1 Introduction

As previously mentioned in Section 1, the ability to track and recognize activities in a smart environment is one of the vital functionalities of the smart environment. For example, to function independently at home, individuals need to be able to com-

plete Activities of Daily Living (ADLs) such as eating, dressing, cooking, and taking medicine [Reisberg et al., 2001]. As pointed out before, the generally accepted approach for activity recognition is using a supervised machine learning method which needs labeled examples of typical activities.

As one might imagine, a number of difficulties arise with a supervised approach. First, there is an assumption that each individual performs most, or all, standard activities in a consistent pre-defined manner in their home environments. This is certainly not always the case. For example, while an individual may regularly eat meals, they may go out to restaurants for the majority of their meals, which would make tracking this activity challenging for a smart home. Even for a single common activity that is performed in a monitored environment, different individuals might perform it in vastly different ways, making the reliance on a list of pre-defined activities impractical due to the inter-subject variability. In addition, the same individual might perform even the same activity in different ways. This calls for methods which can also deal with intra-subject variability.

Second, tracking only pre-selected activities ignores the important insights that other activities can provide. For example, Hayes, et al. [Hayes et al., 2007] found that variation in the overall activity level at home was correlated with mild cognitive impairment. This activity level was not restricted to predetermined activities but was related to the total activity level in the monitored environment. This highlights

the fact that it is important to recognize and monitor all activities that an individual regularly performs in their daily environments.

Third, to track a predefined list of activities, a significant amount of training data must be labeled and be made available to the machine learning algorithm. Individuals perform activities differently due to physical, mental, cultural, and lifestyle differences [Wray and Laird, 2003]. Therefore, sample data needs to be collected and labeled for each individual. Unfortunately, collecting and labeling such sensor data collected in a smart environment is an extremely time-consuming task. If the individual is asked to participate by keeping track of their own activities over a period of time, the process is additionally obtrusive, laborious, and prone to self-report error [Szewcyzk et al., 2009].

We introduce two unsupervised methods for discovering and tracking activities in a smart environment to address the above issues. The first method is called the **D**iscontinuous **V**aried-order **S**equence **M**ining method (DVSM). It is able to find frequent patterns that may be discontinuous and might have variability in the ordering, thereby addressing the intra-subject and inter-subject variability issues. The second method is called COM, which stands for a **C**ontinuous, varied **O**rder, **M**ulti Threshold activity discovery method. COM not only discovers discontinuous patterns and their variations, but is also able to better handle real life data by dealing with different frequencies/sensors problem. COM is able to find a higher percentage of the

frequent patterns, and thus achieves a better discovery and recognition accuracy.

The unsupervised nature of our models provides a more automated approach for activity recognition than is offered by previous approaches, which take a supervised approach and annotate the available data for training. Compared to traditional methods for activity recognition which solely utilize HMMs or other models for recognizing labeled activities, our approach first "discovers" interesting patterns of activity, and then recognizes these discovered activities to provide a more automated approach.

3.2 DVSM

We propose DVSM as a unique mining method for discovering activity patterns which may be discontinuous and might have variability in the ordering. As a result, we are able to address the intra-subject variability issue. Because we discover activity patterns that are common for each individual instead of using pre-selected activities, we are also able to address the issue of inter-subject variability. We employ activity clustering to group the patterns into activity definitions, where the cluster centroids represent the activities that will be tracked and recognized. In the next step, we create a boosted version of a hidden Markov model to represent the activities and their variations, and to recognize those activities when they occur in the smart environment. By recognizing activities as they occur, the smart home can determine when

the activities occur, and perform analysis on their timing to determine long-term trends and assess activity variability. Our DVSM method requires no annotation and no input from the participant. As a result, it represents a fully-automated approach to performing activity tracking. The architecture of our method is shown in Fig. 3.1.

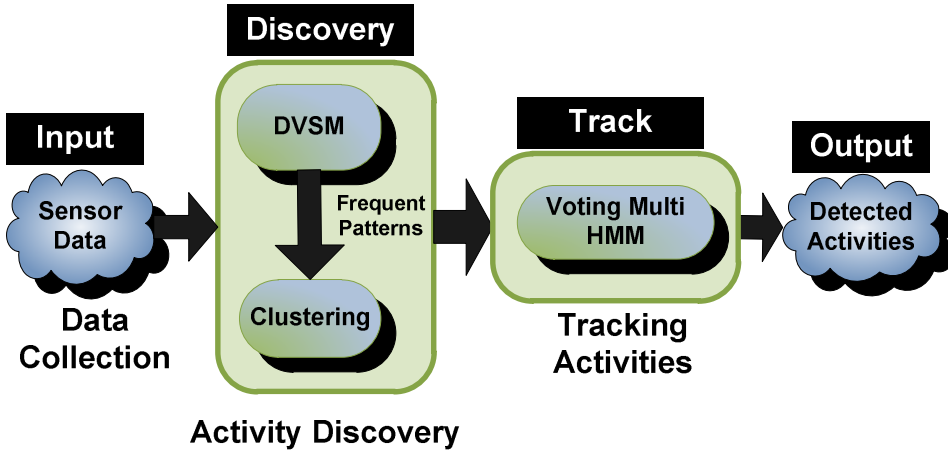


Figure 3.1: Main components of DVSM.

In the following subsections, we detail our approach. Section 3.2.1 explains the discovery of activities using DVSM and then clustering methods. Section 3.2.2 describes how discovered activities can be recognized using the HMM model. Finally in section 3.2.3 we will present the results of our experiments for scripted ADL activities, scripted interleaved ADL activities, and also long-term daily resident activities.

3.2.1 Discovering Activities

The first step of our algorithm is to identify the frequent and repeatable sequences of sensor events that comprise our smart environment's notion of an activity. Once we identify the activity and specific occurrences of the activity, we can build a model to recognize the activity and begin to analyze the occurrences of the activity. But how then do we discover these activities? By applying frequent sequential pattern mining techniques we can identify contiguous, consistent sensor event sequences that might indicate an activity of interest. As pointed out in Section 2.3.1, many methods have been proposed for mining sequential data. These methods include mining frequent sequences [Agrawal and Srikant, 1995], constraint-based mining [Pei et al., 2007], and episode discovery [Heierman and Cook, 2003]. One limitation of these approaches is that they do not discover discontinuous patterns, which can appear in daily activity data due to the erratic nature of human activities. For example, when an individual prepares a meal, the steps do not always follow the same strict sequence; rather, their order may be changed and be interleaved with steps that do not consistently appear each time.

Ruotsalainen et al. [Ruotsalainen and Ala-Kleemola, 2007] introduce their Gais algorithm for detecting interleaved patterns using genetic algorithms, but this is a supervised learning approach that looks for matches to specific pattern templates.

Other approaches have been proposed to mine discontinuous patterns but they have difficulty finding hybrid continuous-discontinuous patterns [Pei et al., 2001, Zaki et al., 1998] and have difficulty finding patterns whose order may vary from one occurrence to another [Chen et al., 2002].

Given that we want to discover sequential patterns that may be discontinuous and have variability in the ordering, another possible approach is to cluster the sensor events. Keogh et al. [Keogh et al., 2003] claim that the clusters that result from processing streaming time series data are essentially random. However, time series and sequence clustering algorithms have shown to be effective in constrained situations. For example, sequence mining algorithms have been successfully used in bioinformatics to discover related gene sequences [Malde et al., 2003]. The limitation of clustering algorithms for our problem is that we do not want to cluster all of the data points, but only those that are part of an activity sequence which is likely to occur frequently and with some degree of regularity or recognizability.

Because both sequence mining and clustering algorithms address a portion of our problem, we combine these two methods into an activity discovery method to identify frequent activities and cluster similar patterns together. Specifically, we apply our own frequent sequence mining algorithm DVSM combined with a clustering algorithm to identify sensor event sequences that likely belong together and appear with enough frequency and regularity to comprise an activity that can be tracked and analyzed.

Discovering frequent discontinuous sequences

Our activity discovery method performs frequent sequence mining using DVSM to discover frequent patterns, and then groups the similar discovered patterns into clusters. DVSM can automatically discover patterns of resident's activities, even if the patterns are somehow discontinuous or have different event orders across their instances. Both situations happen quite frequently while dealing with human activity data. For example, consider a "meal preparation activity". Most people will not perform this activity in exactly the same way each time, rather some of the steps or the order of the steps might be changed (variations). Even one person will perform this differently different times (e.g., heating up oatmeal versus making Thanksgiving dinner). In addition the activity might be interrupted by irrelevant events such as answering the phone (discontinuous). As a more concrete example, consider instances $\{b, x, c, a\}$, $\{a, b, q\}$, and $\{a, u, b\}$. DVSM can extract the pattern $\langle a, b\rangle$ from those instances despite the fact that the events are discontinuous and have varied orders. It should be noted that our algorithm is also able to find continuous patterns by considering them as patterns with no discontinuity.

Our approach is different from frequent itemset mining because we consider the order of items as they occur in the data. Unlike many other sequence mining algorithms, we report a general pattern that comprises all frequent variations of a single pattern that occur in the input dataset $\mathcal{D}$. For general pattern a we denote the

ith variation of the pattern as a_i, and we call the variation that occurs most often among all variations of a the *prevalent* variation, a_p. We also refer to each single component of a pattern as an event (such as a in the pattern $\langle a, b \rangle$).

To find these discontinuous order-varying sequences from input data $\mathcal{D}$, DVSM first creates a reduced dataset D_r containing the top α most frequent events. Next, DVSM slides a window of size 2 across D_r to find patterns of length 2. After this first iteration, the whole dataset does not need to be scanned again. Instead, DVSM extends the patterns discovered in the previous iteration by their prefix and suffix events, and will match the extended pattern against the already-discovered patterns (in the same iteration) to see if it is a variation of a previous pattern, or if it is a new pattern [Rashidi and Cook, 2009b]. To facilitate comparisons, we save general patterns along with their discovered variations in a hash table.

To see if two patterns should be considered as variations of the same pattern, we use the Levenshtein (edit) distance [Levenshtein, 1966] to define a similarity measure $sim(A, B)$ between the two patterns. The edit distance, $e(A, B)$, is the number of edits (insertions, deletions, and substitutions) required to transform an event sequence A into another event sequence B. We define the similarity measure based on the edit distance as in Equation 3.1.

$$sim(A, B) = 1 - \left(\frac{e(A, B)}{max(|A|, |B|)}\right) \tag{3.1}$$

At the end of each iteration, we prune infrequent variations of a general pattern, as well as infrequent general patterns. We identify general patterns as interesting if their compression value according to Equation 3.2 is above the compression threshold C. Similarly, a variation i of the pattern is considered as interesting if its compression value according to Equation 3.3 is above the variation compression threshold C_v. Here DL computes the description length of the argument, which can be computed as the number of bits required to encode data D using an optimal encoding. Also Γ refers to pattern continuity, as will be described shortly.

$$c(a) = \frac{DL(D) * \Gamma_a}{DL(a) + DL(D|a)} \tag{3.2}$$

$$c(a_i) = \frac{(DL(D|a) + DL(a)) * \Gamma_{a_i}}{DL(D|a_i) + DL(a_i)} \tag{3.3}$$

We transform the compression value c to be in range of [0..1] via the so-called softmax scaling technique [Pyle, 2005], as in Equation 3.4.

$$s(c) = \frac{1}{1 + \exp(c)} \tag{3.4}$$

Our approach to identifying interesting patterns aligns with the minimum description length principle [Rissanen, 1978] which advocates that the pattern which best describes a dataset is the one which maximally compresses the dataset by replacing instances of the pattern by pointers to the pattern definition. However, since

we allow discontinuities to occur, each instance of the pattern needs to be encoded not only with a pointer to the pattern definition, but also with a continuity factor, Γ. The discontinuity of a pattern instance is calculated as the number of bits required to express how the pattern varies from the general definition.

To understand what the continuity function measures, consider a general pattern $\langle a, b, c \rangle$ as shown in Fig. 3.2. An instance of the pattern is found in the sequence $\{a, b, g, e, q, y, d, c\}$ where symbols "g e q y d" separate the pattern subsequences $\{a, b\}$ and $\{c\}$. Though this sequence may be considered as an instance of the general pattern $\langle a, b, c \rangle$, we still need to take into account the number of events that appear between subsequences $\{a, b\}$ and $\{c\}$. In terms of calculating a pattern's compression, discontinuities increase the description length of the data because the way in which the pattern is broken up needs to be encoded.

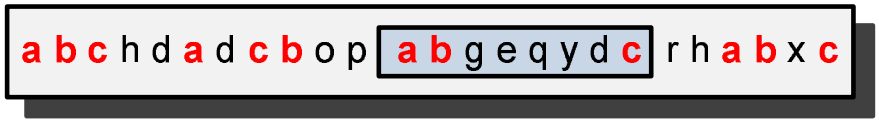


Figure 3.2: A small dataset containing pattern $\langle a, b, c \rangle$.

The continuity between component events, Γ_e, is defined for each two consecutive events in an instance. For each frequent event e', we record how far apart (or separated, denoted by $s_{e'}$) it is from a preceding frequent event in terms of the number of events that separate them in $\mathcal{D}$ (in the above example, $s_c = 5$). Then $\Gamma_e(e')$,

the event continuity for $\acute{e}$, is defined as in Equation 3.5.

$$\Gamma_e(\acute{e}) = \frac{1}{s_{e'} + 1} \tag{3.5}$$

The more the separation that exists between two frequent events, the less will be the event continuity. Based on event continuity, the instance continuity Γ_i reflects how continuous its component events are. For an instance j of a variation a_i, $\Gamma_i(a_i^j)$ will be defined as in Equation 3.6 where $|a_i^j|$ is the length of a_i^j. Here k ranges over the length of the instance.

$$\Gamma_i(a_i^j) = \frac{1}{|a_i^j|} \sum_{k=1}^{|a_i^j|} \Gamma_e(k) \tag{3.6}$$

The continuity of a variation, Γ_v, is then defined as the average continuity of its instances. $\Gamma_v(a_i)$ is defined as in Equation 3.7, where n_{a_i} shows the total number of instances for variation a_i.

$$\Gamma_v(a_i) = \frac{1}{n_{a_i}} \sum_{j=1}^{n_{a_i}} \Gamma_i(a_i^j) \tag{3.7}$$

The continuity, Γ_g, of a general pattern g, is defined as the weighted average continuity of its variations. Γ_g is defined according to Equation 3.8, where the continuity for each a_i is weighted by its frequency f_{a_i} and n_a shows the total number of variations for general pattern a.

$$\Gamma_g(a_i) = \frac{\sum_{i=1}^{n_a} \Gamma_v(a_i) * f_{a_i}}{\sum_{i=1}^{n_a} f_{a_i}} \tag{3.8}$$

Building on this definition of continuity, we can replace Γ in Equation 3.2 and Equation 3.3. Patterns that satisfy the compression threshold are flagged as interesting, as are variations that satisfy the variation compression threshold. The rest of the patterns and variations are pruned. Every iteration, we also prune redundant non-maximal patterns; i.e., those patterns that are totally contained in another larger pattern. This considerably reduces the number of discovered patterns. We continue extending the patterns by prefix and suffix until no more interesting patterns are found. A post-processing step records attributes of the patterns, such as event durations.

Clustering sequences into Groups of Activities

The second step of our algorithm is to identify pattern clusters that will represent the set of discovered activities. Though the mining stage groups together similar variations of a pattern, this is based solely on structural similarity and common sensor events. Therefore, similar patterns activating a different set of sensors will be considered as separate patterns, even if those patterns exhibit a high degree of similarity in start times, duration and occurrence locations. To remedy this problem, we use a clustering algorithm. The clustering algorithm groups patterns as a result of their structure, start time, duration and regional similarity. In the clustering step, not only

do we consider structural similarity as a measure of similarity, but we also take into account the start time similarity, duration similarity and regional similarity. Using clustering also addresses the problem of frequent pattern discovery where too many similar patterns are generated, which makes it difficult to analyze the true underlying major salient ideas.

Specifically, our algorithm groups the set of discovered patterns $\mathcal{P}$ into a set of clusters $\mathcal{A}$ such that each group of patterns representing a specific activity is assigned to one cluster. The resulting set of cluster centroids represent the activities that we will model, recognize, and track. Though our algorithm uses a standard k-means clustering method [Hartigan and Wong, 1979], we still need to define a method for determining cluster centroids and for comparing activities in order to form clusters. A number of methods have been reported in the literature for sequence clustering, such as the CLUSEQ algorithm by Yang et al. [Yang and Wang, 2002] and the ROCK algorithm by Noh et al. [Noh et al., 2006]. The difference between their approach and ours is that they consider purely symbolic sequences with no features attached to them. In contrast, sensor event sequences are not simply strings, but each entry in the sequence also has associated features such as temporal information that need to be considered during the discovery process.

Two methods that are commonly used for comparing the similarity of sequences are edit distance [Levenshtein, 1966] and longest common subsequence (LCS) [Se-

queira and Zaki, 2002] for simple sequences. In addition, Saneifar et al. [Saneifar et al., 2008] proposed a similarity measure for more complex itemset sequences based on the number of common items. These methods are not sufficient to address our clustering problem. This is because while the methods do satisfy our definition of an activity as a sequence of events, they do not process the temporal information that is encoded in our sensor event data nor do they handle the special cases that occur in our application such as reasoning about ordering information. As a result, we refine these existing methods to apply them to our smart environment sensor event sequences.

The patterns discovered by DVSM are composed of sensor events. In the clustering algorithm the pattern is composed of states. States correspond to the pattern's events, but are enhanced to include additional information such as the type and duration of the sensor events. In addition, we can combine several states together to form a new state (we call it an extended state). We combine all consecutive states corresponding to the sensors of the same type to form an extended state. For example, if a motion sensor in the kitchen is triggered several times in a row without another sensor event interrupting the sequence, the series of identical motion sensor events will be combined into one event with a longer duration. This allows our algorithm to have a more compact representation of activities and to allow similar activities to be more easily compared. We refer to the extended list of states for a pattern p as $E(p)$.

To calculate the similarity between two activities X and Y, we compute the distance between their extended state lists $E(X)$ and $E(Y)$ using our general edit distance to account for the state information and the order mapping. In particular, we compute the number of edit operations that are required to make activity X the same as activity Y. The edit operations include adding a step or deleting a step (traditional edit distance), re-ordering a step (order distance), or changing the attributes of a step (for this application step attributes include the event duration and event frequencies). The general edit distance $e_g(X, Y)$ for two patterns X and Y can be defined based on a combination of the traditional edit distance ($e(X, Y)$), the order distance($e_o(X, Y)$), and the attribute distance ($e_a(X, Y)$) between X and Y, as calculated in Equation (3.9), where we refer to the additional term added to the traditional edit distance as Δ.

Note that this similarity metric is different from the one used in the previous section because state information such as event duration and ordering is taken into account.

$$\begin{aligned} e_g(X, Y) &= e(X, Y) + e_o(X, Y) + e_a(X, Y) \\ &= e(X, Y) + \Delta(X, Y) \\ &= e(X, Y) + \sum_{\substack{x \in E(X) \\ y \in E(Y), m(x)=y}} \Delta(x, y) \end{aligned} \tag{3.9}$$

It should also be noted that there will be multiple possible mappings between

the states of the two activities X and Y, and one mapping will lead to the shortest edit distance (the optimal mapping). To find the optimal mapping for a state $x \in E(X)$ (denoted by $m(x)$), the value of $\Delta(x, y)$ is computed for each possible mapping of $x \rightarrow (y \in E(Y))$; the state y is chosen to minimize $\Delta(x, y)$ as in Equation (3.10).

$$\begin{aligned} m(x) &= argmin_y(\Delta(x, y)) \\ &= argmin_y(e_o(x, y) + e_a(x, y)) \end{aligned} \tag{3.10}$$

The attribute distance between two states x and y is calculated as the sum of distances between individual attributes. An important attribute that can help us determine similarity is temporal information attribute (e.g. duration). The order distance between the two states x and y is defined as in Equation (3.11) where $pos(x)$ shows the index of state x in the corresponding list of states $E(X)$.

$$e_o(x, y) = |\frac{pos(x)}{|E(X)|} - \frac{pos(y)}{|E(Y)|}| \tag{3.11}$$

Based on the above descriptions, we can rewrite the term Δ in (3.9) as in (3.12):

$$\Delta = \frac{\sum_x e_o(x, m(x)) + e_a(x, m(x))}{max(|E(X)|, |E(Y)|)} \tag{3.12}$$

The general edit distance gives us a measure to compare activities and also to define cluster centroids. The cluster centroid for each cluster is defined as the activity that has the highest degree of similarity with all the other activities in the

same cluster, or equivalently the lowest edit distance to all the other activities in the cluster. Each cluster representative represents a class of similar activities, forming a compact representation of all the activities in the cluster. The activities represented by the final set of clusters are those that are modeled and recognized by the CASAS smart environment.

It should be noted that currently the number of clusters is provided to the clustering algorithm. However, alternative methods can be used to determine the number of clusters during runtime, by forming incremental clusters until no more change can be perceived, as proposed in the literature [Sugar et al., 2003, Milligan and Cooper, 1985, Yang and Wang, 2002, Noh et al., 2006].

3.2.2 Recognizing Activities

Once the activities are discovered for a particular individual, we want to build a model that will recognize future executions of the activity. This will allow the smart environment to track each activity and determine if an individual's routine is being maintained. As described earlier, researchers have exploited the use of probabilistic models for activity recognition with some success for pre-defined activities. In our approach, we make use of a hidden Markov model to recognize activities from sensor data as they are being performed. Each model is trained to recognize the patterns

that correspond to the cluster representatives found by our algorithm in previous steps.

As mentioned in Section 2.2, a Markov Model (MM) is a statistical model of a dynamic system, which models the system using a finite set of states. Because we are processing data that is collected in real homes in which activities may be interrupted or interleaved, this type of approach would not be as effective. For this task, we employ a hidden Markov model. As with a Markov chain, the conditional probability distribution of any hidden state depends only on the value of a finite number of preceding hidden states. In other words, the observable variable at time t, namely x_t, depends only on the hidden variable y_t at that time slice.

We can specify an HMM using three probability distributions: the distribution over initial states $\Pi = \{\pi_k\}$, the state transition probability distribution $A = \{a_{kl}\}$, with $a_{kl} = p(y_t = l|y_{t-1} = k)$ representing the probability of transitioning from state k to state l; and the observation distribution $B = \{b_{il}\}$, with $b_{il} = p(x_t = i|y_t = l)$ indicating the probability that the state l would generate observation $x_t = i$. We can find the most likely sequence of hidden states given the observation in Equation (3.13) and by using the Viterbi algorithm[Viterbi, 1967].

$$argmax_{x_1 \ldots x_t} P(y_1, ..., y_t, y_{t+1}|x_{1:t+1}) \tag{3.13}$$

Though HMMs can prove to be useful in predicting activity labels, sometimes

they make a very slow transition from one activity to another. For example, consider a case where a HMM is currently in some state y_1 as the most likely activity, but the next sensor event belongs to some other activity y_2. It will take the HMM several sensor events to slowly decrease the probability of activity y_1 and increase the probability of activity y_2. This problem is heightened when residents act in a natural manner and interweave multiple activities. To remedy this problem we use an event-based sliding window that limits the history of sensor events that the model remembers at any given time. The probability values calculated previously are flushed out whenever the model starts processing a new window.

For activity recognition, we use a voting multi-HMM model as a boosting mechanism. Boosting and other ensemble learning methods attempt to combine multiple hypotheses from a number of learning algorithms into a single hypothesis [Meir and Rätsch, 2003, Schapire and Singer, 1999, Freund and Schapire, 1995]. While the value of boosting for classification has been shown, research in the application of boosting to sequence learning has been comparatively limited [International, 1999, Dimitrakakis and Bengio, 2004]. We construct multiple HMMs and recognize activities by combining their classifications using a voting mechanism. To generate an activity label L for a particular sensor event x, we apply the Viterbi algorithm to the sliding window of events that ends in event x for each HMM and choose the activity that receives the highest number of votes. For each individual HMM we let the hidden states represent

the possible activities and we encode observable states to represent sensor values. The multiple HMMs in the multi-HMM model represent alternative variations of the patterns. Specifically, the first HMM represents the first variation of all patterns (one hidden state per pattern), the second HMM represents the second variation of patterns, and so on. The activity label, $L(x)$, is calculated as in Equation (3.14) where $P_k(x, L_i)$ shows the probability of assigning label L_i to x by the k-th HMM. In this equation n is the number of HMMs.

$$L_m(x) = argmax_i(\frac{\sum_{k=1}^{n} P_k(x, L_i)}{n}) \tag{3.14}$$

As an example of our model, Fig. 3.3 shows a portion of an individual HMM for activity data collected in one of the CASAS smart apartments. The probabilistic relationships between hidden nodes and observable nodes, and the probabilistic transitions between hidden nodes, are estimated by the relative frequency with which these relationships occur in the sample data corresponding to the activity cluster.

Given an input sequence of sensor events, our goal is to find the most likely sequence of hidden states, or activities, which could have generated the observed event sequence. We use the Viterbi algorithm [Viterbi, 1967] for each HMM to identify this sequence of hidden states, one hidden state at a time, and then using the described voting mechanism, we identify the most likely hidden state for the multi-HMM based on input from all individual HMMs (see Fig. 3.4). The multi HMM is

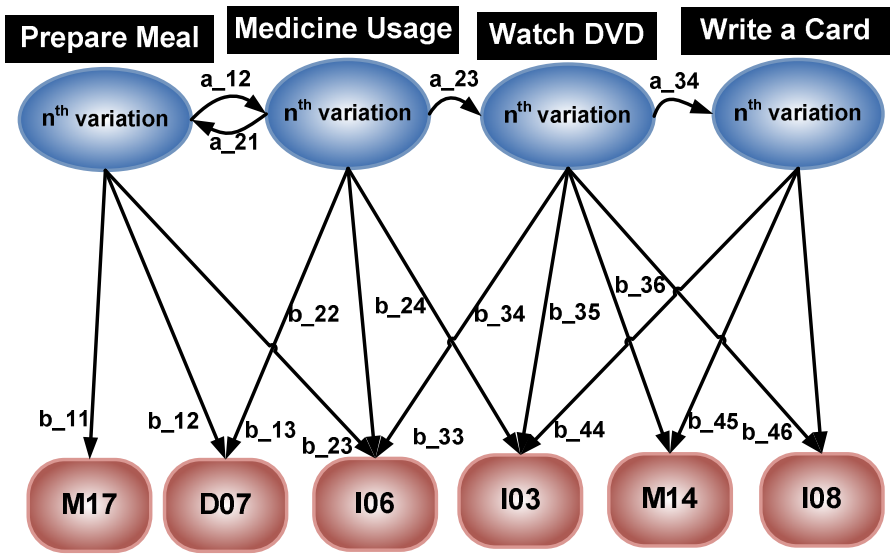


Figure 3.3: A section of an individual HMM, representing the n^{th} variation of patterns. The ovals represent hidden states (i.e., activities) and the rectangles represent observable states. Values on horizontal edges represent transition probabilities a_{ij} between activities and values on vertical edges represent the emission probability b_{kl} of the observable state given a particular current hidden state.

built automatically using the output of our discovery and clustering algorithm.

3.2.3 Experimental Results

We hypothesize that our algorithm will accurately identify activities that are frequently performed in a smart environment. We also hypothesize that the algorithm

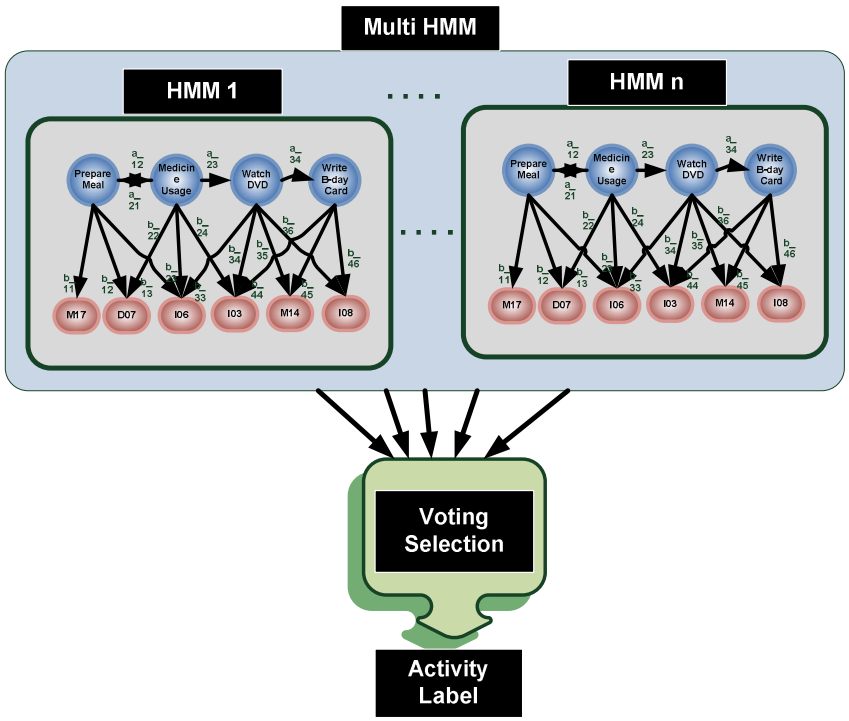


Figure 3.4: A multi HMM consists of n HMMs. It uses a voting mechanism to choose the final output.

can be used to track the occurrence of these regular activities. We validate these hypotheses here using data collected in a physical smart environment.

Experiment Testbed

The testbed for validating our algorithms was a three-bedroom apartment located on the Washington State University campus as part of the CASAS smart home project. As shown in Fig. 3.5, the smart apartment testbed includes three bedrooms,

one bathroom, a kitchen, and a living / dining room. The apartment is equipped with motion sensors positioned on the ceiling approximately 1 meter apart throughout the space. In addition, we have installed sensors to provide ambient temperature readings, and custom-built analog sensors to provide readings for hot water, cold water, and stove burner use. Voice over IP using the Asterisk software [Guru, 2009] captures phone usage, contact switch sensors monitoring the open/closed status of doors and cabinets, and pressure sensors monitor usage of key items such as the medicine container, cooking pot, and phone book. Sensor data is captured using a sensor network that was designed in-house and is stored in a SQL database. Our middleware uses a jabber-based publish/subscribe protocol [Jabber, 2009] as a lightweight platform and language-independent middleware to push data to client tools with minimal overhead and maximal flexibility. To maintain privacy we remove participant names and identifying information and encrypt collected data before it is transmitted over the network.

Normal Activity Discovery Results

For our first experiment, we applied our algorithm to data that is collected in our testbed. Specifically, we gather data for a collection of repeated specific, scripted activities and analyze the data using our algorithm. Because the activities are repeated a number of times, our algorithm should discover activities that correspond to a high degree with the pre-selected activities. If it is successful in discovering

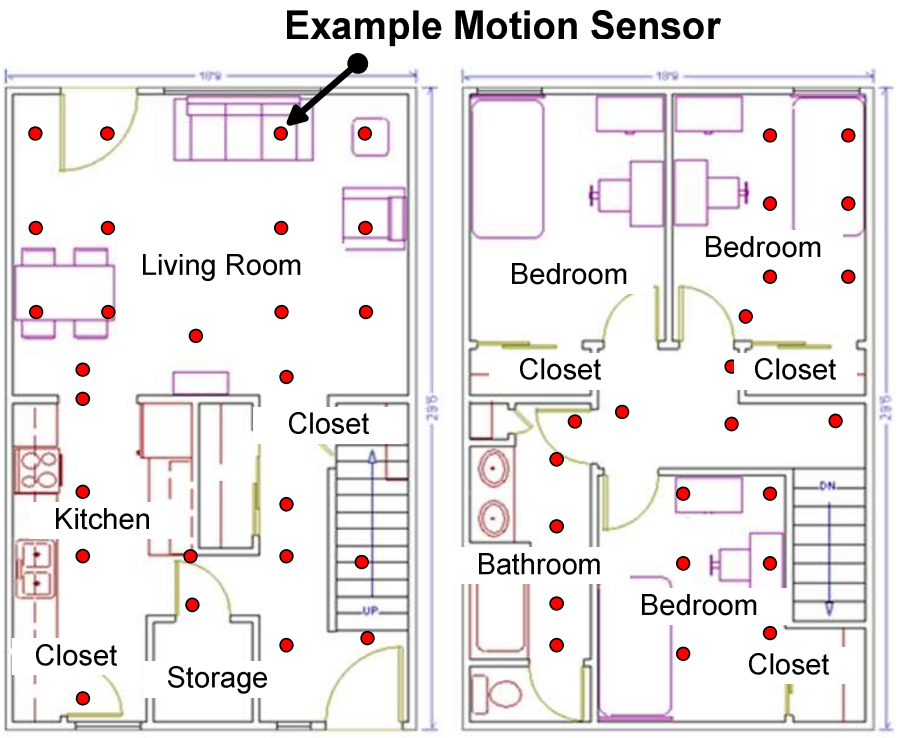


Figure 3.5: Three-bedroom smart apartment used for our data collection. The positions of motion sensors are indicated by circles in the figure.

these activities, this will provide evidence that our unsupervised learning method will automatically identify, recognize, and track sensor event sequences that intuitively represent regular activities, and that our method will identify activities of interest if they occur frequently.

To provide physical training data, we brought 20 WSU undergraduate students recruited from the psychology subject pool into the smart apartment, one at a time,

and had them perform the following five activities:

1. **Telephone Use**: Look up a specified number in a phone book, call the number, and write down the cooking directions given on the recorded message.

2. **Hand Washing**: Wash hands in the kitchen sink.

3. **Meal Preparation**: Cook oatmeal on the stove according to the recorded directions, adding brown sugar and raisins (from the kitchen cabinet) once done.

4. **Eating and Medication Use**: Eat the oatmeal together with a glass of water and medicine (a piece of candy).

5. **Cleaning**: Clean and put away the dishes and ingredients.

The selected activities include both basic and more complex ADLs that are found in clinical questionnaires [Reisberg et al., 2001]. Noted difficulties in these areas can help identify individuals who may be having difficulty functioning independently at home [Schmitter-Edgecombe et al., 2008]. As shown in Fig. 3.6, each sensor reading is tagged with the date and time of the event, the ID of the sensor that generated the event, and the sensor value. Notice that performing activities by different subjects results in considerable inter-subject variability, as participants were performing activities in vastly different ways.

To validate the effectiveness of our activity discovery algorithm, we applied our algorithm to the sensor data collected for the normal activities. Specifically, we first

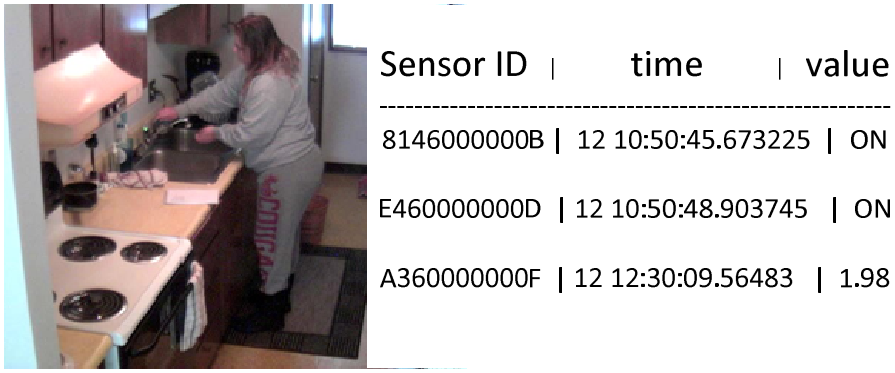


Figure 3.6: Resident performing "hand washing" activity (left). This activity triggers motion sensor ON/OFF events as well as water flow sensor values (right).

discover repeating sequential patterns in the sensor event data. We then cluster the pattern instances into five clusters (the same number as the number of activities in the scripted experiment) and determine if the discovered activities are similar to those that are pre-defined to exist in the sensor data.

In these experiments, we set the minimum compression thresholds C and C_v to 0.3 and 0.1, and the threshold of frequent events used α to 0.6. These are values we found to be effective based on experimentation across multiple datasets. When we analyzed the collected sensor events, DVSM discovered 9 general patterns with lengths varying from 2 to 39 events, and comprising up to 10 variations for each pattern. These results indicate that DVSM is able to find repetitive patterns in a compact form from 100 activity sensor streams, despite the inter-subject variability.

In the next step, we cluster the discovered activities. The resulting clusters provide an even more compact representation of all activities by assigning a centroid pattern for each cluster of similar activities. The attributes considered in these set of activities were duration of states and frequency. As mentioned earlier, the number of clusters was set to 5, equal to the number of pre-defined activities that occurred in the data collection.

In order to determine the ability of our algorithm to find these pre-defined activities, we compare the representatives of the automatically-discovered clusters with the sensor event sequences that occur for the pre-defined tasks. Because our algorithm would eventually be used to find naturally-occurring patterns for each individual, we repeat the discovery and assessment process for each of the participant data files, representing a total of 120 cases. We are interested in determining the percentage of cluster representatives that match the pre-defined activity sensor events. We consider a cluster representative as matching a pre-defined activity if it only contains events that occur within the activity sequence and if it does not overlap with any of the other cluster representative sequences.

We also evaluated the quality of the clusters. The purpose of the clustering algorithm in this project is ultimately to provide groupings of sensor sequences that reflect regularly-performed activities and that can be recognized. The primary performance criteria thus applies to the entire discovery and recognition approach rather than just

to the clustering component. However, we do evaluate the clusters themselves based on two metrics defined below.

1. First, we compute the fraction of clusters that map to the actual defined activity groups. If the number of actual defined activity groups is denoted by $|A|$, and the number of discovered clusters where the representative's label maps to a distinct activity group is $|S|$, then our first cluster quality metric "RepQuality" can be expressed as in Equation (3.15):

$$RepQuality = \frac{|S|}{|A|} \tag{3.15}$$

2. Second, we compute the fraction of activities in each cluster that actually belong to the same defined activity group represented by the cluster representative (e.g., watching DVD). If we denote the cluster S_i's representative by m_i and its actual activity label as $L(m_i)$, and we also denote each activity in the cluster as a_j, its actual activity label as $L(a_j)$, and its discovered label as $\mathcal{L}(a_j)$, then our second cluster quality metric "ClusterCohesion" will be defined as in Equation (3.16):

$$ClusterCohesion = \frac{\sum_{j=1}^{|S_i|} \delta(a_j)}{|S_i|} \tag{3.16}$$

$$\delta = \begin{cases} 0 & if \quad L(a_j) \neq \mathcal{L} \\ 1 & otherwise. \end{cases}$$

To be able to apply such metrics, in our experiments we embedded the actual labels of each sensor event. Note that these annotations do not play a role in the discovery and recognition of the activities; rather the whole purpose of using such annotation is to be able to measure the accuracy of our algorithms at the end, and one can easily remove these annotations in a deployed version of the system.

In our experiment, our algorithm found cluster representatives corresponding to the pre-defined activities for 80.0% of the cases (RepQuality). In addition, 87.5% of the individual sensor events were assigned to the correct clusters, or to the activity clusters that actually were responsible for generating the events (ClusterCohesion).

Some of the activities were assigned to a wrong but similar cluster. For example, because the set of sensor events generated by the hand washing and dish cleaning tasks are very similar, in many cases they were clustered together. A similar result occurred for DVSM as well, where in many cases the algorithm considered these two patterns to be variations of each other. To an extent this highlights the fact that some ADL activities are in fact very similar and could possibly be monitored together as a group. In addition, the large inter-subject variability made assigning an activity more difficult, because participants were performing activities in vastly different ways with

missed and mistaken steps, such that in some cases the clustering algorithm clustered them into two different clusters. In a real world situation, as the discovery is usually performed for a single individual, we anticipate the accuracy will be higher. Refining the process of creating an initial assignment of clusters can also improve the accuracy.

Next, we used our multi-HMM model to recognize the discovered activities. Using such a model, our multi-HMM model was able to recognize 73.8% of the original activities and 95.2% of the activities that were discovered by our algorithm. To provide a comparative analysis, we implemented a simpler clustering method that uses a traditional edit distance measure instead of our general edit distance measure to generate clusters. Using this simpler clustering method, the multi-HMM only achieved a recognition accuracy of 61.0% for the original activities, which indicates that considering additional information such as temporal features can improve the accuracy.

In order to determine the effect that the clustering has on the entire activity recognition and discovery process, we perform a separate set of experiments to discover sequences and recognize the corresponding patterns without involving the clustering set. We randomly chose a number of activities from the DVSM patterns to act as the Markov model hidden nodes and then we used the constructed HMMs to recognize the activities. We expect that the resulting model will achieve a lower recognition accuracy because the randomly-selected activities might not be good rep-

resentatives of the whole set of activities. It is possible, for example, that half of the hidden nodes might actually represent the same activity, such as watching a DVD, because they are selected randomly and there is no criteria to eliminate dissimilarity, as we applied for forming clusters. The results of the experiments confirm our hypothesis, such that for the normal activity recognition the constructed HMMs were able to recognize on average only 48.6% of the original activities over 10 runs (compared to 73.8% when clustering was used).

These results show that the clustering step can improve the choice of activities for hidden nodes as it provides a more distinct set of activity groups with less possible overlaps and most dissimilarity.

Interwoven ADL Activity Discovery Results

In our second experiment, we again examine how well our algorithm can identify activities that are performed in a pre-scripted manner in the CASAS testbed. In this case, we complicate the situation by allowing the activities to be interwoven together when they are performed. Because our algorithm considers the disruption of sequences as part of its discovery process, we hypothesize that it will still be able to discover many of these pre-selected activities. To provide physical training data for our algorithm, we recruited 20 additional volunteer participants to perform a series of activities in the smart apartment, one at a time. For this study, we selected 8 ADL activities:

1. **Fill medication dispenser**: Here the participant removes the items from kitchen cupboard and fills the medication dispenser using the space on the kitchen counter.

2. **Watch DVD**: The participant selects the DVD labeled "Good Morning America" located on the shelf below the TV and watches it on the TV. After watching it, the participant turns off the TV and returns the DVD to the shelf.

3. **Water plants**: For this activity, the participant takes the watering can from the supply closet and lightly waters the 3 apartment plants, 2 of which are located on the kitchen windowsill and the third is located on the living room table. After finishing, he/she empties any extra water from the watering can into the sink and returns the watering can to the supply closet.

4. **Converse on Phone**: Here the participant answers the phone when it rings and hangs up after finishing the conversation. The conversation includes several questions about the DVD show that the participant watched as part of activity 2.

5. **Write Birthday Card**: The participant writes a birthday wish inside the birthday card and a fills out a check in a suitable amount for a birthday gift, using the supplies located on the dining room table. He/she then places the card and the check in an envelope and appropriately addresses the envelope.

6. **Prepare meal**: The participant uses the supplies located in the kitchen cupboard to prepare a cup of noodle soup according to the directions on the container. He/she also fills a glass with water using the pitcher of water located on the top shelf of the refrigerator.

7. **Sweep and dust**: For this task, the participant sweeps the kitchen floor and dusts the dining and the living room using the supplies located in the kitchen closet.

8. **Select an outfit**: Lastly, the participant selects an outfit from the clothes closet to be worn on an important job interview. He/she then lays out the selected clothes on the living room couch.

We instructed the participants to perform all of the activities by interweaving them in any fashion they liked with a goal of being efficient in performing the tasks. The order in which activities were performed and were interwoven was left to the discretion of the participant. Because different participants interwove the tasks differently, the resulting data set was rich and complex.

Similar to the previous experiment, we first ran DVSM on the datasets containing 160 activities, and then clustered the discovered patterns. The parameter values were defined as in the previous experiment, with the exception that the number of clusters was set to 8 to be equal to the new number of pre-defined activities. When it

was applied to the collected sensor data, DVSM was able to find 11 general patterns. Averaging over 10 runs, our algorithm found cluster representatives corresponding to the original activities in 76.4% of the participant datasets (q_1), and 88.2% of the sensor events were assigned to the correct clusters (q_2). The results of this experiment show that despite the fact that the activities in this second set are interwoven and include a larger number of activities, due to the discontinuous order-varied nature of our algorithm, it is still able to discover activities that occur frequently and will likely discover ADL activities if they are performed frequently in a smart environment.

The next step of the process is to recognize and track the discovered activities as they occur in the smart apartment. We constructed our multi-HMM model based on the discovered eight activities and applied the Viterbi algorithm to the remaining data to identify when the discovered activities occurred. Using such a model, our multi-HMM model was able to recognize 77.3% of the original activities and 94.9% of the activities discovered by our algorithm. We also performed the experiment without clustering, as mentioned in the previous experiment for normal activities, which resulted in a recognition accuracy of 55.3% of the original activities (compared to 77.3% with clustering). This again shows the importance of clustering for forming more distinct activity groups that can be used as hidden nodes of the HMM model. In another experiment, a simple clustering method using the traditional edit distance resulted in a lower accuracy of 62.4% for the original activities, which again highlights

the importance of incorporating additional context information into the construction of the activity models.

Long Term Activity Discovery Results

The first two experiments validated that our algorithm is able to discover frequent activities, including those that belong to known sets of ADL activities. A possible use of this technology is to perform activity discovery during a time when the resident is healthy and functionally independent, to establish a baseline of normal daily activities. By modeling and recognizing the activities, our algorithm can then track the activities as they are performed in the smart home. The resident or a caregiver can look at the reported activity times to determine whether activities are being performed as regularly as in the past. Alternatively, a variety of temporal analysis and data mining algorithms could be applied to detect trends in the frequency and regularity of the activities.

To demonstrate how our algorithm can be used for a combination of activity discovery, recognition, and tracking, we applied this process to a long-term data collection in our CASAS smart apartment. In this experiment, we collected 3 months of daily activity data from the smart apartment while two residents lived there and performed their normal daily routines. Sensor data was collected continuously, resulting in $987,176$ sensor events. We applied the activity discovery algorithms to this collected data. The parameter settings were similar to the previous experiments,

although we modified the threshold α of frequent events used in pattern discovery in order to investigate its impact on the number of discovered patterns.

We note that increasing the value of α results in discovering more patterns, as a wider range of frequent events are involved, but at the same time results in pruning more patterns too. As Fig. 3.7 shows, the number of patterns ranged from 10 ($\alpha = 10\%$) to 794 ($\alpha = 90\%$). This validates that even when the top 90% of frequent events are captured from three months of data, our algorithm is still able to provide a rather compact representation of the activity patterns. We also note that the pruning process removes a large number of patterns, considerably reducing the number of redundant patterns (see Fig. 3.8).

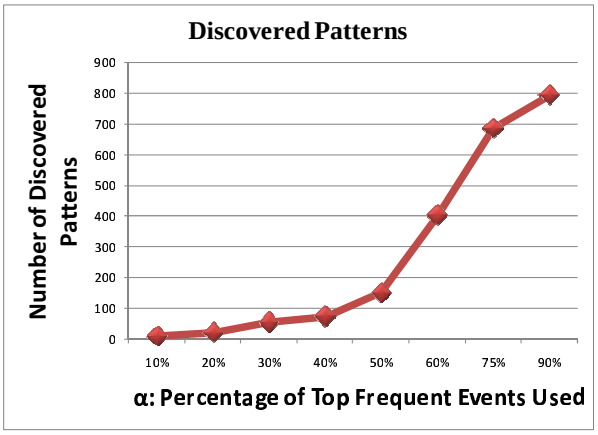


Figure 3.7: Number of discovered patterns as a function of α.

Also note that the compression thresholds (C and C_v) control the way that the "frequent continuous" patterns and the "frequent continuous variations" of pat-

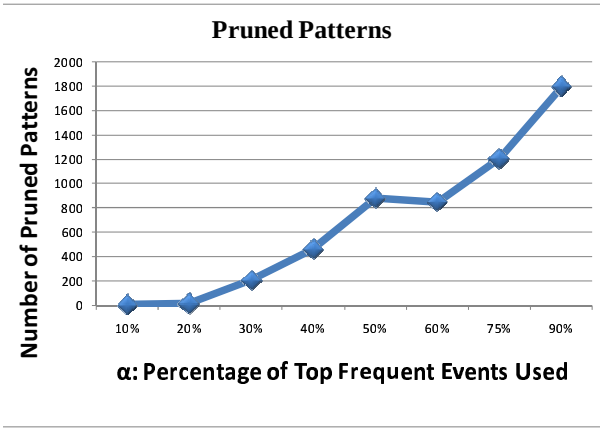


Figure 3.8: Number of pruned patterns as a function of α.

terns are discovered. We performed an experiment to observe the effect of changing the compression threshold. The experiments were run with α set to 30% where we changed C and C_v systematically. The lower we set the C compression threshold, the more patterns will be discovered as this allows for the relatively "less frequent" and "less continuous" patterns to be also included among the interesting patterns. Similarly, the lower we set the compression threshold C_v, the more pattern variations will be discovered (see Fig. 3.9). In our application we wanted to see how variations of patterns can be discovered, so we set C_v to a low number.

After discovering sequential patterns in the sensor event data, we clustered the discovered patterns into a maximum of 10, 15 and 20 clusters. Again, to investigate the impact of the number of discovered patterns on the number of formed clusters,

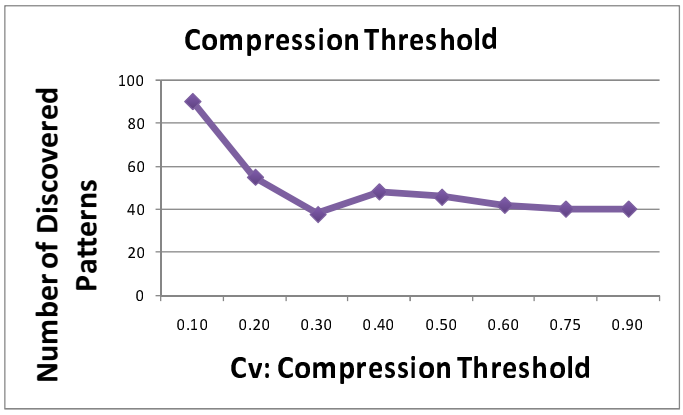


Figure 3.9: Number of discovered patterns as a function of C_v.

we examined the results of varying the α threshold. For smaller values of α, the clusters tended to merge together, as there were less distinguishable patterns. As we increase α and therefore the number of discovered patterns, more distinctive clusters were formed. Once a threshold value of α was reached ($\alpha = 30\%$), the number of clusters remained the same, because fewer distinguishable patterns were discovered. These results are graphed in Fig. 3.10.

Next, we used our multi-HMM model to track the activities that had been identified by our model. To verify the discovered activities, we presented the discovered activities to the residents to see if they can recognize any of those activities as their daily routines. For example, a shortened form of one discovered pattern is $\langle M06, M07, M15, M17 \rangle$, which was interpreted by the residents as one person watch-

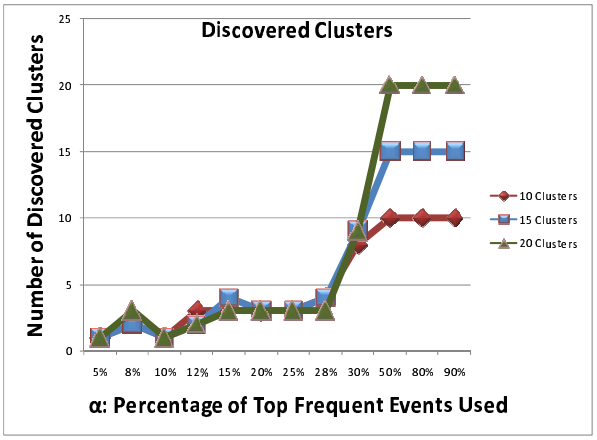


Figure 3.10: Number of discovered clusters as a function of α.

ing TV from the couch, getting up for a snack, then going back to the couch. Some of the patterns that were identified and tracked by the multi-HMM model are highlighted in Fig. 3.11. These patterns include preparing a meal, using the bathroom, watching TV/getting snack and resting in the bedroom (after working on the computer). These results show that DVSM can then track when the activities are performed on a continual basis in the smart home, and can be applied to large datasets collected over a long period of time.

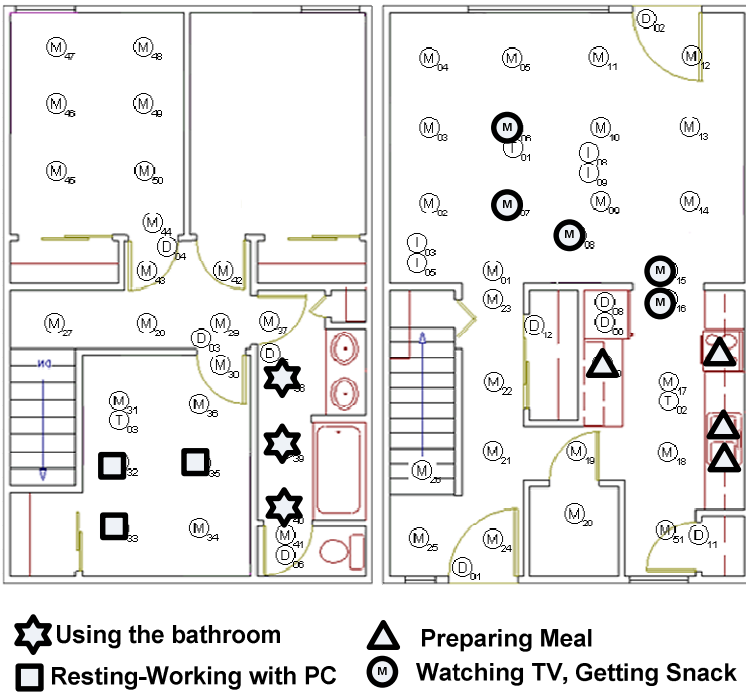


Figure 3.11: Visualization of selected discovered patterns (simplified).

3.3 COM

In Section 3.2 we introduced the DVSM method which is a sequence mining method to find discontinuous and varied order patterns in the data. In this section we introduce an improved version of DVSM. Though we showed that DVSM is able to discover some interesting patterns from sensor event data streams, the method still faces some shortcomings. DVSM works best when using scripted data or data collected under controlled conditions. It faces some difficulty mining real life data.

For example if the activities performed in different regions of home have different frequencies, or if heterogeneous sensors are used (such as motion sensors in combination with contact switch sensors), some of the patterns will not be discovered.

In this section, we introduce COM as an improved version of DVSM. COM stands for the **C**ontinuous, varied **O**rder, **M**ulti Threshold activity discovery method. COM not only discovers discontinuous patterns and their variations, but is also able to better handle real life data by dealing with different frequencies/sensors problem. It is able to find a higher percentage of the frequent patterns, and thus achieving a better discovery and recognition accuracy. Also by pruning irrelevant patterns based on mutual information the method retains only the relevant variations of the patterns, reducing the number of irrelevant variations. COM adopts a more automated approach by eliminating the need for the user to configure some of the parameters, such as the percentage of the top frequent sensor events that should be used to discover the activity patterns, the support threshold for frequent events, or the number of activities. Automating such configurations results in a more overall automated approach. We also provide a pattern visualizer component that is able to visualize the activity patterns and their variations in order to help the users to better identify the variations of the patterns and as a result to detect abnormal or suspicious cases.

The remainder of this chapter is organized as follows. First we describe our approach in more detail, including its three main stages. The first stage discovers

activities by mining data and extracting activity models, while the second stage summarizes the discovered patterns by clustering, and the third stage is responsible for recognizing activities. We then show the results of our experiments on data obtained from two different smart apartments.

3.3.1 COM Model Description

Similar to DVSM, our objective here is again to develop a method that can automatically discover real life patterns of a resident's activities, even if the patterns are somehow discontinuous or have different event orders across their instances. After discovering such patterns through our mining method, we will then summarize and group the discovered patterns to provide a more concise and compact representation using our hierarchical clustering method. Unlike the DVSM method, here we use a clustering method that is automatically able to determine the number of clusters. The clusters are then used to track and recognize the resident's activities. Again we assume that the data is not annotated and the activity boundaries are not specified, i.e. we only have access to unlabeled sensor data.

We already briefly mentioned the problem of different frequency/sensor types for pattern discovery that was ignored in DVSM. By not taking into account the differences in sensor event frequencies across different regions of the space, the patterns

that occur in less frequently used areas of the space might be ignored. For example, if the resident spends most of his/her time in the living-room during the day and only goes to the bedroom for sleeping, then the sensors will be triggered more frequently in the living-room than in the bedroom. Therefore when looking for frequent patterns, the sensor events in the bedroom might be ignored and consequently the sleep pattern might not be discovered. The same problem happens with different types of sensors, as usually the motion sensors are triggered much more frequently than other type of sensors such as cabinet sensors. This problem is known among the data mining community as the "rare item problem" and has been addressed by providing multiple support thresholds when mining association rules or sequential patterns [Liu et al., 1999].

To deal with the sheer volume of pattern instances that we might encounter during pattern generation, we prune the irrelevant variations of patterns based on mutual information [Guyon and Elisseeff, 2003]. Besides pruning irrelevant variations, we also prune the non-maximal, infrequent or highly discontinuous patterns. In order to provide a more automated approach, we do not require the user to provide parameters such as the number of activities. We also provide a visualizer to represent the daily activity patterns and variations to the users/care-givers in a natural, user-friendly way. The architecture of the system can be seen in Figure 3.12, including its mining, clustering, recognition and visualization components.

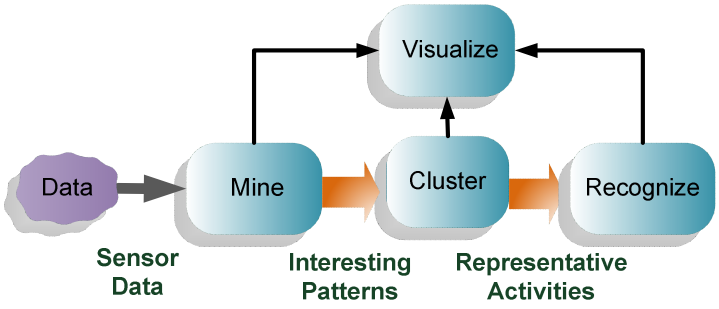


Figure 3.12: Main components of COM for discovering, modeling and recognizing patterns of activities.

The input data to the system is a sequence of sensor events, where each event e appears in the form $e = \langle t, s \rangle$ where t denotes a timestamp, and s denotes a sensor ID. Each sensor ID is associated with its room name (e.g. kitchen) which we will refer to as a location tag L. We define a pattern instance a as a sequence of n sensor events $a = \langle e_1, e_2, ..e_n \rangle$. A pattern itself represents the collection of all of its instances. An example of an activity pattern such as meal preparation is the sensor event sequence $\langle M005, M003, M001 \rangle$ where $M005$, $M003$ and $M001$ refer to motion sensors that are activated in the kitchen. A variation of such a pattern may appear as $\langle M003, M002, M001 \rangle$. Note that in this discussion we use the terms activity and pattern interchangeably. In the following section, we will provide a more detailed description of each one of the components.

Mining

Similar to the DVSM method, using COM we denote a general pattern a as the pattern that encompasses all of its n variations, where each variation is denoted by a_i. To find patterns in data, first we will create a reduced dataset D_r from the input data $\mathcal{D}$. The reduced dataset only contains frequent sensor events that will be used for constructing the patterns. In the DVSM algorithm, the user has to specify what percentage of the top frequent events (α) should be used and then by using a global support threshold on the frequency of sensor events, the reduced dataset is created. As was already discussed, taking such an approach will result in ignoring the problem of "rare sensors". In our studies we found out that different regions of a home exhibit different sensor frequencies, as well as do different types of sensors. Here we do not require the user to identify α, and we use several support thresholds for different regions of the home and for different types of sensors, all determined automatically.

Different regions of homes are identified by the provided location tags L, corresponding to the functional areas such as bedroom, bathroom, etc. The sensors are also categorized depending on their types. Here we categorize the sensors into two categories: motion sensors and key sensors. The key sensors include every sensor except for the motions sensors, e.g. the cabinet sensors or door sensors. The key sensors basically represent the interaction of the user with the environment, while motion sensors provide a trajectory of inhabitant's motion around the home. The type and

the location tag of all sensors is passed on to COM as an initial configuration file. For each region and category, the minimum acceptable frequency is automatically derived as the average frequency for that specific region and category. More specifically, $\overline{f}_k$, the minimum acceptable frequency support of the key sensors for each region, is computed as the average frequency of key sensors in that region. Similarly, $\overline{f}_m$, the minimum acceptable frequency support for motion sensors in a specific region, is computed as the average frequency of motion sensors in that specific region. Figure 3.13 shows a depiction of selected sensors and the minimum acceptable frequency supports for one of the smart homes used in our experiments. If a region contains only motion sensors and no key sensors, we denote its $\overline{f}_k$ as N/A. One can clearly see that the motion sensors are activated more frequently in the living room than any other area in the home, with the result that the frequency support of motion sensors in the kitchen and bathroom is half of the frequency support in the living room. In the kitchen, we can see that the motion sensors are activated 3 times more than key sensors such as cabinet doors or the refrigerator door.

The patterns are then discovered as with the DVSM method. We identify general patterns/ variations as interesting if they provide a minimum compression value according to the Minimum Description Length principle [Rissanen, 1978]. The compression value c for a general pattern a and its variations is defined as with the DVSM method. Patterns with high compression values are flagged as interesting,

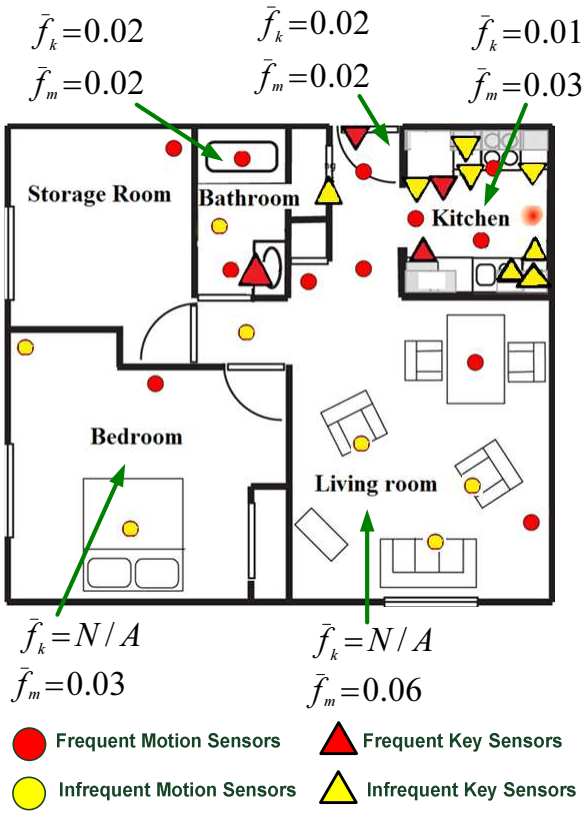


Figure 3.13: The frequent sensors are selected based on the minimum acceptable regional support, instead of a global support. Here the frequencies are normalized to be in the range [0..1].

while patterns with low compression values are discarded (i.e., such patterns are either infrequent or highly discontinuous). The same is applied to prune the highly discontinuous or infrequent variations of a general pattern.

By computing the mutual information [Guyon and Elisseeff, 2003] between the

general pattern and each of its sensor events, we are able to find the set of core sensors for each general pattern. Finding the set of core sensors allows us to prune the irrelevant variations of a pattern which do not contain the core sensors.

$$MI(s,a) = P(s,a) * \log \frac{P(s,a)}{P(s)P(a)} \tag{3.17}$$

Every iteration, we also prune redundant non-maximal general patterns; i.e., those patterns that are totally contained in another larger pattern. This multi-stage pruning process considerably reduces the number of discovered patterns, making it more efficient in practice.

We continue extending the patterns by prefix and suffix until no more interesting patterns are found. A post-processing step records attributes of the patterns, such as event durations and start times.

Clustering

Next, we group discovered patterns together to get an even more compressed representation similar to DVSM, but using a different clustering method. Our clustering algorithm is similar to conventional hierarchal agglomerative clustering techniques [Tan et al., 2005], however it doesn't form the complete hierarchy. Agglomerative clustering techniques build a hierarchy from the individual elements by progressively merging clusters until all data ends up in one cluster. Here we do not continue the hierarchal clustering up to the point of reaching a single cluster, rather the clustering

continues until the similarity between the two closest clusters drops below a threshold ζ. This gives us a set of clusters at the highest level of the hierarchy. After forming the clusters, the The cluster centroids at the highest level are used to track and recognize the resident's activities. Using such a clustering method, the user no longer has to provide the number of clusters in advance. We use a group-average link method [Tan et al., 2005] to compute the proximity matrix based on the similarity measure defined in Equation 3.18. The algorithm itself is shown in 1.

$$\Upsilon(i,j) = \Upsilon_t[i,j] + \Upsilon_d[i,j] + \Upsilon_{\mathcal{L}}[i,j] + \Upsilon_S[i,j] \tag{3.18}$$

The start times are in the form of a mixture normal distribution with means $\Theta = \langle \theta_1..\theta_r \rangle$ to better capture the variability in start times. An example of such a mixture start time distribution can be seen in Figure 3.14 which represents start times for an "eating" activity.

We can see that using a normal mixture model we are able to capture both breakfast and lunch times as the regular meals for the inhabitant. We represent start time θ in an angular form Φ measured in radians instead of a linear representation. This allows for time differences to be represented correctly (2:00 am will be closer to 12:00 am than to 5:00 am). The similarity between the two start time distributions is thus calculated using Equation 3.19.

Algorithm 1 Clustering Method

procedure CLUSTER($\mathcal{P}$) ▷ Each pattern is considered as a cluster at first

$\mathcal{C} = \mathcal{P}$

Compute Proximity Matrix, m

repeat

$sim = \max(m[p, q]) \quad \forall p, q \in \mathcal{C}$

if $sim > \zeta$ **then**

merge p, q

end if

Update m

until $sim > \zeta$

return $\mathcal{C}$

end procedure

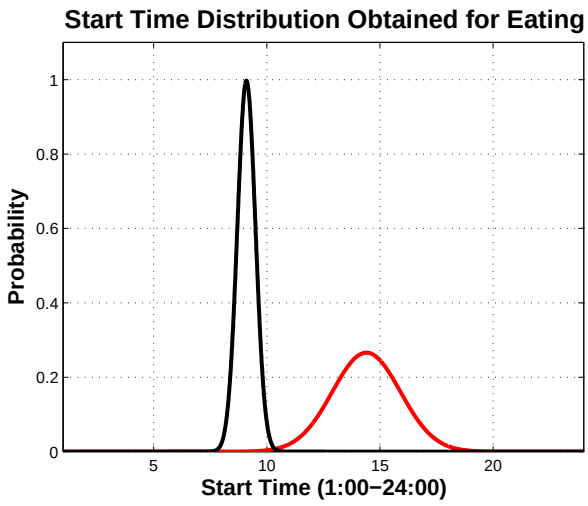


Figure 3.14: Start Time distribution as a mixture normal distribution for one of the apartments in our experiments.

$$\Upsilon_t[i,j] = \max_{\substack{\theta_1 \in \Theta_i \\ \theta_2 \in \Theta_j}} (1 - \frac{|\Phi_{\theta_2} - \Phi_{\theta_1}|}{2\pi}) \tag{3.19}$$

Duration similarity is calculated as in Equation 3.20 where durations are represented in the form of a mixture normal distribution with means $\Gamma = \langle \gamma_1..\gamma_r \rangle$.

$$\Upsilon_d[i,j] = \max_{\substack{\gamma_1 \in \Gamma_i \\ \gamma_2 \in \Gamma_j}} (1 - \frac{|\gamma_2 - \gamma_1|}{\max(\gamma_2, \gamma_1)}) \tag{3.20}$$

The regional and structural similarities are calculated as in Equations 3.21 and 3.22 using a Jaccard similarity measure [Tan et al., 2005]. In Equation 3.21, $\mathcal{E}$ refers to the set of sensors for a pattern.

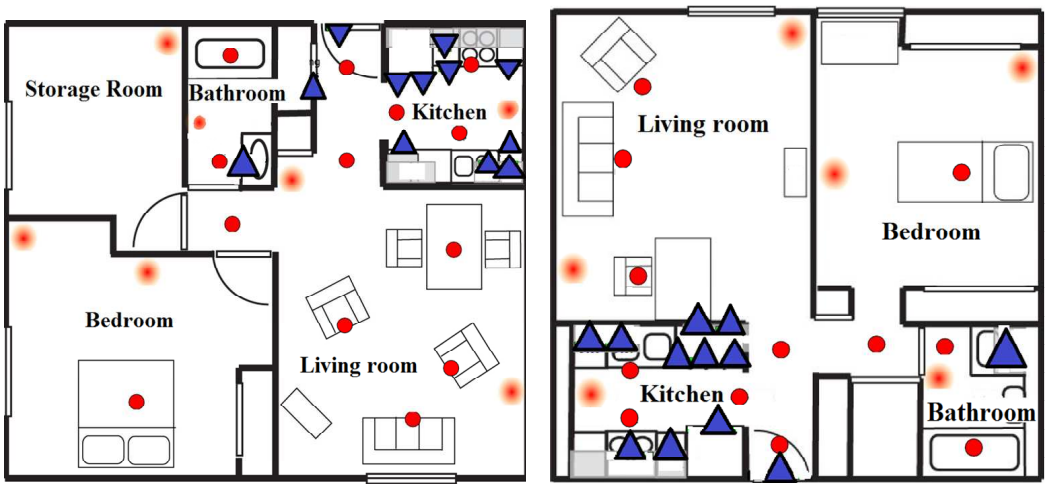
$$\Upsilon_S[i,j] = \frac{\mid \mathcal{E}_i \bigcap \mathcal{E}_j \mid}{\mid \mathcal{E}_i \bigcup \mathcal{E}_j \mid} \tag{3.21}$$

$$\Upsilon_{\mathcal{L}}[i,j] = \frac{\mid \mathcal{L}_i \bigcap \mathcal{L}_j \mid}{\mid \mathcal{L}_i \bigcup \mathcal{L}_j \mid} \tag{3.22}$$

After forming clusters, we used a hidden Markov Model to track activities, similar to the approach taken by DVSM.

3.3.2 Experiments

We evaluated the performance of our COM algorithm using the data that was collected from two different smart apartments. The layout of the apartments including sensor placement and location tags are shown in Figure 3.15. We will refer to apartments in Figures 3.15a and 3.15b as apartments 1 and apartment 2. The data was collected during an approximately three month period. Each apartment is equipped with motion sensors and contact sensors which monitor the open/closed status of doors and cabinets. A total of 10 activities were noted for each apartment. Those activities included bathing, bed-toilet transition, eating, leave/enter home, meal preparation(cooking), personal hygiene, sleeping in bed, sleeping not in bed (relaxing) and taking medicine. The first and second datasets include 3384 and 2602 annotated activity instances, respectively. Table 3.1 also summarizes apartment characteristics.



(a) Apartment 1. **(b)** Apartment 2.

Figure 3.15: Sensor map and location tags for each apartment. On the map, circles show motion sensors while triangles show switch contact sensors. The hollow-shaped motion sensors are the area motion sensors.

Dataset	Num. of Residents	Num. of Activities	Num. of Examples
B3	1	10	3384
C	1	10	2602

Table 3.1: CASAS datasets.

To be able to evaluate the results of our algorithms, each of the datasets was annotated with ADL activities of interest for the corresponding resident and apart-

ment. Unlike the scripted data from DVSM, here the data is not scripted and it was annotated for evaluation purposes.

We ran our algorithm for each one of the apartments. As mentioned before, one major improvement of our algorithm is to use multiple support thresholds for different regions of homes, as well as for different types of sensors. Though a single support threshold might not pose a problem in scripted experiments, where all activities are performed with the same frequency (e.g. 20 times as in DVSM experiments), in real life this assumption results in missing many patterns. To show how using a single threshold affects the accuracy of pattern discovery, we performed a number of experiments, once using the COM algorithm and once using DVSM.

The data mining step was able to discover a considerable number of pre-defined activities of interest. In apartment 1, it discovered 8 out of 10 activities, including bathing, leave/enter home, meal preparation (cooking), personal hygiene, sleeping in bed, sleeping not in bed (relaxing) and taking medicine. In apartment 2, it was able to discover 7 out of 10 activities including bathing, bed-toilet transition, eating, enter home, leave/enter home, meal preparation(cooking), personal hygiene, sleeping in bed, and sleeping not in bed (relaxing). Some of the patterns that have not been discovered are indeed quite difficult to spot and also in some cases less frequent. For example the housekeeping activity happens every 2-4 weeks and is not associated with any specific sensor. Also some of the similar patterns are merged together, as they

use the same set of sensors, such as eating and relaxing activities. It should be noted that some of the activities are discovered multiple times in form of different patterns, as the activity might be performed in a different motion trajectory using different sensors. Figures 3.16 and 3.17 show the number of distinct discovered activities by both COM and DVSM algorithms in apartments 1 and 2. One can clearly see that COM is able to discover a higher number of distinct activities.

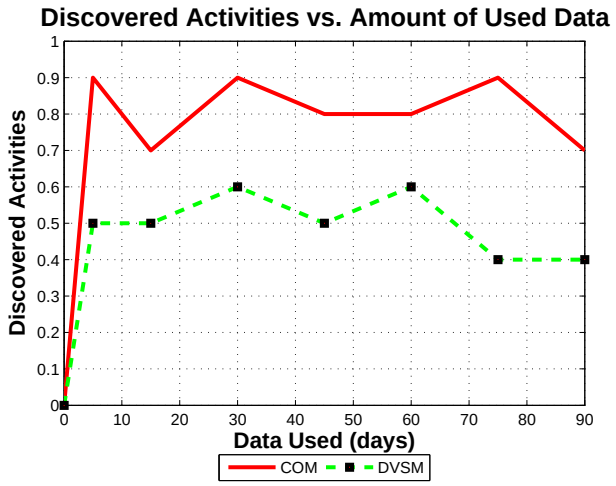


Figure 3.16: Percentage of distinct activities discovered in apartment 1 vs. the amount of data.

Using externally provided annotations to verify our results, we also computed the percentage of non-annotated discovered activities, i.e. the percentage of activities that actually have no annotation, but have been discovered by our algorithm. For apartment 1 the percentage of non-annotated activities with respect to the number of

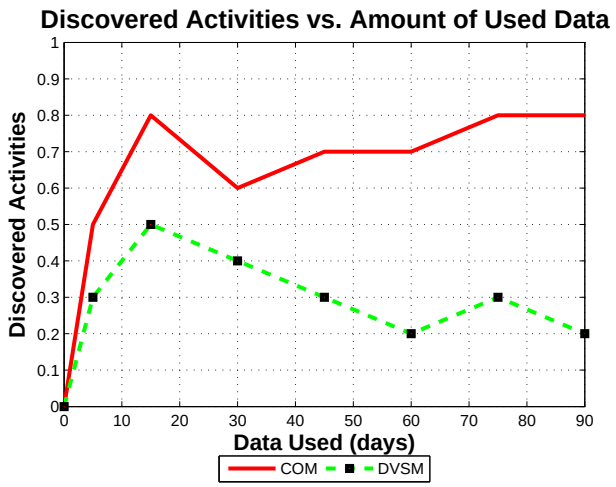


Figure 3.17: Percentage of distinct activities discovered in apartment 2 vs. the amount of data.

total distinct activities was 7.0% and for apartment 2 it was 2.0%. The low percentage of discovered activities that are not annotated shows that most of the discovered patterns are indeed well aligned with those patterns identified by the human annotator as interesting in the first place.

Figures 3.18b and 3.18e show the total number of discovered pattern *instances* for both the COM and DVSM algorithms, as well as the total number of pruned instances. It can be seen that though sometimes COM generates more pattern instances and in general more patterns, it also prunes more pattern instances due to its improved pruning capabilities, while still discovering more distinct patterns (ac-

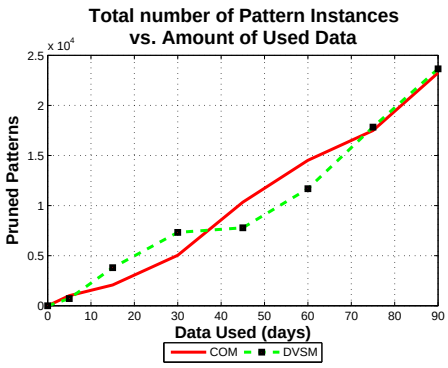
tivities).

In the next step, we clustered the discovered activities together using our agglomerative clustering method. The similarity threshold ζ of 0.75 was found to be a suitable value based on several runs of our experiments. By using the externally provided labels, we were also able to measure the purity of the clustered patterns with respect to the consistency of their variations using Equation 3.23. Equation 3.23, known in the literature as Jaccard similarity, represents a simple matching coefficient and is used to determine the similarity between clusters and actual classes. Here $|v_{11}|$ refers to the number of variations that have the same label as their general pattern, and $|v_{01}|$ and $|v_{10}|$ refer to the number of variations that have a different label other than their general pattern's label. We call this number the *variation consistency*.

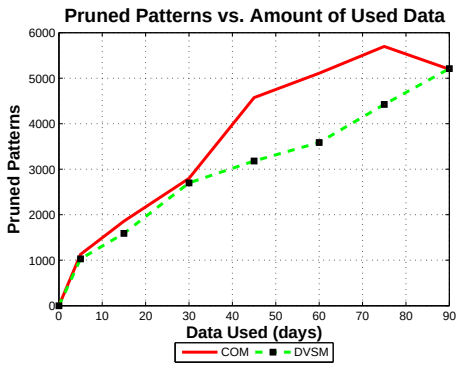
$$purity(P_i) = \frac{|v_{11}|}{|v_{11}| + |v_{01}| + |v_{10}|} \tag{3.23}$$

The variation consistency for clustered patterns in apartment 1 was 0.90 in our experiments, while for apartment 2 it was 0.76. By looking closely at the data, it was revealed that the patterns in the second dataset are much more irregular. Therefore most similar patterns are combined, such as taking medication and meal preparation which usually happen at approximately the same time and the same location (in this case, the kitchen).

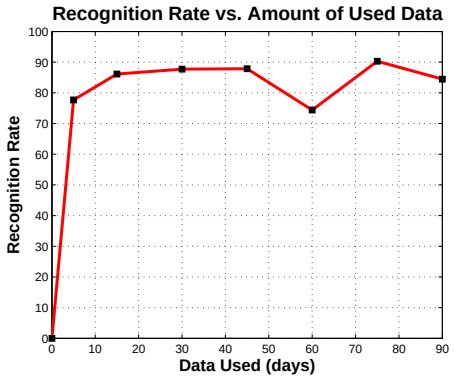
Next, using the discovered patterns an HMM was automatically constructed



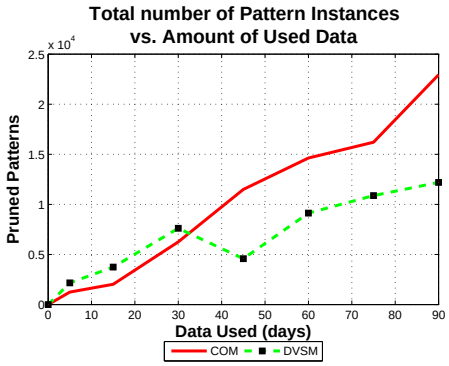
(a) Number of total pattern instances in apartment 1.



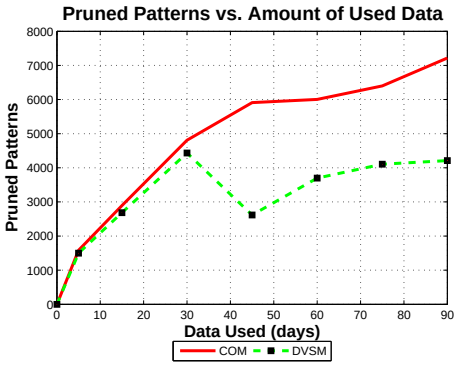
(b) Number of pruned pattern instances in apartment 1.



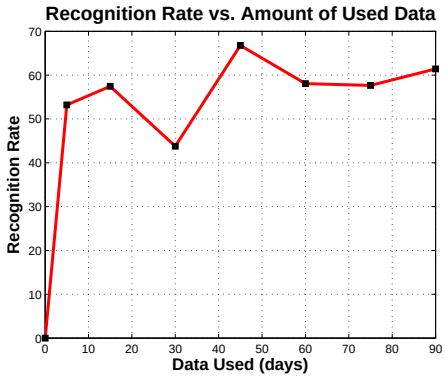
(c) HMM Recognition vs. amount of used data in apartment 1.



(d) Number of total pattern instances in apartment 2.



(e) Number of pruned pattern instances in apartment 2.



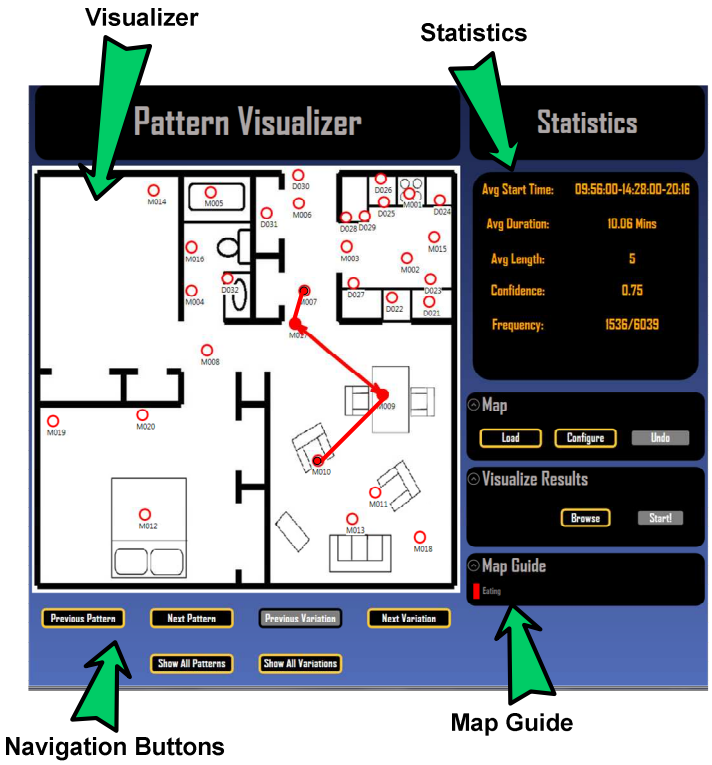
(f) HMM Recognition vs. amount of used data in apartment 2.

Figure 3.18: Results for apartment 1 and 2.

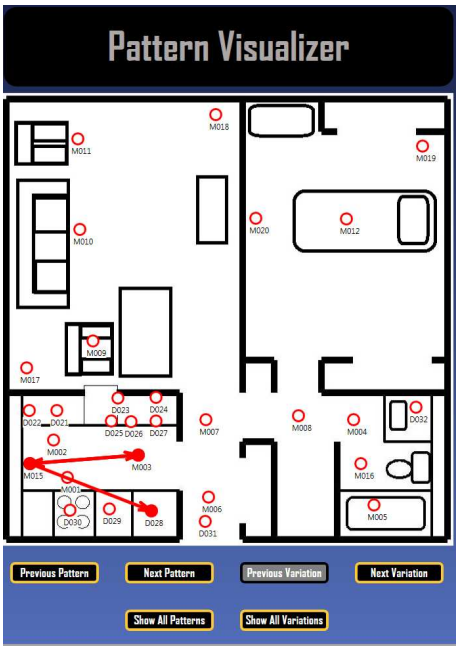
to track and recognize the activities. Figures 3.18c and 3.18f show the recognition results for the two apartments. Table 3.2 a and b show the confusion matrices for all the activities. It can be seen from those tables that it is easier to recognize and track activities in apartment 1 due to the greater behavioral regularity there. The results also reveal that despite the fact that the real life activities can be sometimes hectic and irregular, still our algorithm is able to track and monitor a considerable number of the patterns.

The results of all stages of the algorithm as a set of patterns are represented in an XML format to make it easier to share results across different components. Though XML seems to be an excellent choice for data representation, it might not be the best choice for a natural user-friendly representation. This is particularly true when considering the fact that a pattern has many features such as temporal features, pattern trajectories, etc. Analyzing these patterns manually becomes even becomes more difficult when comparing different variations of a pattern and trying to figure out their relationship. We have designed a visualizer to better help to understand the patterns and their variations. A snapshot of our visualizer can be seen in 3.19a.

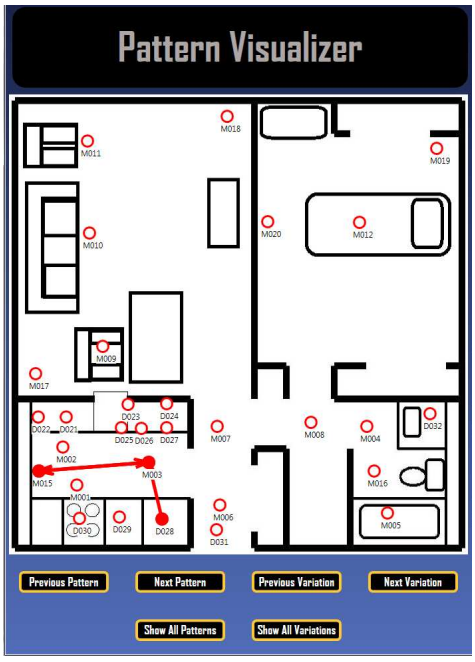
The visualizer shows the patterns on a home map, along with important statistical information such as start time, duration, and frequency shown in a separate panel as in 3.19a. The user can go back and forth between patterns as well as variations of a pattern using the navigation buttons. The user can also see all of the



(a) The pattern visualizer.



(b) "Meal Preparation" activity in apartment 2.



(c) A variation of "Meal Preparation" activity in apartment 2.

Figure 3.19: Snapshots of the visualizer.

	Hygiene	Leave	Cook	Relax	Med	Eat	Housekeep	Sleep	Bath	Bed-Toilet
Hygiene	**91.6**%	0.0%	0.0%	7.2%	0.0%	0.0%	0.0%	0.0%	0.9%	0.0%
Leave	0.0%	**88.3**%	0.0%	10.5%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%
Cook	0.1%	0.0%	**97.8**%	0.1%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Relax	0.0%	1.8%	2.2%	**95.9**%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Med	0.0%	0.2%	55.6%	0.5%	**43.5**%	0.0%	0.0%	0.0%	0.0%	0.0%
Eat	0.0%	0.0%	1.3%	98.6%	0.0%	**0.0**%	0.0%	0.0%	0.0%	0.0%
Housekeep	0.0%	0.0%	98.3%	1.6%	0.0%	0.0%	**0.0**%	0.0%	0.0%	0.0%
Sleep	0.3%	0.0%	0.0%	0.1%	0.0%	0.0%	0.0%	**99.3**%	0.0%	0.0%
Bath	26.4%	0.0%	0.0%	0.5%	0.0%	0.0%	0.0%	0.0%	**72.9**%	0.0%
Bed-Toilet	76.6%	0.0%	0.0%	19.8%	0.0%	0.0%	0.0%	2.9%	0.4%	**0.0%**

(a)

	Hygiene	Leave	Cook	Relax	Med	Eat	Housekeep	Sleep	Bath	Bed-Toilet
Hygiene	**83.9**%	0.0%	3.4%	11.8%	0.0%	0.0%	0.0%	0.0%	0.6%	0.0%
Leave	0.0%	**93.2**%	6.0%	0.7%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Cook	0.4%	0.6%	**94.8**%	4.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Relax	0.1%	0.6%	0.2%	**98.9**%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Med	0.9%	0.4%	95.7%	2.9%	**0.0**%	0.0%	0.0%	0.0%	0.0%	0.0%
Eat	1.3%	0.0%	4.6%	93.9%	0.0%	**0.0**%	0.0%	0.0%	0.0%	0.0%
Housekeep	0.8%	0.8%	93.1%	5.1%	0.0%	0.0%	**0.0**%	0.0%	0.0%	0.0%
Sleep	39.4%	0.0%	42.5%	17.9%	0.0%	0.0%	0.0%	**0.0**%	0.0%	0.0%
Bath	23.5%	0.1%	0.7%	2.3%	0.0%	0.0%	0.0%	0.0%	**73.2**%	0.0%
Bed-Toilet	83.0%	0.0%	3.2%	13.1%	0.0%	0.0%	0.0%	0.0%	0.5%	**0.0**%

(b)

Table 3.2: Confusion matrices for apartment 1 (Table a) and apartment 2 (Table b).

patterns or all of the variations of a pattern at the same time. In this case, each pattern (or variation) will be visualized using a unique color-code as depicted in the "Map Guide". Using such a simple visualizer allows users not to deal with the sensor information in a textual format, which might be confusing and hard to understand. Rather it allows the users to see the patterns in a natural format and quickly diagnose relations between different variations of a pattern.

3.4 Summary

In order to provide robust activity recognition and tracking capabilities for smart homes, we need to consider techniques for identifying the activities that should be recognized and tracked. Most current approaches use supervised methods for activity recognition. However, due to the required effort and time for annotating activity datasets it might not be very practical in real world situations to use supervised methods. Annotating activity data imposes a burden on annotators and residents and often introduces a source of error in the process.

We introduced two alternative methods for tracking activities in smart environments. In our first approach we employ our DVSM algorithm to discover frequent activities that regularly and naturally occur in a resident's environment. These activity patterns can be discontinuous or can have varied event orders. After pattern discovery, models are learned to recognize these particular activity patterns. Our second method, COM, improves DVSM further. COM not only is able to find discontinuous activity patterns and their variations, but it can discover those patterns whose frequencies exhibit difference across different regions in home.

In the next chapter, we will provide an extension of our COM method. This improved method is able to mine data in the form of a continuous unbounded stream. As a result it addresses the problem of online activity mining.

CHAPTER 4. STREAM MINING

O, call back yesterday, bid time return.

— William Shakespeare, King Richard II.

In this chapter, we will show a method for discovering activity patterns from sensor data streams. Unlike our previous sequence mining methods, we no longer assume that an unlabeled dataset is available offline, rather we assume it appears in real time in the form of a potentially unbounded flow of sensor events. We introduce a novel stream mining method by extending our COM method which was discussed in Chapter 3. Our new method is called StreamCOM [Rashidi and Cook, 2010b] and uses a tilted time window approach [Giannella et al., 2003].

In the following sections, first we will provide an overview of our method in Section 4.1. Next, we will describe the tilted-time window in more detail in section 4.2. Our solution is explained in more details in section 4.3. We then show the results of our experiments on data obtained from two different smart apartments in section 4.4.

4.1 Introduction

In Chapter 3 we presented two sequence mining methods for activity discovery from sensor data. We also described a number of other unsupervised methods for activity discovery in Section 2.3. None of these mining approaches take into account the streaming nature of data, nor the possibility that the patterns might change over time. In a real world situation, in a smart environment we have to deal with a potentially infinite and unbounded flow of data. Also the discovered activity patterns can change over time. Mining the stream of data over time not only allows us to find new emerging patterns in the data, but it also allows us to detect changes in the patterns. Detecting changes in the patterns can be beneficial for many applications. For example a caregiver can look at the pattern trends over time and spot any suspicious changes immediately.

In the last decade, many stream mining methods have been proposed as a result of different emerging application domains, such as network traffic analysis, Web click stream mining, and power consumption measurements. Most of the proposed methods try to find frequent itemsets over data streams [Ren and Huo, 2008, Giannella et al., 2003, fu Li et al., 2004, Manku and Motwani, 2002]. Methods have been also proposed for finding frequent sequences over data streams [Chen et al., 2005a, Marascu and Masseglia, 2006, Raïssi et al., 2005]. In contrast, no stream mining method has been

proposed so far for mining human activity patterns from sensor data in the context of smart environments.

In this section, we present a novel stream mining method called StreamCOM for mining activity patterns from a stream. We extend the tilted-time window approach proposed by Giannella et al. [Giannella et al., 2003], in order to discover activity pattern sequences over time. The tilted-time window approach finds the frequent itemsets using a set of tilted-time windows, such that the frequency of the item is kept at a finer level for recent time frames and at a coarser level for older time frames. Such a tilted window approach can be quite useful for human activity pattern discovery. For example a caregiver is usually interested in the recent changes of the patient at a finer level, and in the older patterns (e.g. from three months ago) at a coarser level.

Due to the special requirements of our application domain, we cannot directly use the method proposed by Giannella et al. [Giannella et al., 2003]. First of all, the time-tilted approach [Giannella et al., 2003], as well as most of the other similar stream mining methods [Chen et al., 2005a, Marascu and Masseglia, 2006, Raïssi et al., 2005] were designed to find sequences or itemsets in transaction-based streams. The data obtained in smart environment is a continuous stream of unbounded sensor events with no boundary between episodes or activities. Second, as discussed in our DVSM and COM methods, the complex and erratic nature of human activity necessitates that we consider an activity pattern as a sequence of events. In such a sequence,

the patterns might be interrupted by irrelevant events (discontinuous patterns). The order of events in the sequence might also change from occurrence to occurrence (varied order patterns). Third, similar to our COM method, we also need to address the problem of varying frequencies for activities performed in different regions of the space.

In this chapter, we extend the COM method into a streaming version based on using a tilted-time window [Giannella et al., 2003]. Our proposed method allows us to find discontinuous varied-order patterns in streaming non transaction sensor data over time. Our approach represents the first reported stream mining method for discovering human activity patterns in sensor data over time. Besides activity mining, our StreamCOM method can be useful in other application domains, where different variations of a pattern can reveal useful information, such as Web click mining.

4.2 Tilted-Time Window Model

In this section, we explain the tilted window model [Giannella et al., 2003] in more detail. Figure 4.1 shows an example of a natural tilted-time window where the frequency of the most recent item is kept with an initial precision granularity of an hour (4 quarters), in another level of granularity in the last 24 hours and then again at another level in the last 31 days. As new data items arrive over time, the history

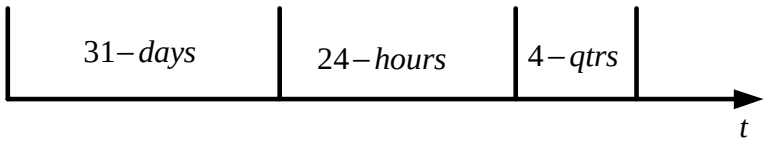


Figure 4.1: Natural tilted-time window.

of items will be shifted back in the tilted-time window to reflect the changes. Other variations such as logarithmic tilted-time windows have also been proposed to provide more efficient storage [Giannella et al., 2003].

The tilted-time window model uses a relaxed threshold to find patterns according to the following definition.

Definition 1. *Let the minimum support be denoted by σ, and the maximum support error be denoted by ϵ. An itemset I is said to be frequent if its support is no less than σ. If support of I is less than σ, but no less than $\sigma - \epsilon$, it is sub-frequent; otherwise it is considered to be infrequent.*

Using the approximation approach for frequencies allows for the sub-frequent patterns to become frequent later, while discarding infrequent patterns. To reduce the number of frequency records in the tilted-time windows, the old frequency records of an itemset I are pruned. Let $\bar{f}_j(I)$ denote the computed frequency of I in time unit j, and let N_j denote the number of transactions received within time unit j. Also let τ refer to the most recent time point. For some m where $1 \leq m \leq \tau$, the

frequency records $\bar{f}_1(I), ..., \bar{f}_m(I)$ are pruned if Equation 4.1 and Equation 4.2 hold [Cheng et al., 2008].

$$\exists n \leq \tau,\ \forall i, 1 \leq i \leq n,\ \bar{f}_i(I) < \sigma N_i \tag{4.1}$$

$$\forall l, 1 \leq l \leq m \leq n,\ \sum_{j=1}^{l} \bar{f}_j(I) < (\sigma - \epsilon) \sum_{j=1}^{l} N_i \tag{4.2}$$

Equation 4.1 finds a point n in the stream such that before that point, the computed frequency of the itemset I is always less than the minimum frequency required. Equation 4.2 finds a point m, where $1 \leq m \leq n$, such that before that point, the sum of the computed support of I is always less than the relaxed minimum support threshold. In this case the frequency records of I from 1 to m are considered as unpromising and are pruned. This type of pruning is referred to as "tail pruning". In our model, we will extend the above defections and pruning techniques for discontinuous, varied order patterns.

4.3 StreamCOM Description

In the following subsections, first we give an overview of definitions and notations, then we will describe our model in more detail.

4.3.1 Definitions

The input data in our model is an unbounded stream of sensor events, each in the form of $e = \langle s, t \rangle$, where s refers to a sensor ID and t refers to the timestamp when sensor s has been activated. We define an activity instance as a sequence of n sensor events $\langle e_1, e_2, .., e_n \rangle$. Note that in out notations an activity instance is considered as a sequence of sensor events, not a set of unordered events.

We assume that the input data is broken into batches $B_{a_1}^{b_1}...B_{a_n}^{b_n}$ where each $B_{a_i}^{b_i}$ is associated with a time period $[a_i..b_i]$, and $a_i < b_i$ and $a_1 < a_n$. The most recent batch is denoted by $B_{a_\tau}^{b_\tau}$ or for short as B_τ. Each batch $B_{a_i}^{b_i}$ contains a sequence of sensor events, whose length is denoted by $|B_{a_i}^{b_i}|$.

As we mentioned before, we use a tilted-time window for maintaining the history of patterns over time. Instead of maintaining frequency records, we maintain "compression" records, which will be explained in more detail in the following sections. Because the frequency of an item is not the single deciding factor, and other factors such as the length of the pattern and its continuity also play a role, we will use the term "interesting" pattern instead of a "frequent" pattern.

The tilted-time window used in our model is depicted in Figure 4.2. This tilted-time window keeps historical records of a pattern during the past 4 weeks at a finer level of week granularity. Records older than 4 weeks are only kept at a

month granularity. Considering our application domain and its end users, a tilted-time window provides a more natural representation than a logarithmic tilted-time window. For example, it would be it easier for a nurse or caregiver to interpret the pattern trend using such a natural representation. Second, as we do not expect activity patterns to change substantially over a very short period of time, we omit the day and hour information for the sake of a more efficient representation. For example, in the case of monitoring dementia patients it takes weeks and months to see some changes to develop in their daily activity patterns. Using such a schema we only need to maintain 15 compression records (11 months + 4 weeks), instead of $365 * 24 * 4$ records in a normal natural tilted-time window keeping day and hour information. Note that we chose such a representation in this study for the reasons mentioned above. However, one can adopt other tilted-window models such as the logarithmic windows, as the choice of tilted-time window has no effect on the model except for efficiency.

Figure 4.2: Our tilted-time window.

To update the tilted-time window, whenever a new batch of data arrives, we will replace the compression values at the finest level of time granularity and shift back to

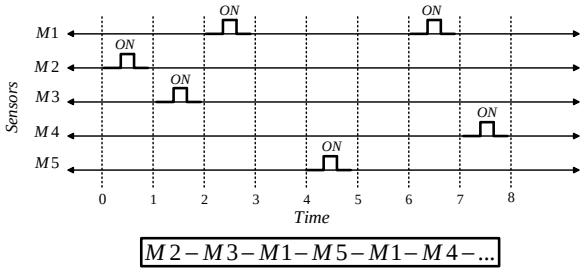
the next level of finest time granularity. During shifting, we check if the intermediate window is full. If so, the window is shifted back even more; otherwise the shifting stops.

Note that all of the previous tilted-time approaches consider data to appear in a transactional format. However, input data stream in a smart environment is a continuous flow of unbounded data. Figure 4.3 depicts the difference between transaction data and sensor data. As can be seen in Figure 4.3a, for transaction data, each single transaction is associated with a set of items and is identified by a transaction ID, making it clearly separated from the next transaction. The sensor data has no boundaries separating different activities or episodes from each other, and it is just a continuous stream of sensor events over time.

Approaches proposed by sensor stream mining community [Papadimitriou et al., 2003, Loo et al., 2005] try to turn a sensor stream into a transactional dataset using techniques such as the Apriori technique [Agrawal and Srikant, 1995] to group frequent events together. Another method is to simply use fixed or varied clock ticks [Loo et al., 2005]. In our scenario, using such simple techniques does not allow us to deal with complex activity patterns that can be discontinuous, varied order, and of arbitrary length. To deal with this problem, we extend the DVSM method to group together co-occurring events into varied-order discontinuous activity patterns.

Transaction ID	Items
1	$\{A, B, D, F, G, H\}$
2	$\{D, F, G, H, X\}$
3	$\{A, B, C, X, Y\}$

(a)



(b)

Figure 4.3: Transaction data vs. sensor data.

4.3.2 Mining Activity Patterns

Our goal is to develop a method that can automatically discover resident activity patterns over time from streaming sensor data, even if the patterns are discontinuous or have different event orders across their instances. We discover the sequential patterns from the current data batch B_τ by using an extended version of COM that is able to find patterns in streaming data. After finding patterns in the current data batch B_τ, we will update the tilted-time windows, and will prune any pattern that is

unpromising.

To find patterns in data, first a reduced batch B_τ^r is created from the current data batch B_τ. The reduced batch contains only frequent and subfrequent sensor events, which will be used for constructing longer patterns. A minimum support is required to identify such frequent and subfrequent events. COM only identifies the frequent events, but we introduce the maximum sensor support error ϵ_s to allow for the subfrequent patterns to be also discovered. We will also automatically derive multiple minimum supports values corresponding to different regions of the space.

As mentioned in Section 3.3.1, in mining real life activity patterns, the frequency of sensor events can vary across different regions of the home or other space. This situation also applies to streaming data. Here, by extending COM into a streaming method we propose a solution for the problem of rare items. Our proposed solution for solving the problem of rare items in data streams can be applied to other application domains, such as Web click mining. For example, a web page might have a lower chance of being visited by visitors, but we still might be interested in finding click patterns in such pages.

We will automatically derive multiple minimum sensor support values across space and over time. To do this, we identify different regions of the space using location tags l, corresponding to the functional areas such as bedroom, bathroom, etc. Different sensor types might also exhibit varying frequencies. In our experiments, we

again categorize the sensors into two classes: motion sensors and key, or interaction-based sensors.

For the current data batch B_τ, we compute the minimum regional support for different categories of sensors as in Equation 4.3. Here l refers to a specific location, c refers to the sensor's category, and $\mathcal{S}_c$ refers to the set of sensors in a category c. The symbol $f_T(s)$ refers to the frequency of a sensor s over a time period T.

$$\sigma_T^c(l) = 1/\sum_{s \in \mathcal{S}_c^l}^{|\mathcal{S}_c^l|} f_T(s) \quad s.t. \quad \mathcal{S}_c^l = \{s \mid s \in l \ \wedge \ s \in \mathcal{S}_c\} \tag{4.3}$$

Using the minimum regional sensor frequencies, frequent and subfrequent sensors are defined as following.

Definition 2. *Let s be a sensor of category c located in location l. The frequency of s over a time period T, denoted by $f_T(s)$, is the number of times in time period T in which s occurs. The support of s in location l and over time period T is $f_T(S)$ divided by the total number of sensor events of the same category occurring in L during T. Let ϵ_s be the maximum sensor support error. Sensor s is said to be frequent if its support is no less than $\sigma_T^c(l)$. It is sub-frequent if its support is less than $\sigma_T^c(l)$, but no less than $\sigma_T^c(l) - \epsilon_s$; otherwise it is infrequent.*

Only the sensor events from the frequent and subfrequent sensors will be added to the reduced batch B_τ^r, which is then used for constructing longer sequences. We use a pattern growth method as in [Rashidi and Cook, 2009b] which grows a pattern

by its prefix and suffix. To account for the variations in the patterns, the concept of a general pattern similar to DVSM and COM is introduced. During pattern growth, if an already discovered variation matches a newly discovered of pattern, its frequency and continuity information will be updated. If the newly discovered pattern matches the general pattern, but does not exactly match any of the variations, it is added as a new variation. Otherwise it will be considered as a new general pattern.

At the end of each pattern growth iteration, infrequent or highly discontinuous patterns and variations will be discarded as uninteresting patterns. Instead of solely using a pattern's frequency as a measure of interest, we use a compression objective based on the minimum description length (MDL) [Rissanen, 1978]. The compression value of a general pattern a over a time period T is defined as in Equation 4.4. The compression value of a variation a_i of a general pattern over a time period T is defined as in Equation 4.5. Here DL refers to the MDL description length, and Γ refers to continuity as defined for DVSM and COM methods.

$$\alpha_T(a) = \frac{DL(B_T) * \Gamma_a}{DL(a) + DL(B_T|a)} \tag{4.4}$$

$$\beta_T(a_i) = \frac{(DL(B_T|a) + DL(a)) * \Gamma_{a_i}}{DL(B_T|a_i) + DL(a_i)} \tag{4.5}$$

Variation compression measures the capability of a variation to compress a general pattern compared to the other variations. Compression of a general pattern

shows the overall capability of the general pattern to compress the dataset with respect to its length and continuity. Based on using the compression values and by using a maximum compression error, we define interesting, sub-interesting and uninteresting patterns and variations.

Definition 3. *Let the compression of a general pattern a be defined as in Equation 4.4 over a time period T. Also Let σ_g and ϵ_g denote the minimum compression and maximum compression error. The general pattern a is said to be interesting if its compression α is no less than σ_g. It is sub-interesting if its compression is less than σ_g, but no less than $\sigma_g - \epsilon_g$; otherwise it is uninteresting.*

We also give a similar definition for identifying interesting/sub-interesting variations of a pattern. Let the average variation compression of all variations of a general pattern a over a time period T be defined as in Equation 4.6. Here the number of variations of a general pattern is denoted by n_a.

$$\tilde{\beta}_T(a) = \frac{1}{n_a} * \sum_{i=1}^{n_a} \frac{(DL(B_T|a) + DL(a)) * \Gamma_{a_i}}{DL(B_T|a_i) + DL(a_i)} \tag{4.6}$$

Definition 4. *Let the compression of a variation a_i of a general pattern a be defined as in Equation 4.5 over a time period T. Also Let ϵ_v denote the maximum variation compression error. A variation a_i is said to be interesting over a time period T if its compression $\beta_T(a_i)$ is no less than $\tilde{\beta}(a)_T$. It is sub-interesting if its compression is less than $\tilde{\beta}(a)_T$, but no less than $\tilde{\beta}(a)_T - \epsilon_v$; otherwise it is uninteresting.*

During each pattern growth iteration, based on the above definitions, the uninteresting patterns and variations are pruned, specifically those patterns and variations that are either highly discontinuous or infrequent (with respect to their length). We also prune redundant non-maximal general patterns, specifically those patterns that are completely contained in another larger pattern. To maintain only the very relevant variations of a pattern, variations of a pattern that are determined to be irrelevant based on mutual information [Guyon and Elisseeff, 2003] are discarded.

We continue extending the patterns by prefix and suffix at each iteration until no more interesting patterns are found. We refer to the pruning process performed during the pattern growth on the current data batch as normal pruning. Note that this is different from the tail pruning process which is performed on the tilted-time window to discard the unpromising patterns over time. In the following discussion we will describe how the tilted-time window is updated after discovering patterns in the current data batch.

4.3.3 Updating the Tilted-time Window

After discovering the patterns in the current data batch as described in the previous subsection, the tilted-time window will be updated. Each general pattern is associated with a tilted-time window. The tilted-time window keeps track of the

general pattern's history as well as its variations. Whenever a new batch arrives, after discovering its interesting general patterns, we will replace the compressions at the finest level of granularity with the recently computed compressions. If a variation of a general pattern is not observed in the current batch, we will set its recent compression to 0. If none of the variations of a general patterns are perceived in the current batch, then the general pattern's recent compression is also set to 0.

In order to reduce the number of maintained records and to remove unpromising general patterns, we propose the following tail pruning mechanisms as an extension of the original tail pruning method in [Giannella et al., 2003]. Let $\alpha_j(a)$ denote the computed compression of general pattern a in time unit j. Also let τ refer to the most recent time point. For some m, where $1 \leq m \leq \tau$, the compression records $\alpha_1(a), ..., \alpha_m(a)$ are pruned if Equations 4.7 and 4.8 hold.

$$\exists n \leq \tau,\ \forall i, 1 \leq i \leq n,\ \alpha_i(a) < \sigma_g \tag{4.7}$$

$$\forall l, 1 \leq l \leq m \leq n,\ \sum_{j=1}^{l} \alpha_j(a) < l * (\sigma_g - \epsilon_g) \tag{4.8}$$

Equation 4.7 finds a point n in the stream such that before that point, the computed compression of the general pattern a is always less than the minimum compression required. Equation 4.8 computes the time unit m, where $1 \leq m \leq n$, such that before that point, the sum of the computed compression of a is always

less than the relaxed minimum compression threshold. In this case the compression records of a from 1 to m are considered as unpromising and are pruned.

We define a similar procedure for pruning the variations of a general pattern. We prune a variation a_k if the following conditions in Equations 4.9 and 4.10 hold.

$$\exists n \leq \tau,\ \forall i, 1 \leq i \leq n,\ \beta_i(a_k) < \tilde{\beta}_i(a) \tag{4.9}$$

$$\forall l, 1 \leq l \leq m \leq n,\ \sum_{j=1}^{l} \beta_j(a_k) < l * (\tilde{\beta}_l(a) - \epsilon_v) \tag{4.10}$$

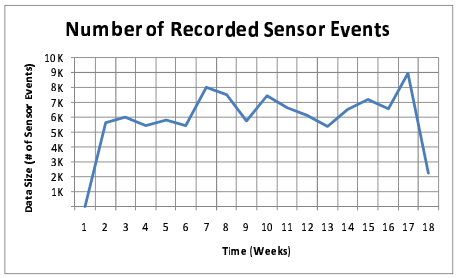
Equation 4.9 finds a point in time where the computed compression of a variation is less than the average computed compression of all variations in that time unit. Equation 4.10 computes the time unit m, where $1 \leq m \leq n$, such that before that point the sum of the computed compression of a_i is always less than the relaxed minimum support threshold. In this case the compression records of a_i from 1 to m are considered as unpromising and are pruned.

4.4 EXPERIMENTS

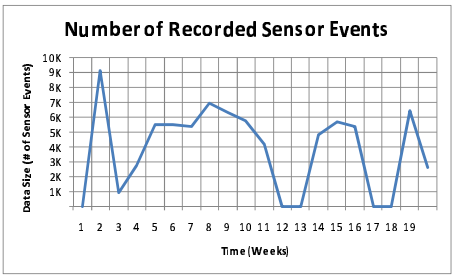
The performance of the system was evaluated on the data collected from two smart apartments. The layout of the apartments including sensor placement and location tags are similar to those for COM experiments in Section 3.3.2 and in Table

3.1. We will refer to those apartments as apartments 1 and apartment 2. The apartments were equipped with infrared motion sensors installed on ceilings, infrared area sensors installed on walls, and switch contact sensors to detect open/close status of the doors and cabinets. The data was collected during 17 weeks for apartment 1, and during 19 weeks for apartment 2. During data collection, the resident in apartment 2 was away for approximately 20 days, once during week 12 and once during week 17. We note that the last week of data collection in both apartments does not include a full cycle. In our experiments, we constrain each time batch to contain approximately one week of data. In our experiments, we set the maximum errors ϵ_s, ϵ_g and ϵ_v to 0.1, as suggested in the literature. The value of σ_g was set to 0.75 based on several runs of experiments.

To be able to evaluate the results of our algorithms based on ground truth, each one of the datasets was annotated with activities of interest. A total of 10 activities were noted for each apartment. Those activities included bathing, bed-toilet transition, eating, leave home/enter home, housekeeping, meal preparation (cooking), personal hygiene, sleeping in bed, sleeping not in bed (relaxing) and taking medicine. Apartment 1 includes $193,592$ sensor events and $3,384$ annotated activity instances. Apartment 2 includes $132,550$ sensor events and $2,602$ annotated activity instances. Figures 4.4a and 4.4b show the number of recorded sensor events over time. As we mentioned, the resident in apartment 2 was not at home during two different time



(a) Apartment 1.

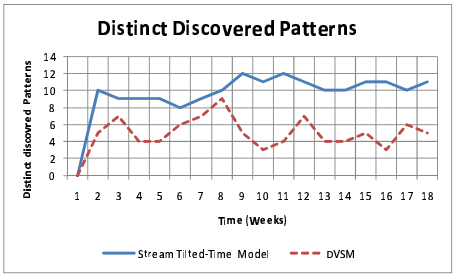


(b) Apartment 2.

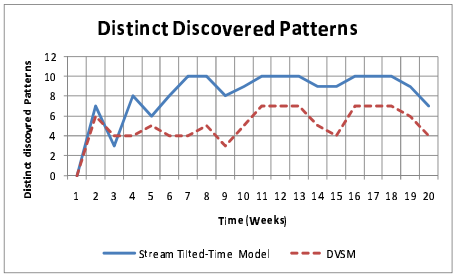
Figure 4.4: Total number of recorded sensor events over time (time unit = weeks).

periods, hence we see the gaps in Figure 4.4b where the number of sensor events is 0 for a period of time.

We ran our algorithms on both apartments' datasets. Figures 4.5a and 4.5b show the number of distinct patterns that are discovered over time based on using a global support employed by DVSM versus using multiple regional support as in our proposed method. The results again confirm our hypothesis that a COM-based method is able to detect a higher percentage of interesting patterns using multiple regional support values. Some of the patterns that have not been discovered are indeed quite difficult to spot and also in some cases are less frequent. For example, the housekeeping activity happens every 2-4 weeks and is not associated with any specific sensor. Some of the similar patterns are merged together, as they use the same set of sensors, such as eating and relaxing activities. It should be noted that some of the activities are discovered multiple times in form of different patterns, as the



(a) Apartment 1 (time unit = weeks).



(b) Apartment 2 (time unit = weeks).

Figure 4.5: Total number of distinct discovered patterns over time.

activity might be performed in a different motion trajectory using different sensors. One also can see that the number of discovered patterns increases at the beginning and then fluctuates over time depending on the perceived patterns in the data. The number of discovered patterns depends on perceived patterns in current data batch and previous batches, as well as the compression of patterns in tilted-time window records. Therefore, some of the patterns might disappear and reappear over time which can be a measure of how consistently the resident performs those activities.

As already mentioned, to reduce the number of discovered patterns over time, our algorithm performs two types of pruning. The first type of pruning, called normal pruning, prunes patterns and variations while processing the current data batch. The second type of pruning is based on tail pruning to discard unpromising patterns and variations stored in tilted-time window. Figures 4.6a and 4.6b show the results of both types of pruning on the first dataset. Figures 4.6d and 4.6e show the results

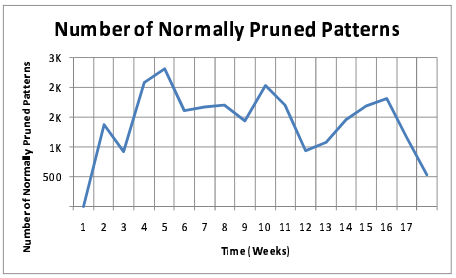
of both types of pruning on the second dataset. Figures 4.6c and 4.6f show the tail pruning results in tilted-time window over time. Note that the gaps for apartment 2 results are due to the 20 days when resident was away.

By comparing the results of normal pruning in Figures 4.6a and 4.6d against the number of recorded sensors in Figures 4.4a and 4.4b, one can see that the normal pruning follows the pattern of recorded sensors. If more sensor events are available, more patterns would be obtained, and also more patterns would be pruned. For the tail pruning results, depicted in Figures 4.6b, 4.6e, 4.6c and 4.6f, the number of tail pruned patterns at first increases in order to discard the many unpromising patterns at the beginning. The trend lines in Figure 4.6b and 4.6e also confirm this. The number of tail pruned patterns then decreases over time as the algorithm stabilizes.

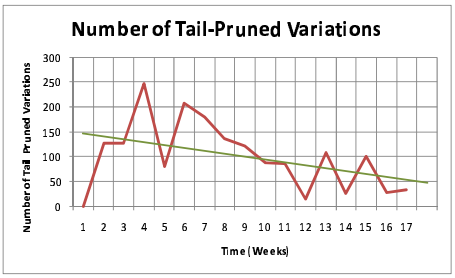
To see how consistent the variations of a certain general pattern are, we used a measure called "variation consistency" based on using the externally provided labels. We define the variation consistency as in Equation 4.11. Here $|v_{11}|$ refers to the number of variations that have the same label as their general pattern, and $|v_{01}|$ and $|v_{10}|$ refer to the number of variations that have a different label other than their general pattern's label.

$$purity(P_i) = \frac{|v_{11}|}{|v_{11}| + |v_{01}| + |v_{10}|} \tag{4.11}$$

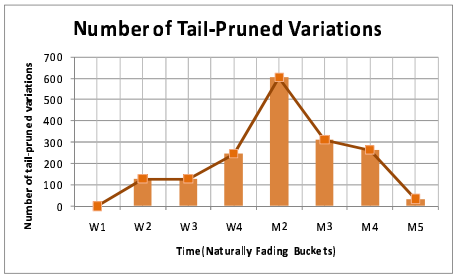
Figures 4.7a and 4.7b show the average variation consistency for apartments



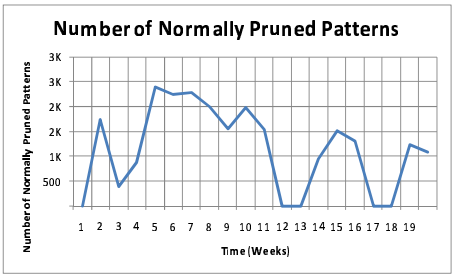
(a) Apartment 1 (time unit = weeks).



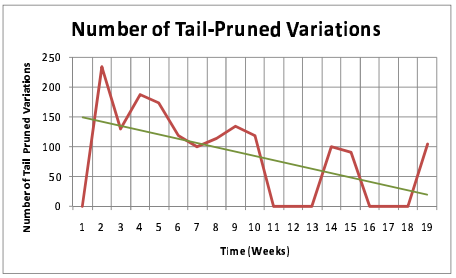
(b) Apartment 1 (time unit = weeks).



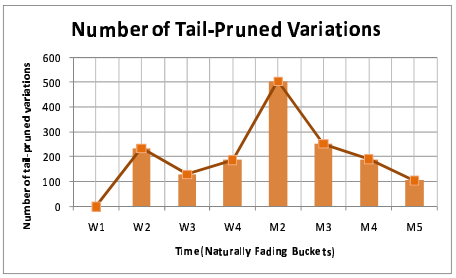
(c) Apartment 1 (time unit = tilted-time frame)



(d) Apartment 1 (time unit = weeks).



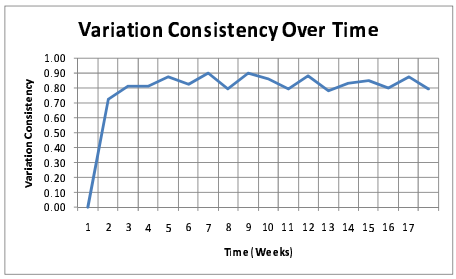
(e) Apartment 2 (time unit = weeks).



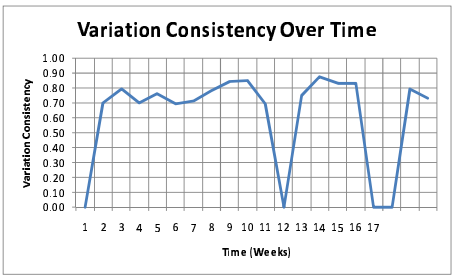
(f) Apartment 2 (time unit = tilted-time frame)

Figure 4.6: Total number of tail-pruned variations over time. For the bar charts, W1-W4 refers to the week 1-4, and M1-M5 refers to month 1-5.

1 and 2. As mentioned, for each current batch of data the irrelevant variations are discarded using mutual information. The result confirms that that the variation consistency increases at the beginning, and then it quickly stabilizes due to discarding irrelevant variations for each batch.



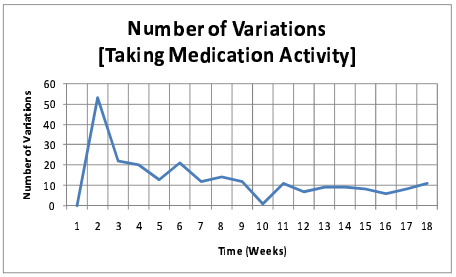
(a) Apartment 1 (time unit = tilted-time frame).



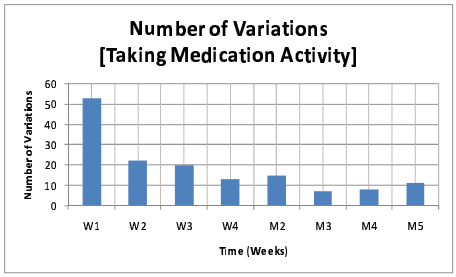
(b) Apartment 2 (time unit = weeks).

Figure 4.7: Total number of distinct discovered patterns and their variation consistency over time.

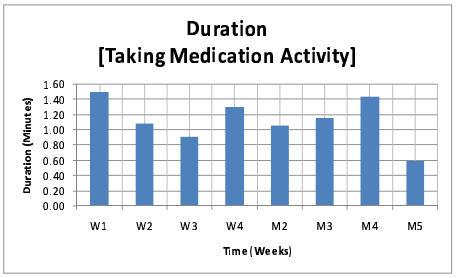
To highlight the changes of a specific pattern over time, we show the results of our algorithm for "taking medication" activity over time. Figure 4.8a graphs the number of discovered variations over time for "taking medication" activity. Figure 4.8b graphs the same results in the tilted-time window over time. We can clearly see that the number of discovered variations quickly drops due to the tail pruning process. This shows that despite the fact that we are maintaining records over time for all variations, many of the uninteresting, unpromising and irrelevant variations will be pruned, making our algorithm more efficient in practice.



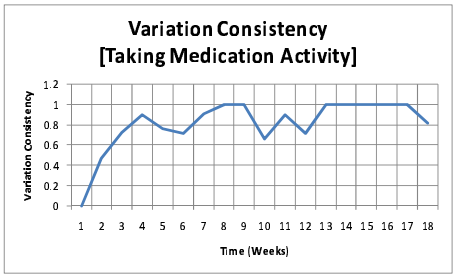
(a) (Number of discovered variations (time unit = weeks).



(b) Number of discovered variations (time unit = tilted-time frame).



(c) Duration (time unit = weeks).



(d) Variation consistency (time unit = weeks).

Figure 4.8: Number of discovered variations, duration and consistency for "taking medication" activity pattern over time.

We also show how the average duration of the "taking medication" pattern changes over time in Figure 4.8c. Presenting such information can be informative to caregivers because it allows them to detect any anomalous events in the patterns. Figure 4.8d shows the consistency of the "taking medication" variations over time. Similar to the results obtained for the average variation consistency of all patterns, we see that the variation consistency is increased and then stabilized quickly.

In summary, the results of our experiments confirm that we can find sequential patterns from a steam of sensor data over time. It also shows that using two types of pruning techniques allows for a large number of unpromising, uninteresting and irrelevant patterns and variation to be discarded, in order to achieve a more efficient solution that can be used in practice.

4.5 Summary

In this chapter we showed a method for discovering sequential patterns over time from a stream of sensor data. Our method is based on the same concepts as COM and DVSM. We provided an extension of the tilted-time window model for continuity-based, varied order sequential patterns according to the special requirements of our application domain. Not only can our proposed method be used in an activity discovery and recognition system, but it also can be applied to other application domains, such as Web click mining.

This chapter as well as the previous chapter have used unsupervised methods in order to deal with the annotation problem in smart environments. In the next chapter, we introduce yet another method to recognize activities when few or no labeled activity data is available, based on transfer learning ideas.

CHAPTER 5. TRANSFER LEARNING

You don't understand anything until you learn it more than one way.

— Marvin Minsky

In this chapter, we introduce several novel activity recognition algorithms based on ideas from transfer learning. Transfer learning allows us to reuse activity knowledge from previous domains and apply to a new domain.

First we show a method called **M**ulti **R**esident **T**ransfer **L**earning, or for short MRTL. MRTL can transfer activity knowledge between residents in the same physical space. Our second method is called **H**ome to **H**ome **T**ransfer **L**earning, or for short HHTL. HHTL can transfer activity knowledge from a single physical space to another target physical space. Our third method is called **M**ulti **H**ome **T**ransfer **L**earning, or for short MHTL. MHTL can transfer activity knowledge from multiple physical spaces to a target physical space. Finally we present a method for selecting among a number of available sources, in order to select the best sources for transferring.

5.1 MRTL

In this section we introduce the **M**ulti **R**esident **T**ransfer **L**earning method, or for short MRTL. MRTL is an unsupervised method to recognize and transfer learned activities across different residents. We use DVSM to discover interesting patterns in data in order to capture the intra-subject variability. Then we employ an activity mapping method to map activities from a source resident to a target resident to address the inter-subject variability and to provide a degree of similarity between two sets of activities.

We consider each state of the activity to have arbitrary attributes, such as duration or frequency which contribute to finding an appropriate mapping. We do not require target activities to have the same structure as the source activities, such as an equal number of states. Considering such general aspects, we define a flexible method to map and measure similarity between source and target activities. In addition to reducing the required amount of data, finding a mapping between activities allows us to exploit gained knowledge in the source context. For example, if we have discovered that certain types of cue detail and cue timing work best for a specific source activity this information can be used for the mapped target activity too, depending on the similarity degree.

In following sections, we first explain the general model, and then we describe

each of the components in more detail. Finally we present the results of our experiments.

5.1.1 Model

In our model, we hypothesize that the data collected for one resident in the source space Φ_s can be used to recognize activities for that resident, and more importantly can be transferred to a target space Φ_t to learn activities for a different resident. The obtained mapping and similarity measure will also allow us to eventually transfer the source activity's related knowledge, including cue timings and context. In our current work, however, we initially constrain the problem to consider transferring knowledge between different residents. We will also show methods for transferring knowledge between different physical spaces, or between spaces with different sets of sensors. In our model, the input consists of activities $\mathcal{A}$ from source Φ_s (which we assume have already been discovered using DVSM), and a small dataset $\mathcal{D}$ from target Φ_t. Using DVSM, we first identify interesting patterns $\mathcal{P}$ as sensor sequences that usually appear together in $\mathcal{D}$. Next, we cluster $\mathcal{P}$ into $|\mathcal{A}|$ clusters where $|\mathcal{A}|$ is the number of activities in $\mathcal{A}$. By comparing each cluster k's representative, c_k, to each activity in $\mathcal{A}$, and finding a similarity measure between them, we are able to generate a mapping from Φ_s to Φ_t (see Figure. 5.1). Note that a source activity can

be mapped to another target activity even if the target contains more or fewer states than the source, and even if the target has different state attributes. In addition, the order of states does not need to be preserved by the mapping.

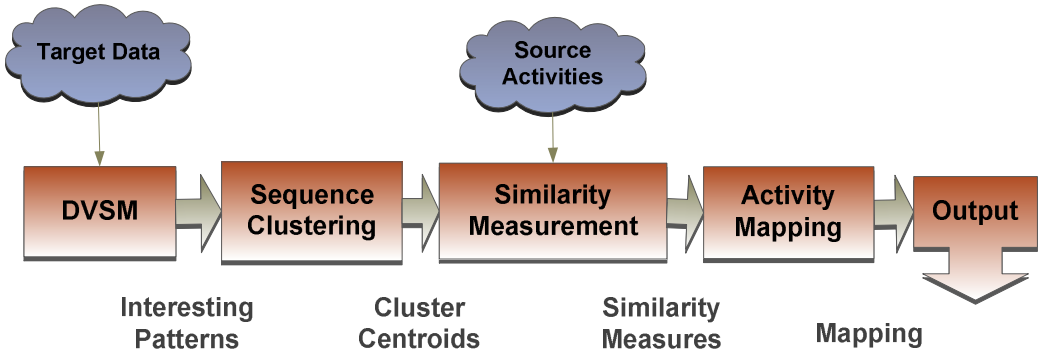


Figure 5.1: MRTL architecture.

Next, we will describe our model in more detail.

Activity Mapping Method

To transfer activity pattern knowledge, the first step is to discover all the activity patterns $\mathcal{P}$ in the sensor dataset $\mathcal{D}$ using our DVSM method. Next, we will cluster the discovered patterns $\mathcal{P}$ into a set of clusters $\mathcal{A}$, the size of which is equal to the number of activities in Φ_s. Clustering $\mathcal{P}$ into $|\mathcal{A}|$ clusters allows us to better identify an accurate 1-1 mapping between source and target activities, as we will compare each cluster's representative c with each activity $a \in \mathcal{A}$, instead of comparing every $p \in \mathcal{P}$ with each $a \in \mathcal{A}$. The clustering method we are using is a standard k-means clustering [MacQueen, 1967], however we need a method for defining representatives

and comparing activities in order to form clusters.

Two methods for comparing similarity of sequences are "edit distance" [Levenshtein, 1966] and "LCS" [Sequeira and Zaki, 2002] for simple sequences of symbols. Saneifar et al. [Saneifar et al., 2008] have proposed a similarity measure for more complex itemset sequences based on the number of common items. Those methods do satisfy our complex definition of an activity as a sequence of events with arbitrary state attributes, and do not deal with general aspects of sequences such as temporal information, order of states, etc. In our model it is possible to map a combination of two states to one state. This can happen, for example, if the state pair's total duration in the source context is close to the duration of a single state in the target context.

To calculate the similarity between two activities a and b, we need to determine similarity between their set of states $\mathcal{S}_a$ and $\mathcal{S}_b$, in addition to the order similarity. Using our algorithm it is possible to combine several states during mapping, cases where one state immediately follows another state with a sensor of the same type that provides the same functionality. To find possible combinations of states, we define $Ext(\mathcal{S})$, the extension of a state set $\mathcal{S}$, by considering possible combinations between consecutive states with sensors of the same type to form a new state. For example, for three consecutive sensors of the same type (a, b, c), it's possible to consider three new extended states, $a' = \{a, b\}$, $b' = \{b, c\}$ and $c' = \{a, b, c\}$. Note that additive

attributes such as duration will be summed. We denote the union of actual and extended states for an activity a as $\Pi_a = \{\mathcal{S}_a \cup Ext(\mathcal{S}_a)\}$.

We also need to define order similarity, to see if activities a and b have the same relative order. We define the order similarity, $s_o(i,j)$, between state $i \in \mathcal{S}_a$ and state $j \in \mathcal{S}_b$ as in Eq. 5.1 where $pos(s)$ shows position of state s in its corresponding activity (for extended states, the total number of states will not be equal to $|\mathcal{S}|$, therefore it is replaced by corresponding new size).

$$s_o(i,j) = 1 - |\frac{pos(i)}{|\mathcal{S}_a|} - \frac{pos(j)}{|\mathcal{S}_b|}| \tag{5.1}$$

To measure similarity between two activities a and b, we need to find the best possible mappings between their states. We start with an initial mapping and then resolve any resulting conflicts. This can happen, for example, if two separate actual or extended states are mapped to the same state. For each state $i \in \Pi_a$, we find the best possible mapping state $j \in \Pi_b$. The state similarity, $s_s(i,j)$, between two states i and j is defined as in Eq. 5.2. Here, attribute k found in both states i and j is denoted by k_i and k_j; w_k is a weight applied to attribute k to indicate the importance of k in calculating similarity (e.g. we might consider duration more important than frequency); and m denotes the total number of attributes. In our model, we only map sensors of the same type (e.g. motion sensors to motion sensors).

$$s_s(i,j) = 1 - (\sum_{k=1}^{m} w_k * (\frac{k_i}{max(k_i)} - \frac{k_j}{max(k_j)}) * s_o(i,j)) \tag{5.2}$$

Then, the best possible mapping for state $i \in \Pi_a$ is defined as in Eq. 5.3.

$$map(i) = argmax_{j \in \Pi_b}\{s_s(i,j)\} \tag{5.3}$$

The cumulative similarity, $s_c(a,b)$, between a and b is defined as in Eq. 5.4. Note that the subset of states selected from Π_a and Π_b for mapping will be denoted by Υ_a and Υ_b.

$$s_c(a,b) = \frac{1}{\Upsilon_a} \sum_{i=1}^{|\Upsilon_a|} s_s(i, map(i)) \tag{5.4}$$

To resolve any conflicts between a subset of states $\mathcal{S}_c \subset \Pi_a$ that map to a single state c, we will find a new mapping for each of the states $s \in \mathcal{S}_c$. The new mappings will be selected from $\Pi_b - \{c \cup \Upsilon_b\}$. The state with the least new similarity will be mapped to c, as such an assignment will cause the least amount of decrease in the cumulative similarity. The rest of the states will be assigned according to new mappings. If still there is any conflict, it will be resolved in an iterative manner. If no mappings can be found for a state, it will be mapped to a null state.

The cluster representative c_k for cluster k is defined as following: the number of states for c_k will be equal to the average number of states in k, the attributes for each state will equal the average values for that state in k, and each state's sensor type

will be the most common type sensor type in k for that state. After the clustering is finished, we will compare each $a \in \mathcal{A}$ to each c_k, and by finding $argmax_k\{s_c(c_k, a)\}$, we can map activities in Φ_s and Φ_t. This mapping and similarity measure can act as a guideline for transferring related activity knowledge.

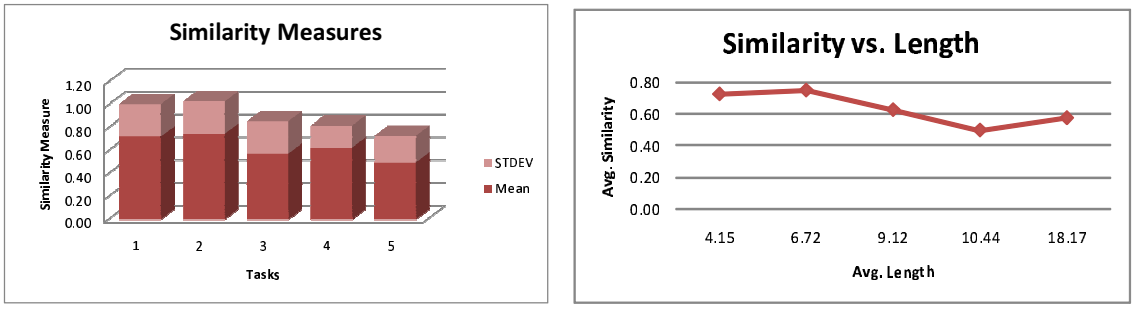
5.1.2 MRTL Experiments

The testbed is a 3-bedroom smart apartment, located on Washington State University campus. It is the same testbed as in experiments performed in Section 3.2.3 and in Figure 3.5. Sensor data was collected from 59 sensors including motion, temperature, water, burner, phone usage, and the presence of key items.

We brought 23 participants into the smart apartment, one at a time. Each participant was asked to perform a script of five iADLs, typically found in clinical questionnaires assessing everyday functional activities; including (1) telephone usage, (2) hand washing, (3) meal preparation, (4) medication use, and (5) household cleaning. Our data sets consisted of sensor event data for the series of 5 ADLs, each repeated for about 3 times with random events injected between activities up to 50%.

Using a 10-fold cross validation approach, we assessed the ability of our algorithm to accurately recognize activities for new target participants based on models learned from a source participant.

Our model was able to map and measure similarity between activities correctly, despite the fact that the activities were performed in vastly different ways, including different sensor event orders, different durations and different activity lengths (see Figure. 5.2). It was interesting to note that longer patterns usually had a lower similarity measure. This might be explained as there are more possibilities for differences when more symbols are involved.



(a) Similarity measures for all 5 tasks.

(b) Similarity vs. Length

Figure 5.2: Similarity measure results.

In another experiment, we randomly changed the order of events in two sets of datasets similar to the above datasets to simulate the effect of misplaced steps, one dataset containing shorter patterns of 1 to 14 events, and the other containing longer patterns of 14 to 47 events. Though for longer patterns the misplacement can generate much more diverse patterns and therefore make it more difficult to detect patterns, still our algorithm was able to find patterns (see Fig. 5.3).

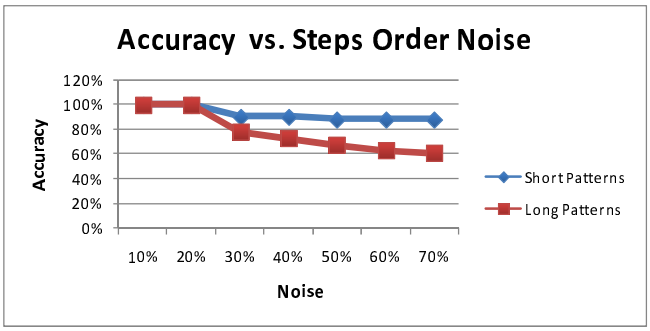


Figure 5.3: Accuracy vs. injected order noise for short and long patterns.

The above results confirm our hypothesis that our method is able to map activities despite inter-subject variability. The small datasets used for those experiments (each dataset containing an activity repeated three times with 50% random activities in between) also show how our method can use knowledge from the source space to effectively recognize activities in the target space.

5.2 HHTL and MHTL

In this section, we explain our HHTL (**H**ome to **H**ome **T**ransfer **L**earning) and MHTL (**M**ulti to **H**ome **T**ransfer **L**earning) methods. As HHTL is a specific case of MHTL by using only a single source, we do not describe the method separately. However, we will compare the two methods in our experiments.

5.2.1 Introduction

Traditionally, in smart environments each environmental situation is treated as a separate context in which to perform learning. What can propel research in cyber-physical systems forward is the ability to leverage experience of previous environments in new different environments. However, current activity recognition approaches do not exploit the knowledge learned in previous spaces in order to kick start activity recognition in a new space. This results in a delayed installation period in practice due to the need for collecting and annotating huge amounts of data for each new space. It also leads to redundant computational effort and excessive time investment, and results in ignoring insights gained from previous spaces.

Using conventional unsupervised methods such as frequent or periodic data mining methods, the long data collection period and prolonged installation process becomes a problem in practice. Using supervised methods, a greater burden is placed on the user of the smart environment, who must annotate sufficient data in order to train the recognition algorithms. Hand labeling from raw sensor data is very time consuming. Data collected in our testbeds required at least one hour of an expert's time to annotate a single day's worth of sensor data. This becomes particularly problematic if we are targeting an installation in the home of an older adult who may not be able to accurately annotate a large amount of data. Learning the model of

each environment separately and ignoring what has been learned in other physical settings also leads to redundant computational effort, excessive time investment, and loss of beneficial information that can improve the recognition accuracy. Therefore, it is beneficial to develop models that can exploit the knowledge of learned activities by employing them in new spaces. Exploiting the transferred knowledge results in reducing the need for data collection, reducing or eliminating the need for data annotation, and accelerating the learning pace. Using multiple sources and fusing their data together can leverage the learning process even more by using a more diverse set of activity models that can help in discovering and recognizing the target activities.

In this section, we introduce a method for transferring the knowledge of learned activities from multiple physical smart environments to a new target physical smart environment. In our approach, the layout of the spaces and the residents' schedules can be different. In addition, the type and number of sensors in two spaces might be different. We validate our algorithms using data collected from six different smart apartments with different layouts and different residents.

Compared to some previous preliminary work in this area such as by Kasteren et al. [van Kasteren et al., 2008], in our approach, the activity model includes much more information based on using structural, temporal and spatial features of the activities. Unlike their approach, we do not manually map the sensor networks. Instead, we learn sensor mappings based on the available data and activity models. In order to exploit

the knowledge learned in different spaces, we transfer the activities from multiple physical source spaces to a target physical space. First, we use a location based data mining method to find target activities in the target data. Then the activities from both source and target spaces are represented in a canonical form called an "activity template" in order to allow for a more efficient mapping process. Next, we use a semi-EM (Expectation Maximization) framework [Dempster et al., 1977] to map source activities from each single source to the target activities. Finally, by using an ensemble learning method based on a weighted majority vote [Dietterich, 2000] and fusing multiple data sources we assign activity labels to the target activities.

5.2.2 Model Description

Our objective is to develop a method for transferring knowledge in the form of activity models learned in multiple source physical spaces to a target physical space in order to reduce data collection time, reduce or eliminate data annotation time and exploit prior source knowledge in a new target space. We will refer to our method as Multi Home Transfer Learning (MHTL for short). We denote N individual sources as $S_1, ..S_N$ and the single target space as T.

We assume that the physical aspects of the spaces, the number and type of sensors, and also the residents and their schedules can be different. We also do not

require all of the activities to exist across all spaces. In this work, the number of available sources (N) is limited and computationally manageable, as reducing the number of sources and source selection is beyond the scope of our work. We do not discriminate between activities performed by different residents. Multi-resident problems have been studied by several researchers and interested readers can refer to related literature [Crandall and Cook, 2008]. We also assume that the activities' steps are contiguous, i.e. there is no interrupting event in the middle of a specific activity. The activities are also assumed to be consistent over time, i.e. we assume that their pattern does not change over time. In order to prevent any label mismatch between different environments, we define a set of standard rules for annotating activity data. All the annotations adhere to these standard rules. It is also possible to apply a preprocessing step to achieve a unified labeling.

We hypothesize that we will be able to recognize activities in the target space T using little to no labeled data and using only limited unlabeled data. This assumption turns the nature of the problem into a domain adaptation problem [Pan and Yang, 2010]. In other words, labeled data is available in the source domain, but no or few data labels are available in the target domain. This allows us to reduce several weeks or months of data collection and annotation in the target space to only a few days' worth of data collection. An effective way to perform this collection and annotation with minimal queries is discussed in the next chapter.

Our ultimate objective is to be able to correctly recognize the activities in the target space. By using our method, labeled target activity data becomes available that can be consumed by conventional learning algorithms to perform activity recognition. The labeled activities also can be used as a bootstrap method for other techniques such as active learning techniques in order to quickly improve the recognition results over time using limited data. In the remainder of this section we describe our notations and also we will provide a high level description of the algorithm.

The input data is a sequence of sensor events e each in the form $e = \langle t, id, l \rangle$ where t denotes a timestamp, id denotes a sensor ID, and l refers to the activity label, if available. Each sensor ID is associated with its room name (e.g., kitchen) which we will refer to as a location tag. We use a standard set of location tags across all different sources. The location tags define a simple ontology based on location of sensors. This facilitates transfer of activity patterns between different spaces.

We define an activity as $a = \langle \mathcal{E}, l, t, d, \mathcal{L} \rangle$, where $\mathcal{E}$ is a sequence of n sensor events $\langle e_1, e_2, ..e_n \rangle$, l is its label (if available), t and d represent the start time and duration of the activity, and $\mathcal{L}$ represents the set of location tags where a has occurred. Note that the start time and duration in general are represented as mixture normal distributions, though initially most activities' start times and durations consist only of a single data point — later during activity consolidation the distribution will be formed. As can be seen from the activity's definition, each activity has structural

information in the form of a sensor sequence $\mathcal{E}$, temporal information in the form of t and d, and spatial information in the form of $\mathcal{L}$. These features allow us to convert raw data into an activity model suited for mapping.

We denote the set of activities in each individual source space S_k by $\mathcal{A}_k$. The set of all source activities is denoted by $\mathcal{A}$ where $\mathcal{A}$ is the union of activities from all individual sources, i.e. $\mathcal{A} = \bigcup_k \mathcal{A}_k$. We denote the set of target activities by $\mathcal{A}_T$. The set of all source sensors is denoted by by $\mathcal{R}$, and the set of sensors for S_k is denoted by $\mathcal{R}_k$. The set of target sensors is denoted by $\mathcal{R}_T$.

In order to be able to map activities from the source space to a target space, we need to find a way to map the source sensor network to the target sensor network, as the source sensors can have different locations and properties than the target sensors. Therefore we need to find the mapping $\mathcal{G}(\mathcal{R}_k) = \mathcal{R}_T$. Finding a sensor mapping $\mathcal{G}$ allows us to map a source activity to a target activity based on structural similarity (sensor similarity) and based on the way that the source activity's sensors map to the target activity's sensors. Based on using the sensor mappings $\mathcal{G}$ (the structural mapping) as well as on available temporal and spatial features, we will find the activity mapping function $\mathcal{F}(\mathcal{A}_k) = \mathcal{A}_T$.

To show how well a source activity (or sensor) is mapped to a target activity (sensor), we use mapping probabilities. The mapping probability of an activity $a \in \mathcal{A}_k$ to activity $b \in \mathcal{A}_T$ is reflected in matrix M_k, where $M_k[a, b] \in [0..1]$ shows the

likelihood that activity a and b have the same label. Similarly, a second matrix $m_k[p, q] \in [0..1]$ shows the probability that sensor $p \in \mathcal{R}_k$ maps to sensor $q \in \mathcal{R}_T$ based on their location and their role in activity models. Note that the mappings need not to be one to one, as the number of sensors and the number of activities can be different in the source and target spaces.

MHTL's activity discovery and knowledge transfer is performed in several stages (see Figure 5.4). First we process the labeled data from the source space and mine the available unlabeled data from the target space in order to extract the activity models in each space. In the source space, for each individual source S_k we extract the activity models $\mathcal{A}_k$ by converting each contiguous sequence of sensor events with the same label to its activity model. To reduce the number of activities and to find a canonical mapping, we consolidate similar activities in $\mathcal{A}_k$ together to represent an "activity template". To avoid mapping irrelevant sensors, we use a filter feature selection method based on mutual information [Guyon and Elisseeff, 2003] to discard the irrelevant sensors for each activity template. In the target space we mine the data to find unlabeled activity patterns based on using location closure. Target activities are then consolidated using an incremental clustering method [Can, 1993]. If any labeled data is available in the target space, it can be used to refine the target activity models.

In the next step, we map the source activity templates to the target activity

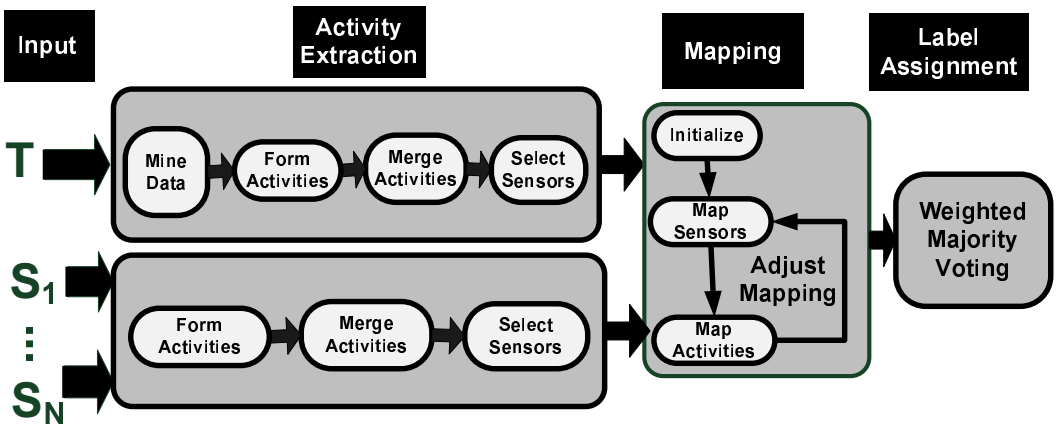


Figure 5.4: Main components of MHTL for transferring activities from multiple source spaces to a target space.

templates. MTHL begins by computing the activity templates' initial mapping probabilities based on structural, temporal and spatial similarities. The sensors' initial mapping probabilities are assigned based on a spatial similarity measure. After initialization, we compute the mapping probabilities in an iterative manner using an Expectation Maximization-like framework [Dempster et al., 1977], which we refer to as a semi-EM process. First, we adjust the sensor mapping probabilities based on the activity mapping probabilities. Next, we adjust the activity mapping probabilities based on the updated sensor mapping probabilities. This continues until no more changes are perceived or until a user-defined number of iterations is reached.

The final step involves assigning labels to target activities. We assign an activity

label to each target activity b based on the obtained activity mapping probabilities M. To map activity labels we use an ensemble method based on a weighted majority voting. Each space S_k casts a vote for the label of the target activity b. The voted label is selected the same as the label of a source activity a that maximizes the mapping probability M_k for b. Each vote is weighted by the overall similarity between the source space S_k and the target space T, as will be described later. At the end, the label with the maximum weighted votes is considered as the label of the activity b. Note that in this method all the sources contribute to the label mapping process in order to generate a final activity label for each target activity. We provide a more detailed description of these steps in the next sections.

Activity Model Extraction

The first step of the multi-stage MHTL algorithm is to extract the activity models from input data in both source and target spaces. For each single source space S_k we convert each contiguous sequence of sensor events with the same label to an activity a. This results in finding the set of activities $\mathcal{A}_k$ for each one of the individual source spaces S_k. The start time of the activity is the timestamp of its first sensor event, while its duration is the difference between its last and first timestamps. Due to the prohibitively large number of extracted activities and possible similarity among them, we combine similar activities together as an "activity template". Representing a set of similar activities as an activity template allows for a more efficient canonical

mapping from source to target, as only a few activity templates will be mapped from source to target instead of mapping a large number of similar activities with only minor differences. The activity template for a set of activities is itself an activity, formed by merging activities' sensors, durations, and start times where the merged start times and durations form a mixture normal distribution.

The temporal mixture model allows us to capture and model variations of the same activity that occur at different times. For example, consider the "eating" activity which usually happens three times a day, once in the morning as breakfast, once at noon as lunch, and once at night as dinner. Using a mixture model for the start time we are able to capture all three variations by using a single activity model. The method for obtaining the mixture model is demonstrated in Algorithm 2. The input to algorithm 2 is the set of all timestamps for an activity a and the time interval granule in hours denoted by r. Using this method allows for a variable number of distributions to be discovered. A similar method is used for obtaining duration distributions where the number of duration distributions equals the number of obtained start time distributions for that activity.

During activity consolidation, all the source activities that have the same label will be merged into one single activity template. Note that as each activity template is itself an activity, we use the terms activity and activity template interchangeably.

The next step after similar activities are consolidated is to perform sensor se-

Algorithm 2 The Start Time Mixture Model

procedure FINDMIXTUREMODEL(a, r)

for all timestamps t belonging to a **do**

Find $t' \in [1..\frac{24}{r}]$ s.t. $t \in [t' - \frac{r}{2} .. t' + \frac{r}{2}]$ ▷ Find the right interval

$c[t'] = c[t'] + 1$ ▷ Increase its count

end for

$\tilde{c} = \frac{r * \sum c}{24}$

for all $c[t'] > \tilde{c}$ **do** ▷ If frequency > Average

Make t' a centroid ▷ Form initial centroids

end for

for all timestamps t **do** ▷ Find final centroids

Assign t to closest centroid

Recompute centroids

end for

end procedure

lection for each activity template a by preserving only relevant sensors. Performing sensor selection on each activity template allows for an even more compact representation and a more accurate mapping, as it allows us to map only the relevant sensors and to avoid mapping the irrelevant sensors as noise. Our sensor selection method is a filter feature selection method based on mutual information [Guyon and Elisseeff, 2003]. For each activity template a and each sensor s we define their mutual information $I(s, a)$ as in Equation 5.5. This value measures their mutual dependence and shows how relevant sensor s is in predicting the activity's label. Here $P(s, a)$ is the joint probability distribution of s and a, while $P(s)$ and $P(a)$ are the marginal probability distributions, all computed from the sensor and activity occurrences in the data. A high mutual information value indicates the sensor is relevant for the activity template. We simply consider sensors with a mutual information above the midpoint (0.5) as relevant, otherwise they will be discarded.

$$I(s, a) = P(s, a) * \log \frac{P(s, a)}{P(s)P(a)} \tag{5.5}$$

To find activity patterns in unlabeled target data, we perform a data mining step on the input data. First we partition the input data into activities. A sensor event $e_1 = \langle t_1, id_1, l_1 \rangle$ and its successor sensor event $e_2 = \langle t_2, id_2, l_2 \rangle$ are part of the same activity if $L_1 = L_2$, i.e. if both sensors are in the same location. Such a local partitioning allows us to have a baseline for finding individual activities. This

approach is based on the intuition that occurrences of the same activity usually happen within the same location (such as preparing meals in the kitchen, grooming in the bathroom, etc.), and more complex activities occurring in different locations can be composed of those basic activities. Notice that as we only have access to limited input data (perhaps a few days or even a few hours), we cannot use conventional activity discovery methods such as frequent or periodic sequence mining methods to find activity patterns in the data. Therefore, exploiting the spatial closure can be a way to overcome this problem.

After partitioning data into initial activities, we consolidate those activities by grouping together similar activities into an activity template. To combine activities together, we use an incremental clustering method [Can, 1993], such that each activity is assigned to the most similar centroid if their similarity is above threshold ς, and then the centroid is recomputed. Otherwise the activity forms a separate cluster. The centroid is itself represented as an activity template. At the end all the activities in one cluster are consolidated together and the sensor selection is carried out. For two activities a and b, their similarity $\Upsilon(a, b)$ is defined similar to what we defined in Equation 3.18.

Mapping Sensors and Activities

The next step after the activity models for the source and target space have been identified is to map the source activity templates to the target activity templates.

First we initialize the sensor and activity mapping matrices, m_k and M_k, for each source and target pair (S_k, T). The initial values of the sensor mapping matrix $m_k[p, q]$ for two sensors $p \in \mathcal{R}_k$ and $q \in \mathcal{R}_T$ is defined as 1.0 if they have the same location tag, and as 0 if they have different location tags. The initial value of $M_k[a, b]$ for two activities $a \in \mathcal{A}_k$ and $b \in \mathcal{A}_T$ is obtained based on exploiting related spatial and temporal information and also prior activity label information (if available), as in Equation 5.6. Note that in Equation 5.6 the first case applies to the few labeled target activities, while for the majority of the target activities the second case is applied.

$$M_k[a, b] = \begin{cases} 1.0 & if \quad l_a = l_b \\ \Upsilon(a, b) & otherwise \end{cases} \tag{5.6}$$

For computing subsequent mapping probabilities, we use an EM-like framework [Dempster et al., 1977] by estimating the mapping probabilities in an iterative manner. First, the sensor mapping probabilities are computed; and in the next step the activity mapping probabilities are maximized based on the sensor probabilities. Though this model doesn't exactly reflect an EM algorithm, due to its iterative manner and likelihood estimation in two steps, we refer to it as a semi-EM framework.

To compute sensor mapping probabilities $m_k[p, q]$ for sensors $p \in \mathcal{R}_k$ and $q \in \mathcal{R}_T$, we rely on activities in which p and q appear, as in Equation 5.7. The learning

rate α refers to how fast we want to converge on the new values, while $m_k^n[p,q]$ and $m_k^{n+1}[p,q]$ refer to the current and updated values of $m_k[p,q]$ in iterations n and $n+1$, respectively.

$$m_k^{n+1}[p,q] = m_k^n[p,q] - \alpha * \Delta m_k[p,q] \tag{5.7}$$

$$\Delta m_k[p,q] = m_k^n[p,q] - \frac{1}{|X_p||Y_q|} \sum_{a \in X_p} \sum_{b \in Y_q} M_k[a,b] \tag{5.8}$$

$$\begin{aligned} X_p &= \{a \in \mathcal{A}_k | p \in \mathcal{E}_a\} \\ Y_q &= \{a \in \mathcal{A}_T | q \in \mathcal{E}_a\} \end{aligned} \tag{5.9}$$

In Equation 5.8, X_p and Y_q give us all the activities in which sensors p and q appear. This means that those activities which do not include a given sensor will not contribute to that sensor's mapping probability.

In the next step, to adjust the mapping probability between each two activities, we use Equation 5.10 to account for the updated sensor mappings. Here $M_k^n[a,b]$ and $M_k^{n+1}[a,b]$ refer to the current and updated values of $M_k[a,b]$ in iteration n and $n+1$, respectively.

$$M_k^{n+1}[a,b] = M_k^n[a,b] - \alpha * \Delta M_k[a,b] \tag{5.10}$$

$$\Delta M_k[a,b] = M_k^n[a,b] - \frac{1}{|\mathcal{E}_a|}\sum_{p\in\mathcal{E}_a} \arg\max_{q\in\mathcal{E}_b}\{m_k[p,q]\} \tag{5.11}$$

The above procedure for computing sensor mapping and activity mapping probabilities is repeated until no more changes are perceived or until a pre-defined number of iterations is reached. Next, the labels are assigned to the target activities based on the obtained probability mapping matrices.

Label Assignment

In order to assign labels to the target activities, we use a voting ensemble method [Dietterich, 2000] based on the activity models $\mathcal{A}_k$ for each space S_k. Combining data from different sources to improve the accuracy and to have access to complimentary information is known as data fusion or as a form of ensemble learning [Jacobs et al., 1991]. Ensemble learning is a strategic way to combine multiple models, such as different classifiers or hypotheses to solve a computational problem. In our problem, using multiple sources allows us to fuse data from different sources and to form different activity models, therefore being able to model target activities using knowledge gleamed from a more diverse set of source activities. Note that for the HHTL method, instead of a voting method, we use the suggested label from the single source directly.

In order to be able to successfully apply the ensemble learning technique, an ensemble system needs classifiers whose decision boundaries are adequately different from each other. The most popular method is to use different training datasets to train

individual classifiers. The diversity condition of ensemble learning in our problem is achieved by using different training sets from N different physical source spaces, resulting in N different hypotheses. We build a classifier based on each individual hypothesis h_k and then by combining the predicted labels of all classifiers for a certain target activity we are able to make a decision about the activity's final label.

Each hypothesis h_k is constructed based on using the activity templates $\mathcal{A}_k$ for space S_k plus the activity and sensor mapping probabilities M_k and m_k. We represent each hypothesis as $h_k = \{\mathcal{F}(\mathcal{A}_k), \mathcal{G}(\mathcal{R}_k)\}$ where $\mathcal{F}$ and $\mathcal{G}$ denote the activity and sensor mapping functions. For a single space S_k, Equations 5.12, 5.13 and 5.14 provide us with the activity mapping function $\mathcal{F}$, sensor mapping function $\mathcal{G}$ and the assigned label l_b for each activity $b \in \mathcal{A}_T$. As can be seen in Equation 5.14, the target activity's label is selected to be the same as the label of a source activity $a \in S_k$ that maximizes the mapping probability M_k for a.

$$\mathcal{F}(a) = \max_b(M_k[a, b]) \tag{5.12}$$

$$\mathcal{G}(p) = \max_q(m_k[p, q]) \tag{5.13}$$

$$l_b = l_a \quad s.t. \quad M_k[a, b] = \max_z(M_k[z, b]) \tag{5.14}$$

In order to combine the assigned labels for each b using different hypotheses,

Algorithm 3 The weighted majority vote algorithm for label assignment

procedure AssignLabel($\mathcal{A}$, M, m, b)

for all S_k **do**

$l \leftarrow l_a \quad s.t. \quad M_k[a,b] = \max_z(M_k[z,b])$ ▷ Vote

add l to the set of voted labels

$Sim(S_k, T) \leftarrow \sum_{a \in \mathcal{A}_k} M_k[a, \mathcal{F}(a)]$ ▷ Find overall similarity

$W[l] = W[l] + \frac{Sim(S_k,T)}{|\mathcal{A}_k|}$ ▷ Increase label's weight

end for

$l_b \leftarrow \max_l W[l]$ ▷ Find the final label

return l_b ▷ Assigned label is l_b

end procedure

we use the weighted majority voting algorithm as in Algorithm 3. The input of this algorithm is the source activities $\mathcal{A}$, the activity mapping probabilities M, the sensor mapping probabilities m, and activity $b \in \mathcal{A}_T$. The output of the algorithm is the label of b as l_b. For each source space S_k we find the label of b by using Equation 5.14. Each predicted label l is associated with a weight $W[l]$, which is the total similarity between the source S_k and T. The total similarity between S_k and T is calculated as in Equation 5.15 by summing over the best mapping from S_k to T for each $a \in S_k$. Obviously a label can be voted for by different hypotheses and its weight will be increased as a result.

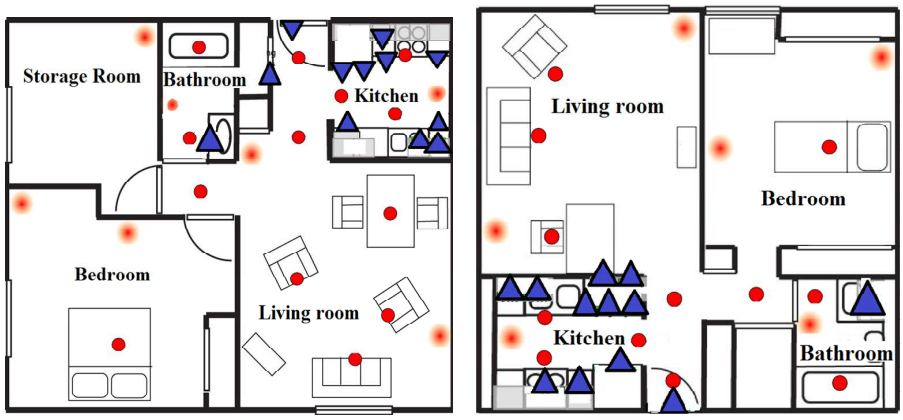
$$Sim(S_k, T) = \sum_{a \in \mathcal{A}_k} M_k[a, \mathcal{F}(a)] \tag{5.15}$$

At the end, the label with the greatest number of weighted votes is selected as the label of the activity b. After obtaining the labels of all target activities, we can use the obtained labels to train a conventional activity recognition algorithm. We also can use the labels in conjunction with other techniques such as active learning in order to further improve the results.

5.2.3 Experiments

We evaluated the performance of our MHTL algorithm using the data collected from six different smart apartments. The layout of the apartments, including sensor placement and location tags, are shown in Figure 5.5. We will refer to the apartments in Figures 5.5a through 5.5f as apartments 1 to 6. The data was collected during a three month period for apartments 1, 2, and 3, and during a two month period for apartments 4, 5 and 6. Each apartment is equipped with motion sensors, and most of the apartments are also equipped with contact sensors which monitor the open/closed status of doors and cabinets. Apartment 5 is also equipped with light sensors and some item sensors to sense the presence of key items. As can be seen in Figure 5.5 the apartments have different layouts. For example, apartments 3, 4 and 6 have two bedrooms, while apartments 1 and 2 have one bedroom. In addition, some functional spaces might not be available in all five apartments, such as the workspace, laundry room or the music room. All the sensor data is captured and stored in a SQL database, using a publish/subscribe protocol middleware.

The residents also have quite different schedules, as can be seen from the activity distribution diagrams shown in Figures 5.6a through 5.6f. For example, in apartment 1 housekeeping is performed each Friday, while in apartment 2 this is performed once a month, and in the third apartment the housekeeping activity is replaced by a work



(a) Apartment 1. (b) Apartment 2.

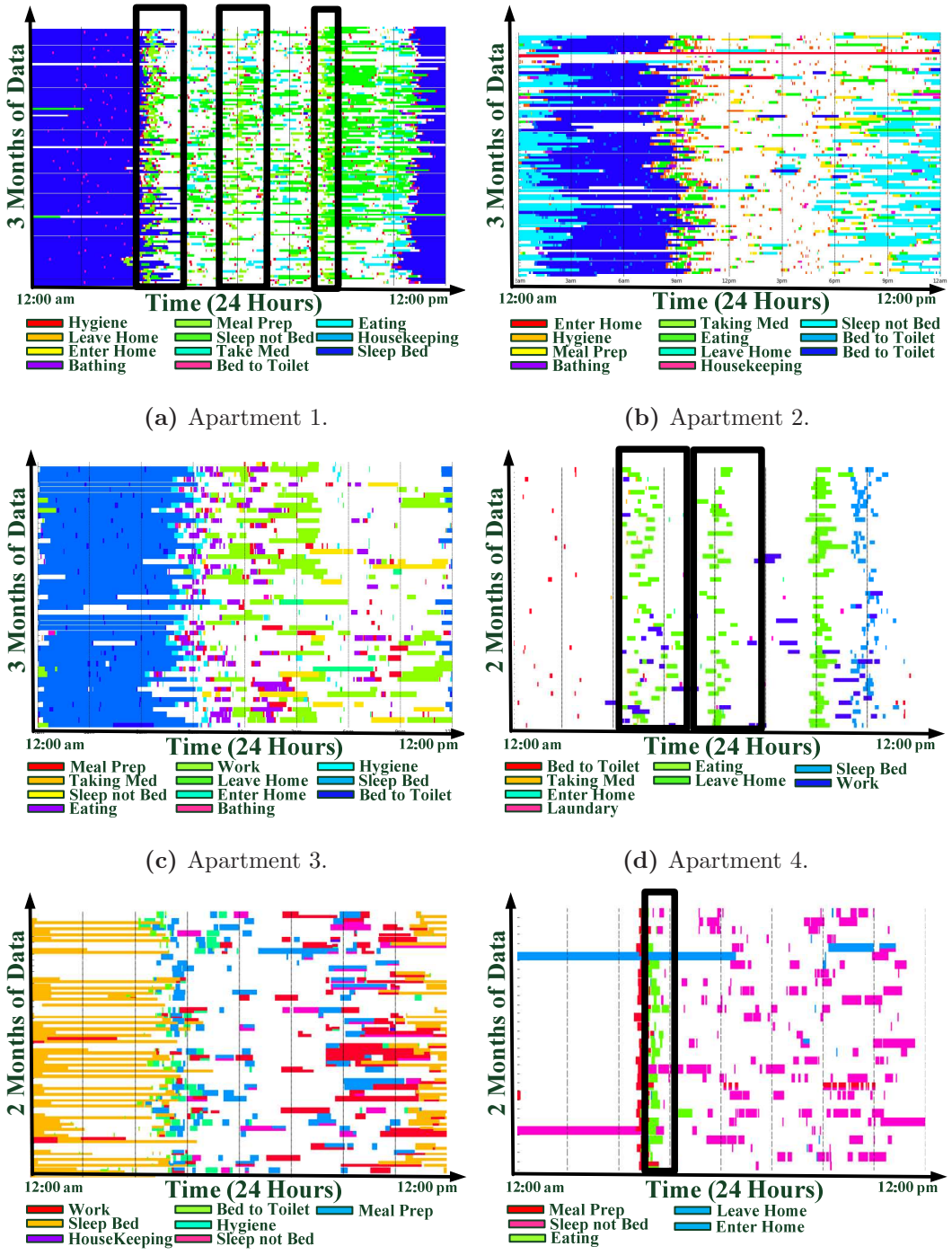
(c) Apartment 3. (d) Apartment 4.

(e) Apartment 5. (f) Apartment 6.

Figure 5.5: Figures (a-e) show sensor map and location tags for each apartment. Circles: motion sensors, triangles: contact sensors, stars: light sensors, hollow-shaped sensors: area motion sensor.

activity. Also the activity level in each apartment is different, as can be seen clearly by comparing activity distribution diagrams for apartments 4 and 6 versus other apartments. The activity level is dependent on the activity level of the residents as well as the number of sensors that monitor the activities. The three first apartments were single resident apartments, while for apartment 4 the residents included a man, a woman, and a cat. Apartments 5 and 6 included two residents. All the data was collected while residents were performing their normal daily activities during a 2-3 month period.

Each of the datasets was annotated with activities of interest for the corresponding resident and apartment. A total of 11 activities were noted for apartments 1, 2 and 3. Those activities included bathing, bed-toilet transition, eating, enter home, housekeeping (for the third apartment this is replaced by "work"), leave home, meal preparation, personal hygiene, sleeping in bed, sleeping not in bed (relaxing) and taking medicine. For apartment 4, 7 activities were noted including bed-toilet transition, taking medicine, eating, leaving home, laundry, sleeping in bed and working. Apartment 5 included the 7 activities working, sleeping in bed, bed-toilet transition, personal hygiene, meal preparation, housekeeping, sleeping not in bed (relaxing). Apartment 6 activities included the 5 activities meal preparation, sleeping in bed, eating, leaving home and entering home. As can be seen from activity labels in various spaces, we have defined standard rules for annotating activity data. The



(a) Apartment 1.

(b) Apartment 2.

(c) Apartment 3.

(d) Apartment 4.

(e) Apartment 5.

(f) Apartment 6.

Figure 5.6: Figures (a-e) show residents' activity per a 24 hour for 2-3 month of data. The boxes show the approximate boundary of "eating" activity for apartments 1, 4 and 6.

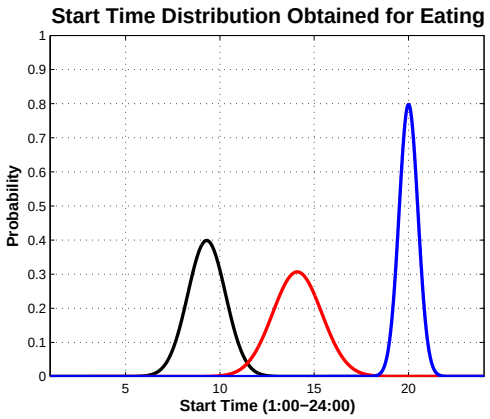
annotators are required to adhere to those rules to prevent any label mismatch between different environments. One can expect that in a real world situation either such standardization is enforced, or a preprocessing step is done to achieve a unified labeling.

We ran our algorithm for each one of the apartments as the target space, resulting in six different transfer learning problems. In each setting, all the apartments except for the target apartment were used as the source apartments. In each setting, we used all the available source labeled data, 1 to 7 days of target unlabeled data, and 0 to 1 days of target labeled data.

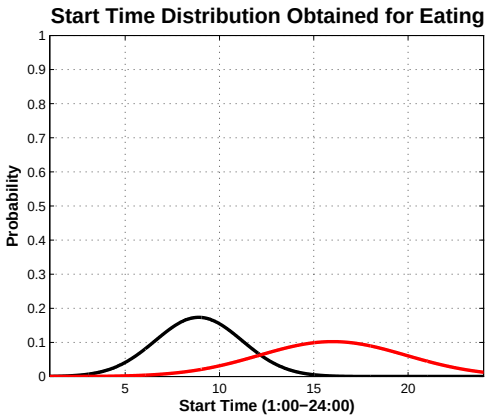
The first step, activity extraction, resulted in a considerable reduction in the number of source activities. In particular 3384, 2602, 1032, 428, 492, and 643 activity instances from apartments $1-6$ were represented by as few as 11, 10, 9, 7, 7, and 5 activity templates. The reason that we have obtained fewer templates than the 11 predefined activities in the second and third apartment is that the "eating" activity was done rather in an erratic way and in different locations, therefore our sensor selection algorithm did not choose any specific sensor for that activity, and as a result the activity was eliminated. The same applied for "taking medicines" in apartment 3. This shows how our algorithm can avoid mapping very irregular activities. It is also possible to obtain a more regular form of such activities by using object sensors such as RFID tags on items.

Our results also show how the algorithm condensed the activity instances into a compressed representation, as we approximately obtained the 11 predefined activities for the first three apartments and exactly 7, 7 and 5 activities for the last three apartments. During activity extraction, the number of sensors included for each activity template was reduced from an average of 76.85 sensors to 4.04 sensors, as the algorithm removed the irrelevant sensors and preserved only the relevant sensors. This shows that for each activity a few key sensors can be used to identify the activity, e.g. taking medicine can be identified by the cabinet sensor where the medicines are kept. The algorithm was able to successfully find the mixture normal distributions for temporal features. Figure 5.7 shows the obtained distributions for the "Eating" activity in apartments 1, 4, and 6. It can be seen from the distributions that apartment 1 resident has a regular schedule for eating (three times a day). The residents in apartment 4 seem to have a more irregular schedule, usually missing dinner, while apartment 6's residents seem to have a very regular schedule only for breakfast. Comparing the discovered mixture models to the activity distribution diagrams in Figure 5.5 confirms the correctness of our results ("Eating" activity is surrounded by boxes in Figures 5.6a, 5.6d and 5.6f).

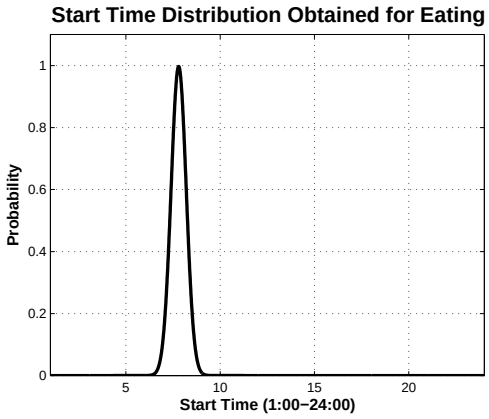
In the target space, data was mined to extract the activity templates. For example, using three days of unlabeled target data and no labeled target data, we discovered 8, 7, 7, 5, 5 and 5 activity templates for apartments 1 through 6. The



(a) Apartment 1.



(b) Apartment 4.

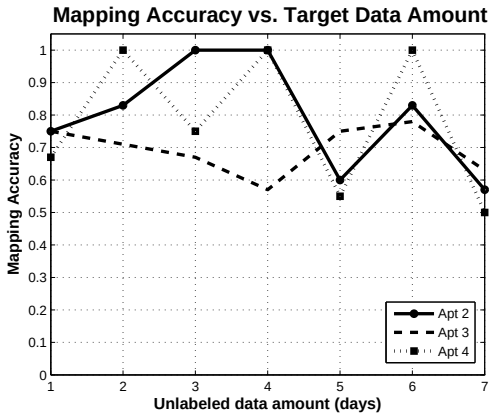


(c) Apartment 6.

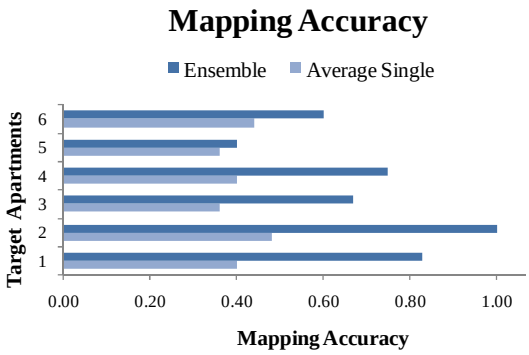
Figure 5.7: Start time's mixture normal distribution for the "eating" activity. Here the time interval granule r was set to 6 hours.

similarity threshold ς in those experiments was set to the midpoint 0.5. The reason that fewer activity templates are discovered compared to the predefined activities is because some similar activities might be merged into one activity, such as relaxing and eating which happen at similar times and similar places. In addition, some activities cannot be easily discovered based only on a few days of data. One example is the housekeeping activity which happens quite rarely compared to other activities; and even if it happens to be in the data, because of its erratic nature and occurring all over the home, it is not very easy to discover.

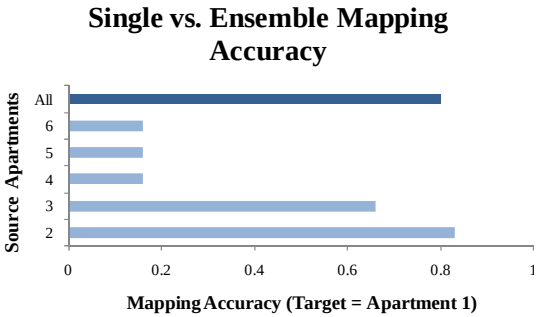
In the next step the source activities were mapped to the target activities. In order to be able to evaluate the mapping accuracy of our algorithm, we embedded the actual labels of target activities in data. This label is not used during training, rather it is only used at the end to verify the correctness of the results. Mapping accuracy is defined as the number of activities in the target space whose transferred label matches the correct expert-supplied label. Figure 5.8a shows the mapping accuracy for different amounts of unlabeled target data and no labeled target data, in several different settings. Figure 5.8b shows a comparison between mapping accuracy based on using multiple sources vs. average mapping accuracy using a single source, using 3 days of unlabeled target data. Figure 5.8c shows the individual single mapping accuracies for apartment 1 versus its ensemble mapping accuracy. It can be seen that on average the ensemble method can beat the single source method. The mapping



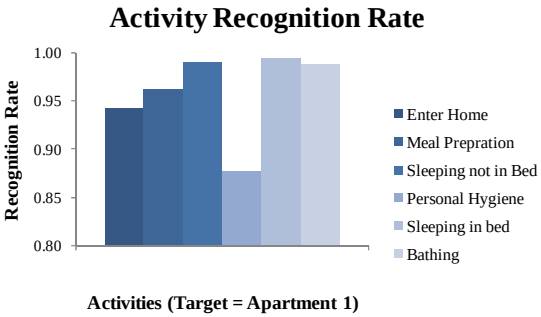
(a) Mapping accuracy in several different settings.



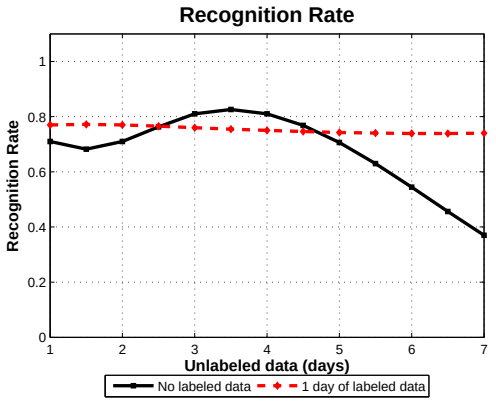
(b) Single vs. ensemble mapping accuracy.



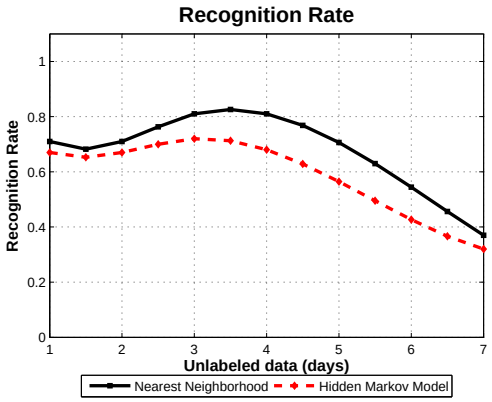
(c) Single vs. ensemble mapping accuracy for apartment 1.



(d) Activity recognition rate for apartment 1.



(e) 1NN's recognition rate using 0 and 1 day of labeled target data.



(f) Recognition rate for 1NN and HMM using no labeled data.

Figure 5.8: Mapping accuracies and recognition rates.

accuracies vary from space to space, depending on the consistency of activities in target space, as well as the similarity between the source and target spaces, the amount of data available in the source space used to create the activity model, the dimension of the space that has been shifted (new environment, new sensors, or new person). It should be noted that some activities might not be present in all spaces, such as working or housekeeping. The same applies for lack of certain spaces in different apartments, such as laundry room or workspace. We noted that transfer between apartments that have a more similar layout and functional structure is more satisfactory.

We tested two of our own activity recognition algorithms on the transferred labeled data. The first algorithm is a nearest neighborhood (1NN) algorithm based on the similarity measure in Equation 3.18. The second algorithm represents activities and sensor events with a hidden Markov model and learns the activities using the Viterbi algorithm. The models performed almost the same with the nearest neighborhood algorithm sometimes slightly outperforming HMM due to its use of temporal and spatial features. Though HMM is considered as a temporal method, it should be noted that it considers temporal information in the sense of "order of states", not "start time" or "duration" of states. The HMM does not take into account how much time has been spent in a specific state, or when did a transition occur to another state. Using the embedded labels we define the recognition rate as the percentage of

sensor events predicted with the correct label. Figure 5.8d shows the recognition rate of individual activities in apartment 1 using 3 days of unlabeled target data. Figure 5.8e shows 1NN's recognition rate for apartment 1 as the target apartment using 0 and 1 day of labeled target data. Figure 5.8f shows 1NN and HMM recognition rates for apartment 1 as the target apartment.

Our results show that despite using little to no labeled target data, and having different layouts, schedules and different activities, both algorithms still achieve a reasonable recognition rate in a target space using data from a source space. It should be noted that in some cases, transferring from a source space to a target space might not provide the best results. This happens when the source and target are not very similar. In such cases, one can apply a source selection method to select the best subset of sources among a set of available sources.

5.3 Domain Selection

In this section, we introduce a method for selecting the best promising sources among a number of available sources and then adapting the activity models from those sources. Our method employs ideas from domain adaptation and domain selection solution proposed by NLP community. We show how by selecting domains in a multiple source problem, we can achieve even higher accuracy rates. Also for

domain adaptation, in previous methods we relied on directly mapping the sensors and activities from a source domain to a target domain. Here, we take a slightly different approach by using common features across our different domains.

5.3.1 Introduction

In previous sections, we have shown a method of transferring activity knowledge from a source environment to a target environment based on an EM framework. There we assumed that the number of sources is limited and that all of the sources are equally similar. In this section, we generalize the problem in the context of multi source domain adaptation, and we propose a novel method of combining hypotheses from different source environments. Here we assume that not all the sources are equally similar. We explore a method of measuring the distance between a source and a target environment, called $\mathcal{H}$ distance. Based on such a measure, we propose a method for selecting the most promising sources from a collection of available sources. This allows us to achieve a high accuracy rate, while maintaining efficiency. It also allows us to avoid the negative transfer effect [Rosenstein et al., 2005]. The negative transfer effect refers to a case where adapting from a source domain might indeed result in performance degradation. It happens if the source and target domains not very similar. To the best of our knowledge, this is the first work to address activity

recognition as a multi source domain adaptation problem, and to provide a method for measuring the distance between activities performed in different environments.

Our method is based on multiple source domain adaptation [Pan and Yang, 2010, Blitzer et al., 2006, Belkin et al., 2006, Blitzer et al., 2007, Gupta and Sarawagi, 2008, Duan et al., 2009]. Proposed by the machine learning community, domain adaptation essentially tries to adapt an algorithm that has been trained on a source domain to a target domain. This problem arises in many applications, such as natural language processing, speech processing, computer vision and many other applications. For example, one might train an algorithm on documents collected from a financial domain, but test it on documents collected from a medical domain [Belkin et al., 2006]. A more complex variant of the problem, multi source domain adaptation, tries to combine information from multiple source domains to make predictions about a target domain [Duan et al., 2009, Crammer et al., 2008, Luo et al., 2008].

Adopting the domain adaptation formulation, we assume that we have access to relatively large amounts of labeled data from multiple source smart environments (e.g. home A and B), and limited "unlabeled" data is available from a new target smart environment (e.g home C). Our objective is to recognize activities in a target environment based on the source environments' knowledge. This allows us to skip the laborious and invasive step of activity annotation in a target environment, while achieving an accelerated deployment. Using multiple sources increases the generaliza-

tion capability of our algorithm and increases the chance that an activity is correctly recognized in a target environment, as we showed in previous sections. In addition, selecting the most promising sources among multiple sources allows us to achieve a tradeoff between efficiency and accuracy, while avoiding the negative transfer effect as much as possible. Our bootstrap domain adaptation technique can be combined with other techniques such as active learning [Lewis and Gale, 1994] in order to achieve even higher accuracies, by tailoring the solution to the target environment over time.

The remainder of this section is organized as follows. First we provide the definitions and notations in Section 5.3.2, followed by explaining $\mathcal{H}$ distance in Section 5.3.2. The proposed model is described in Section 5.3.2. We then show the results of our experiments on eight different datasets obtained from different smart apartments in Section 5.3.3.

5.3.2 Model

In the following subsections, first we will give an overview of definitions and notations. Next, we will describe our model in more detail.

Definitions

The input data to our system is a sequence of sensor events. Each sensor event is timestamped with a timestamp τ, and is associated with a sensor ID. The sensor ID

uniquely identifies the sensor that has been triggered. Also, in case of annotated data, each sensor event is associated with an activity label l. An activity label identifies the activity which this sensor event is part of, such as "meal preparation" or "sleeping". Such activity labels usually are provided manually by a human annotator for each new environment. As mentioned previously, obtaining such annotations is very laborious and time consuming. Note that we assume that we only have access to a few days of unlabeled data in the target environment, while relatively large amounts of labeled data might be available from source environments. This is in contrast to most domain adaptation methods which assume that we have access to ample amounts of unlabeled data from target domain.

We also assume that the source distributions are different. One can imagine that activities performed in different environments by different residents will not have the same distribution. Figure 5.9 confirms our conjecture. It shows the start time distribution of activities over a 24 hour period for two different datasets used in our experiments. One can clearly see that though the distributions show some similarity, however they are not identical.

One of the main approaches for adapting a source domain to a target domain is instance transfer [Pan and Yang, 2010]. It tries to re-weight the data instances according to their distribution in the target domain. Instead of weighting each individual instance, we weight the decisions made by different source domains, according to the

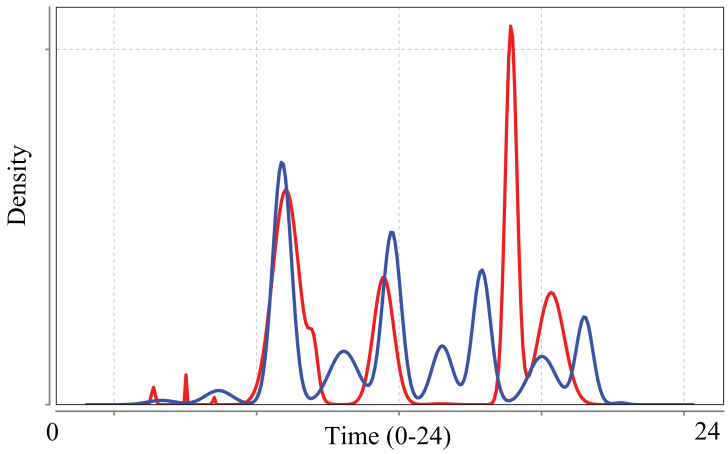


Figure 5.9: Start time distribution of activities in B1 (blue) vs. M (red) datasets.

source and target domains' distribution similarity for a particular activity. Another commonly used approach in domain adaptation is "feature representation change" [Pan and Yang, 2010], which tries to find a common representation of features in source and target domains. We use a common simple representation of features in the source and target domains in order to be able to generalize between activities performed in various environments.

We define a domain $\mathcal{D}$ as a pair consisting of two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(\mathcal{X})$. Each $x \in \mathcal{X}$ can be associated with a label $y \in \mathcal{Y}$ and is written as (x, y). Here $\mathcal{Y}$ refers to the set of labels. For example, features $\mathcal{X}$ can be defined as sensor events and their timestamps, $P(\mathcal{X})$

can be defined as the activation probability of sensors during different time intervals, and $\mathcal{Y}$ will correspond to the set of activity labels, (e.g., {sleeping, bathing, meal preparation}).

For a domain $\mathcal{D}$, we have a label function $f(\mathcal{X}) : \mathcal{X} \rightarrow \mathcal{Y}$ serving as ground truth. For each domain we can also infer a hypothesis function $h(\mathcal{X}) : \mathcal{X} \rightarrow \mathcal{Y}$ that approximates f and can be learned from data. Note that $f(\mathcal{X})$ is available only in domains with labeled data. In our model, f corresponds to the annotations provided by human annotators as the ground truth. If no such annotations are provided, then we usually seek to learn a hypothesis h to approximate f. From a probabilistic point of view, $h(x)$ can be written as $P(y|x)$, for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

Unlike most domain adaptation problems, here we do not adopt the covariate shift assumption [Pan and Yang, 2010]. The covariate shift assumption, adopted in most domain adaptation problems, states that the labeling rule, i.e. $P(\mathcal{Y}|\mathcal{X})$, is identical in source and target domains. Instead of the covariate shift assumption, we assume that there might exist some hypothesis that works reasonably well on both domains. It is also possible that a hypothesis from a particular source domain might not work well on the target domain, causing the negative transfer effect.

Casting our problem as a multiple source domain adaptation problem, we consider m environments as source domains, referred to as $\mathcal{D}_s$ for $s = 1 \,..\, m$. We also consider a target environment $\mathcal{D}_t$. In our scenario, the source domains refer to those

smart environments in which we previously have collected data and annotated it. The target domain is a new smart environment in which we intend to deploy an activity recognition system, and in which we only have collected a few days worth of unlabeled data. In other words, we are provided with $f_s(\mathcal{X})$, while $f_t(\mathcal{X})$ is unknown. Our objective is to combine hypotheses $h_1(\mathcal{X})..h_m(\mathcal{X})$ in order to infer $h_t(\mathcal{X})$.

Model Overview

In order to recognize activities in a target smart environment, we combine hypotheses from multiple source smart environments. First, we take a small sample from each one of the source datasets. The size of the sample is chosen to approximately match the size of the target dataset in order to avoid the effects of an unbalanced dataset [Searle, 1987].

Next, we calculate the pairwise similarity between each sample source dataset and target dataset, for each particular activity. We use a distance measure called $\mathcal{H}$ distance. It computes the distribution difference based on finite samples of data [Kifer et al., 2004]. The overall similarity of the two domains is defined as their average similarity with respect to all of their common activities.

In order to avoid the negative transfer effect, we select the most promising sources based on the computed average similarity. This allows us to select only the sources that have a similarity above the average. Next, we combine the hypotheses from various selected sources using a weighted combination. The weights correspond

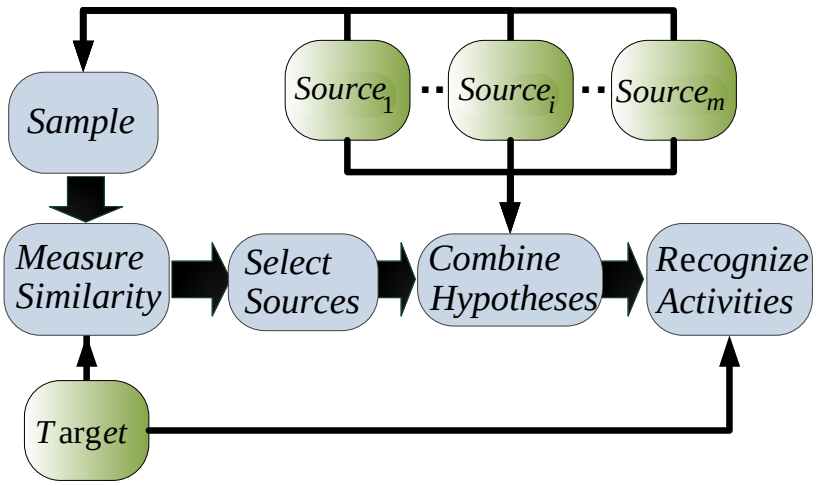


Figure 5.10: Domain Selection and Adaptation Architecture.

to the product of hypothesis' confidence by the similarity of the predicted act in the source and target domains.

Figure 5.10 shows the architecture of our model. In the next subsections, we will explain the details of our model.

Measuring Domain Distance

In order to be able to select the most promising domains among a list of available domains, we need to quantify the difference between the source and target environments. Selecting the most promising sources instead of using all the sources in a brute force manner allows us to avoid the negative transfer effect. The negative transfer effect refers to a case where the performance might actually degrade due to the transfer

[Rosenstein et al., 2005].

A number of measures have been developed for determining distance between two distributions. One such measure is Jensen-Shannon divergence (JSD) [Lin, 1991]. However it requires estimating the distribution forms and only works with discrete distributions. The same is true for Kullback Leibler divergence (KL) [Kullback and Leibler, 1951] which requires us to know the exact form of distributions. Other commonly used distances such as L^1 or L^p norms are either too sensitive or too insensitive, as pointed out by Batu et al. [Batu et al., 2000]. In addition, in a distribution-free setting such as our scenario, we cannot obtain accurate estimates of common measures of divergence such as L^1 or KL divergence from finite samples.

We use a classifier induced divergence measure called $\mathcal{H}$ distance. Proposed by Kifer et al. [Kifer et al., 2004], it allows us to measure the divergence between two distributions. Kifer et al. [Kifer et al., 2004] showed that if the hypothesis space is of finite complexity, we can measure the divergence from finite unlabeled samples of data. The more different the two domains are, the more divergent the distributions would be. Blitzer et al [Blitzer et al., 2006] used a variant of $\mathcal{H}$ distance as a criterion for adaptability between unlabeled domains. Here, we use it as a distance measure in order to combine hypotheses from various domains, and to discard less promising sources to avoid the negative transfer effect.

There are several reasons for using $\mathcal{H}$ distance. The key advantage is that while

two domains can differ in arbitrary ways, $\mathcal{H}$ distance gives us the difference only affecting the classification accuracy. Also, unlike other distribution distance measures such as JSD or KL divergence, it's possible to compute it from finite unlabeled samples of two distributions. More importantly, one can use the well known machine learning methods to estimate the distance.

Let $\mathcal{A}$ be a family of subsets of $\Re^k$ corresponding to the characteristic functions of classifiers. Then the $\mathcal{H}$ distance between two probability distributions $\mathcal{D}$ and $\acute{\mathcal{D}}$ is defined as in Equation 5.16. Here sup refers to supremum and $|.|$ refers to the absolute value.

$$d_{\mathcal{H}}(\mathcal{D}, \acute{\mathcal{D}}) = 2 * \sup_{A \in \mathcal{A}} |Pr_{\mathcal{D}}(A) - Pr_{\acute{\mathcal{D}}}(A)| \tag{5.16}$$

Ben-David et al. [Ben-david et al., 2007] showed that computing $\mathcal{H}$ distance for a finite sample is exactly the problem of minimizing the empirical risk of a classifier that discriminates between instances drawn from $\mathcal{D}$ and $\acute{\mathcal{D}}$. In other words, we should label all the data points in the source domain as 0, and all the data points in the target domain as 1, regardless of their actual class labels. Then we can train a classifier to discriminate between the two domains, separating data points with a label of 0 from those data points with a label of 1. The more similar are the data points in the source and target domains, the more difficult it would be to separate the source and target data points, and therefore the higher would be the empirical risk.

Based on this conclusion, we re-write Equation 5.16 as Equation 5.17. Here ϵ_h is the empirical error when discriminating between source domain $\mathcal{D}_s$ and target domain $\mathcal{D}_t$ based on some hypothesis $h \in \mathcal{H}_s$. Here $\mathcal{H}_s$ is the hypothesis space of source domain.

$$d_{\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) = 2 * (1 - \arg\min_{h_j \in \mathcal{H}_s} (\epsilon_{h_j})) \tag{5.17}$$

We compute such a distance measure for each pair of source and target domains, and for each activity $y \in \mathcal{Y}$ belonging to the target environment. This allows us to obtain a clear picture of domain similarity based on various activities.

Model Details

As pointed out earlier, various methods have been proposed for facilitating domain adaptation, including change of feature representation [Pan and Yang, 2010]. Change of representation is basically a transformation function $g : \mathcal{X} \rightarrow \mathcal{Z}$ such that

$$P(z) = \sum_{\substack{x \in \mathcal{X} \\ g(x)=z}} P(x) \quad \forall z \in \mathcal{Z}. \tag{5.18}$$

Equation 5.18 states that the transformation function should preserve the probability distributions of features. In our model, we have to transform the original raw features into new features. The sensor IDs are unique to each specific environment and might carry different meanings in each environment. For example, id_1 in one

home might refer to an infrared motion sensor in the bedroom for detecting sleep activity. The same sensor ID id_1 in another home might refer to a cabinet sensor in the kitchen. We use the simple mapping $\mathcal{S} \rightarrow L(\mathcal{S})$ where $\mathcal{S}$ refers to the set of sensor IDs, and L refers to the location where a sensor is located (e.g. bedroom, or medicine cabinet). Obviously such a mapping preserves the probability distribution of features, as the activation probability of the sensors will sum up for a location. We maintain a unified list of locations across different environments.

We use an N-gram model to represent the sequential nature of data [Manning and Schtze, 1999]. N-gram models have been used extensively by the natural language processing community to represent the sequential data in a simplified and efficient manner. An N-gram model for our sensor data shows the currently activated sensor, in addition to $N-1$ previously activated sensors, providing history context. We also use the start time of the activations as another feature to provide temporal context.

The source domains $\mathcal{D}_1$ through $\mathcal{D}_m$ provide us with m hypotheses h_1 .. h_m. First we select $n \leq m$ hypotheses based on the overall similarity between our target and each one of the sources. We then combine n hypotheses in order to infer a hypothesis h_t for the target domain.

To combine hypotheses h_1 .. h_n, we weight the hypotheses by their corresponding pairwise activity similarities with the target domain, multiplied by the hypothesis confidence. Here we adopt the smoothness assumption [Belkin et al., 2006], which

states that two nearby (similar) patterns in a high density area (high confidence) should share similar decision values. Therefore we can assign a certain label to a particular activity in the target domain, if the two domains are similar in terms of that particular activity, and also if our confidence is high about the predicted label of that activity. For our domain adaptation problem, we assume that the target function h_t shares values with similar domains.

In order to use $\mathcal{H}$ distance in our activity recognition system, we need to adapt it to our scenario. Not only is it important to measure the overall similarity between source and target environments, but also it's important to know how many similar activities exist and what is their individual similarity value. For example, "eating" and "cooking" can be similar in the two domains (similar structure, similar start times), but "sleeping" might be different. This indicates that our source domain might be a good candidate for recognizing "eating" and "cooking" activities in the target domain, but not for "sleeping" activity.

We denote the distance between two domains $\mathcal{D}_s$ and $\mathcal{D}_t$ based on activity y as $d^y_{\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t)$. It's similar to computing the generic H distance, with the exception all the data points with a label other than y will be ignored during calculation. Note that such a distance measure is not symmetric. Consequently the activity based similarity measure is defined as in Equation 5.19.

$$\Phi^y(s,t) = 1 - d^y_{\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) \tag{5.19}$$

In order to determine similarity between a source and target domain based on individual activities, we need to identify the target activities. As we assume that the target data is unlabeled, we have no way of telling which activity a certain sensor event belongs to. To solve this problem, we utilize a two-stage algorithm. First, we build an ensemble classifier using source datasets to assign a label to the target data points. Next, we will compute the individual activity based source-target similarity, and will select the promising sources. Then we will adjust the target points labels according to the source-target activity similarity.

The base classifier in our algorithms is a kernel based naive Bayes classifier [Pérez et al., 2009]. A naive Bayes classifier is a classifier based on Bayes theorem that works surprisingly well on N-gram data [Hand and Yu, 2001]. The naive Bayes classifier predicates the class of an instance, y^*, as in Equation 5.20.

$$y^* = \max_{y_j} P(y_j) \prod_i P(x_i|y_j) \tag{5.20}$$

In a kernel naive Bayes classifier, the conditional probability $P(x_i|y_j)$ is computed via kernel density estimation of class y_j, as in Equation 5.21. This allows us to estimate the distribution of non-categorical values, such as the start time of activities. Here N is total number of data points.

$$P(x|y_j) = \sum_{i}^{N} K_j(x, x_i) \tag{5.21}$$

In Equation 5.21, $K_j(x, x_i)$ is a kernel function. We used the widely adopted Gaussian function as a kernel (Equation 5.22). Also we used Laplace smoothing to prevent high influence of zero probabilities [Zadrozny and Elkan, 2001].

$$K_j(x, x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-x_i)^2}{2\sigma^2}} \tag{5.22}$$

After finding the individual activity similarities $\Phi^y(s,t)$ as in Equation 5.19, the overall similarity between a given source and target can be defined as in Equation 5.23. Here Y_t shows the subset of all activity labels that appear in the target environment, and $|Y_t|$ refers to the cardinality of Y_t.

$$\Phi(s,t) = \frac{1}{|Y_t|} * \sum_{y \in Y_t} \Phi^y(s,t) \tag{5.23}$$

In the next step, we select the most promising sources. To select the top n sources, given n, we choose n sources with the highest similarity value $\Phi(s,t)$.

If n is not given, then we need to find a suitable value for n. To select the most promising sources out of m sources, we consider the overall source-target similarity $\Phi(s,t)$, in addition to the total number of selected sources. This is reflected in Equation 5.24. The sources with Ψ above a threshold θ_Ψ will be selected.

$$\Psi(s_i) = \frac{\Phi(\mathcal{D}_{s_i}, \mathcal{D}_t)}{\sum_{s=1}^{m} \Phi(\mathcal{D}_s, \mathcal{D}_t)} * \sqrt{1 - \frac{n}{m}} \tag{5.24}$$

The first term in Equation 5.24 limits our choice of source domains to those with a similarity Φ above the average similarity. The second terms imposes a restriction on the number of selected source domains. This allows for a more efficient method, while achieving the desired accuracy.

Next, we combine the hypotheses from n selected sources. We assume that the confidence of a classifier i for a predicted label y is given by $h_i(y|x)$. Therefore, we can write the hypothesis combination rule as in Equation 5.25.

$$h_t(x) = \sum_{i=1}^{n} \arg\max_{y} h_i(y|x) * \Phi^y(s_i, t) \tag{5.25}$$

It should be noted that as we are using an ensemble of source datasets to decide on a target dataset, we should consider the conditions for constructing an ensemble. Dietterich [Dietterich, 2000] points out that an ensemble of classifiers should be diverse and accurate. An accurate classifier is one that has an error rate better than random guessing, i.e. better than 0.5. Two classifiers are diverse if they make different errors on new data points. The diversity condition is reached as we are using sources from different domains. The accuracy condition is achieved by ensuring that an individual source has an accuracy of more than 0.5 when tested on its own dataset in our experiments.

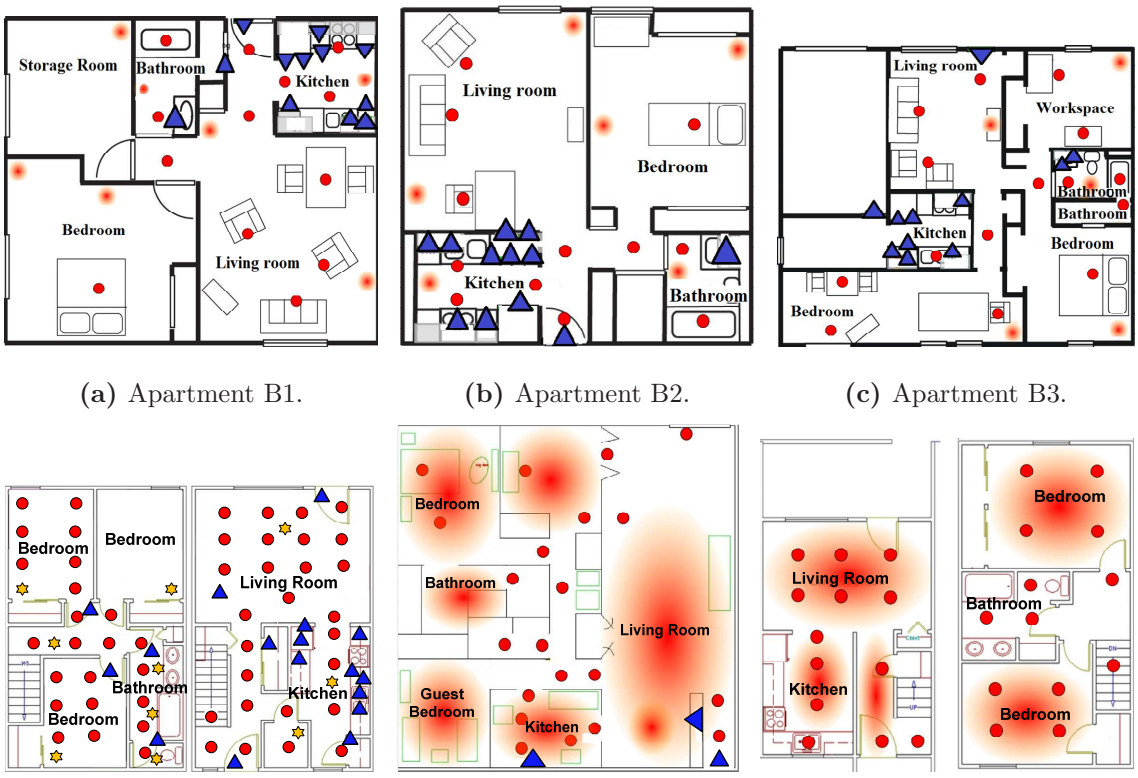
5.3.3 EXPERIMENTS

To evaluate our algorithm, we tested our algorithms on 8 different datasets collected from 6 smart apartments. The layout of the apartments, along with their sensor layouts are shown in Figure 5.11. We refer to those 8 datasets as B1, B2, B3, K1, K2, M, T1 and T2. Datasets K1, K2 were collected from the same apartment, but within different time periods, housing different pairs of residents, and were annotated with different activities. The same is true for T1 and T2. It should be noted that all datasets are collected during a normal day to day life of the residents. A more detailed description of datasets can be found in Table 5.1.

Dataset	**B1**	**B2**	**B3**	**K1**	**K2**	**M**	**T1**	**T2**
Environment	B1	B2	B3	K	K	M	T	T
Num. of Residents	1	1	1	2	3	1+ pet	2	2
Num. of Sensors	32	32	32	71	72	32	20	20
Num. of Instances	5714	4320	3361	497	844	1513	1431	166
Dataset Size (Days)	126	234	177	61	38	88	120	10

Table 5.1: Characteristics of each dataset.

The sensors in our smart environments consist of various ambient sensors, such



(a) Apartment B1. **(b)** Apartment B2. **(c)** Apartment B3.

(d) Apartment K. **(e)** Apartment M. **(f)** Apartment T.

Figure 5.11: Apartment layouts. Circle: motions sensor/area sensor. Triangle: item sensor/door sensor. Star: light sensor.

as infrared motion sensors, oven sensors, switch contacts on doors and cabinets, and light sensors. The data has been collected using our publish/subscribe middleware and is stored in a SQL database.

To be able to evaluate the results of our algorithms based on a ground truth, we annotated the datasets with activities of interest. We considered 10 different

activities. However, as one would expect in a real world setting, not all activities are common between different environments. The list of activities for each dataset can be viewed in Table 5.2. One should note that not only the activities are different in the different datasets, but also their distributions are different.

	B1	B2	B3	K1	K2	M	T1	T2
Hygiene	✓	✓	✓	✓	✓	✓		✓
Leave Home	✓	✓	✓		✓		✓	✓
Cook	✓	✓	✓	✓	✓	✓	✓	✓
Relax	✓	✓	✓		✓	✓	✓	✓
Take Med	✓	✓	✓					
Eat	✓	✓	✓		✓		✓	✓
Sleep	✓	✓	✓	✓	✓	✓		✓
Bathing	✓	✓	✓	✓	✓			✓
Bed to toilet	✓	✓	✓		✓	✓		✓
Work			✓	✓	✓	✓		✓

Table 5.2: Annotated activities in each dataset.

To test our algorithm, we considered 8 different problem settings. In each set-

ting, 7 datasets were considered as source environments, while the remaining dataset was considered as the target environment.

After running our algorithm in a 10 fold cross validation manner, we found the midpoint threshold $\theta_\Psi = 0.5$ to be a suitable value for our experiments. The cross validation results also found the 3-Gram model to best represent the data, and higher gram values did not significantly change our results. The target dataset included 3 days of unlabeled data. The samples from sources datasets were also chosen to include approximately 3 days of data. The unified list of locations included bathroom, shower, bedroom, kitchen, living room, work area, med cabinet, and entrance. All experiments were carried out in a 10 fold cross validation manner.

Figure 5.12 shows the average individual activity similarities versus their recognition rates in a target environment M. It can be clearly seen that the activity recognition rate closely follows the activity similarity.

Figure 5.13 shows source-target activity similarities in various settings. Note that the results confirm our intuition about the dataset similarities. For example, one can clearly see that the most similar dataset for apartment B2 is B1, as they have a similar number of residents and contain common activities. Their floorplan is also quite similar, as they were located in the same building complex. The next most similar dataset is B3 which again has a very similar list of activities. As it is located in the same building complex, it also has a similar floorplan. However

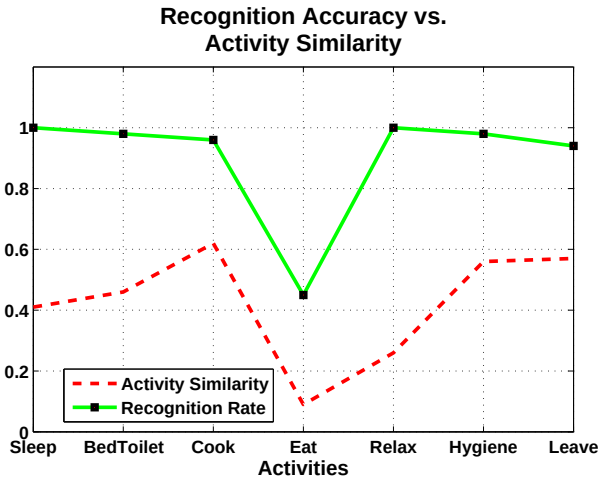


Figure 5.12: Activity similarity vs. recognition rate for dataset M.

it is a two bedroom apartment, having a workspace in the second bedroom and an additional work activity. For datasets T1 and T2, though they were collected from the same apartment, however they have different set of activities, therefore showing lower similarities. These results show how various latent factors can be important for finding similarity between two environments, and how our algorithm is able to infer such a similarity value using its knowledge about the way that the activities are performed in different environments.

Figure 5.14 shows the overall activity recognition rates based on choosing the top n sources. We also performed experiments based on randomly selecting the top n sources (averaged over 10 runs), using a simple linear combination rule (no weights).

(a) Apartment B1.

(b) Apartment B2.

(c) Apartment B3.

(d) Apartment K1.

(e) Apartment K2.

(f) Apartment M.

(g) Apartment T1.

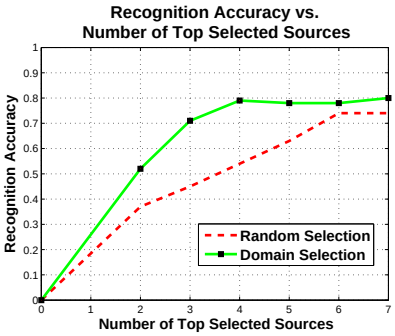
(h) Apartment T2.

Figure 5.13: Pairwise similarity between various source and each target.

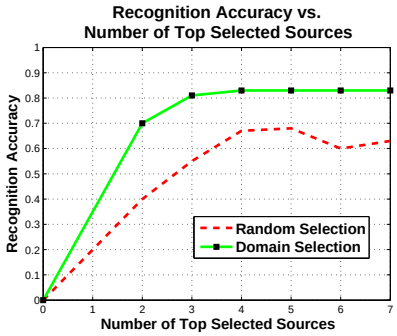
One can clearly see that our domain selection algorithm outperforms the random selection algorithm. It should be noted that the results of our algorithm are based on using only a small sample of source and target datasets and it is possible that the chosen samples are not representative of the entire dataset, leading to lower recognition rates in some cases. Still, despite the fact that we are relying only on a small unlabeled target dataset, and despite the fact that the apartment layouts, sensors and residents are different, we were still able to achieve very reasonable recognition rates.

In Figure 5.14 one can also see the effect of negative transfer. We can see that adding more data sources does not necessarily increase the recognition accuracy in the target environment. We used our source selection algorithm for choosing the best number of sources. Using our method, 95% of the maximum achievable accuracy was achieved using only 4.5 sources on average. The average accuracy was 74.88%. This shows how our algorithm can approximate the best number of promising sources, balancing efficiency and accuracy.

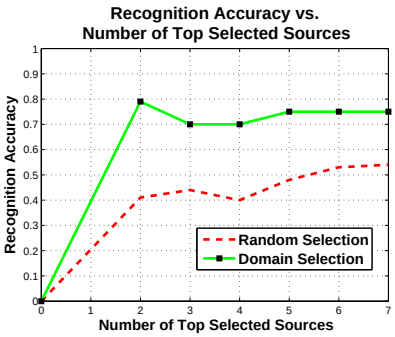
The detailed accuracies based on choosing the best number of sources are shown in Table 5.3. Table 5.3 shows accuracy results for a similar supervised method that uses 100% of the "labeled" target dataset for training. In contrast our method uses about 8.7% of the target dataset on average. It can be seen that though our method uses only a small fraction of target dataset which is also unlabeled, it surprisingly



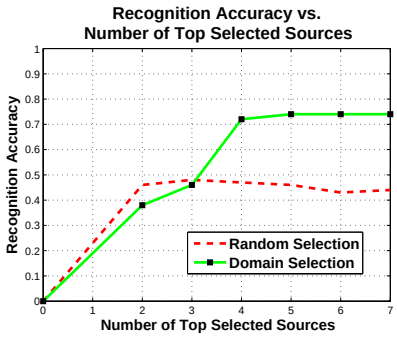
(a) Apartment B1.



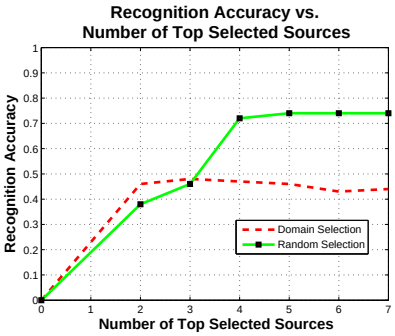
(b) Apartment B2.



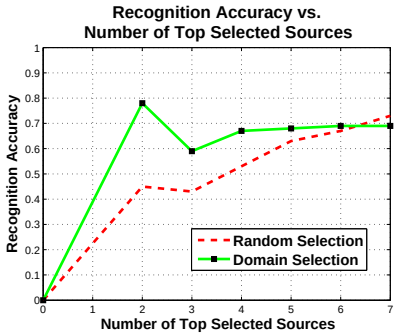
(c) Apartment B3.



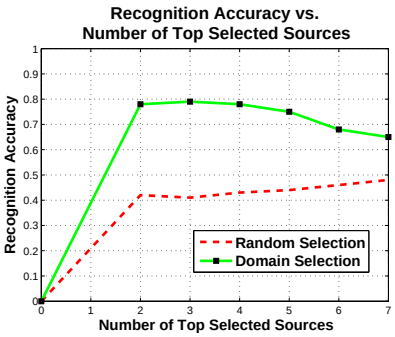
(d) Apartment K1.



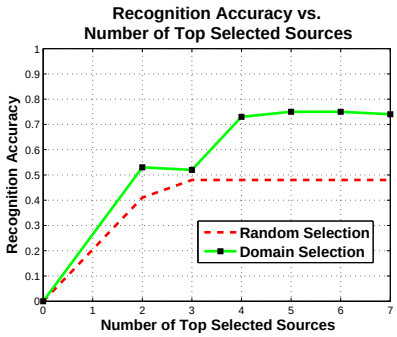
(e) Apartment K2.



(f) Apartment M.



(g) Apartment T1.



(h) Apartment T2.

Figure 5.14: Activity recognition accuracies for different numbers of source domains.

works well. In some cases, such as for the K2 dataset, our algorithm even outperforms the supervised method. These experiments show how our method can be useful in a real world situation where supervised methods are expensive and impractical due to the time constraints and annotation costs. One can also combine our bootstrap method with active learning methods to achieve even higher accuracy in a target environment.

Dataset	**B1**	**B2**	**B3**	**K1**	**K2**	**M**	**T1**	**T2**	**Avg.**
Supervised.	0.86	0.86	0.9	0.63	0.79	0.73	0.88	0.76	0.80
Domain Selection	0.79	0.83	0.75	0.69	0.74	0.68	0.78	0.73	0.75

Table 5.3: Domain selection recognition rates.

5.4 Summary

In this chapter, we introduced a number of methods based on transfer learning concepts. These methods can be used to transfer activity models between different residents as well as between different physical spaces. We also described a method for selecting the most promising sources among a number of available sources. Those activity transfer methods allow us to reuse activity models in another domain, therefore eliminating or reducing the annotation time and effort and expediting the deployment process.

In the next chapter, we will introduce two novel active learning methods which can be used to optimize the annotation time and effort, by posing targeted queries to a human annotator.

CHAPTER 6. ACTIVE LEARNING

Judge a man by his questions rather than by his answers.

— *Voltaire*

In this chapter, we introduce two novel active learning methods to reduce the annotation time and effort. We test our algorithms on data collected from our smart apartments. In addition to CASAS datasets, we also test these algorithms on several real world datasets from the UCI repository to show how our method can be used in general.

6.1 Introduction

In recent years, a variety of active learning methods have been proposed [Settles, 2009, Tomanek and Olsson, 2009] and it has been used in various application domains such as drug discovery [Warmuth et al., 2003], text classification [McCallum and Nigam, 1998], media retrieval [Chen et al., 2005b] and medical image classification [Hoi et al., 2006]. Active learning is primarily used when little labeled data is available, but unlabeled data is abundant. Its goal is to minimize the human annotation efforts via posing targeted queries to an oracle, instead of labeling the whole dataset. Active

learning methods usually select an informative unlabeled instance and ask the oracle for the label of the instance. The oracle, typically a human oracle, provides the correct label of the instance and then the newly labeled instance is added to the training dataset.

Despite enormous progress in the active learning field in recent years, there are still some shortcomings that need to be addressed. Traditional active learning methods usually ask for the label of a specific unlabeled instance. Though this might result in some accuracy improvement, it might not be very easy for an oracle to label a very specific case. This can be especially true if the query contains many features, and if those features represent high precision numeric data. Some features might also be irrelevant for a certain query and eliminating those features can result in a shorter and more readable query. This can also prevent the oracle's confusion.

For example in the real world, a medical domain expert prefers to be presented with a generic diabetes query which (1) embodies similar patient cases together, (2) only includes relevant symptoms and (3) involves range values instead of very specific exact values for the lab test results. Such a query will be shorter, less confusing and more intuitive. In addition, the domain expert will be able to answer more queries in less time and the learning algorithm can achieve higher accuracy rates by posing fewer queries.

We present two novel methods for constructing generic active learning queries.

The first method is based on feature selection while the second method called is based on rule induction. We refer to our first method as "template based query active learning". We call our second method RIQY, standing for **R**ule **I**nduced active learning **Q**uer**Y** method. Both methods follow the same steps for choosing the most informative instance and its similar cases by exploiting the underlying distribution. Their difference is how they construct a general query from the most informative instance and its similar cases. In both cases, the generic query not only is shorter and more readable, but it also condenses multiple similar cases into a single query.

In our template based method, we select an informative instance as well as its nearest neighbors (most similar) and farthest enemies (most dissimilar). By finding the set of relevant features that separate the enemies and neighbors of an informative instance, we are able to construct a template query. The template query is composed of relevant features which can takes on point and range values. We test it on 6 real world datasets from the UCI repository, as well as several CASAS datasets.

Our RIQY method again exploits the underlying density distribution to find the most informative instance as well as its most similar cases. Then by using a rule induction classifier to infer rules for separating those similar cases from the rest of data, we construct a generic query. We again evaluate our algorithm on two different sets of data. The first set of data consists of various real world datasets from the UCI repository [Frank and Asuncion, 2010], as well as several CASAS datasets. Our RIQY

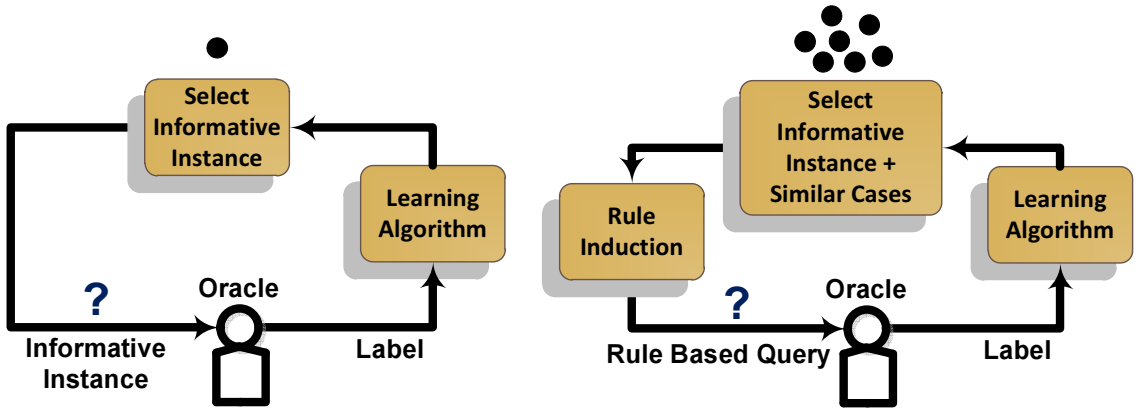
method is also relatively easy to implement as it employs the well known machine learning components.

The remainder of this chapter is organized as follows. First we will present a motivational example in Section 6.2, followed by an overview in Section 6.3. In Section 6.4, we explain our approach for selecting the most informative instance. Then we explain the query construction details of template based method in Section 6.5, and the RIQY method in Section 6.6. We then present the results of our experiments in Section 6.7.

6.2 Motivational Example

We show by an example how our methods can pose shorter and more meaningful queries to the oracle and how such queries can aggregate multiple similar cases together. Figure 6.1 also better highlights the differences between a traditional active learning method versus one of our methods, RIQY. Note that our template based method also follows a similar process, only instead of rule induction it uses a heuristic feature selection method.

For example consider a heart disease dataset. We assume that it has 20+ features and the classification task is to predict whether a patient has heart disease or not. Consider the following example query:



(a) Traditional active learning. **(b)** Our RIQY active learning method.

Figure 6.1: Traditional active learning methods versus our RIQY active learning method.

"*What is the class label if (sex= female) and (age =39) and (chest pain type = 3) and (serum cholesterol = 150.2 mg/dL) and (fasting blood sugar = 150 mg/dL) ... and (electrocardiographic result = 1) and (maximum heart rate achieved = 126) and (exercise induced angina = 90) and (heart old peak = 2.3) and (number of major vessels colored by fluoroscopy = 3)?*"

Note that for the sake of brevity here we only show a sample of the total set of features. In reality all of the features would be included in the query that is presented to the oracle. From this example query, one can clearly see how such a long and overly specific query with so many features can be difficult to be answered by a domain expert. A similar case would also require another query to be posed to the oracle, without taking advantage of the previous cases. In addition, many features

might be indeed irrelevant to the query at hand.

Although one can apply a feature selection method as a preprocessing step here, but it should be noted that it would discard the features in a global manner. In contrast, we are looking at the problem of discarding a "locally" irrelevant feature for a number of similar instances. For example, "exercise induced angina" feature might be relevant for the whole dataset, but it might not very discriminating for a group of patients whose "age > 65".

In the real world, a domain expert usually expects queries in a shorter and more intuitive form, with range values instead of exact values and with similar cases aggregated together as a generic case. One such example is:

"*What is the class label if (age* > 65*) and (chest pain type = 3) and (serum cholesterol* > 240 *mg/dL) ?*"

In the following sections, we show how we can construct such generic queries as a form of rule induction.

6.3 Preliminary

Our input data is an n-dimensional feature vector which is denoted by $x = \langle x_1, x_2, \cdots, x_n \rangle$. Each labeled instance x is assigned a class $y_j \in y$. The function that measures the informativeness of an instance x is denoted by $\Phi(x)$ and the most

informative instance is denoted by x^*. We assume that each dataset is composed of a small labeled dataset $\mathcal{L}$, a large unlabeled dataset $\mathcal{U}$ and a test dataset $\mathcal{T}$ which is set aside for the evaluation purposes only.

We also assume that the oracle is able to answer our queries and to provide us with a label l and a confidence value c. There is a proximity matrix M that shows the similarity between each two data points in our dataset. The proximity matrix is used to identify the nearest neighbors of x^*. All the proximities are cached and pre-computed as mixed Euclidean distances. For a more efficient nearest neighbor search, one can employ methods such as kd-tree [Panigrahy, 2008] or locality sensitive hashing (LSH) [Gionis et al., 1999]. If the original dataset contains a large number of features, one can perform conventional feature selection methods to obtain a more representative dataset. But as we mentioned before, a preprocessing feature selection step is quite different than the local feature selection process during rule induction, as the locally relevant features can vary from query to query.

In summary our methods work as following. First we train a classifier $\mathcal{C}$ on a small labeled dataset $\mathcal{L}$ to identify the potential informative instances. This step is similar to most of the traditional active learning methods. Our informativeness measure is a variation of density weighted method that also takes into account the dissimilarity to the previously labeled data in order to achieve a more efficient method.

After identifying the most informative instance, we select its nearest neighbors

as well as its enemies. The nearest neighbors and the enemies are combined together into a single dataset. The nearest neighbors are assigned a label of 0, while the enemies are assigned a label of 1, regardless of their original labels. By separating the nearest neighbors from the enemies using either a rule induction classifier (RIQY) or feature selection (template based), we can obtain the essential discriminating features. Next, we present the induced rules to the oracle and update the labeled dataset. This is repeated until a maximum number of queries is posed or until a specified accuracy improvement is achieved.

Figure 6.2 shows the main components of our RIQY method. The template based method is similar, except that the rule induction step is replaced by heuristic feature selection.

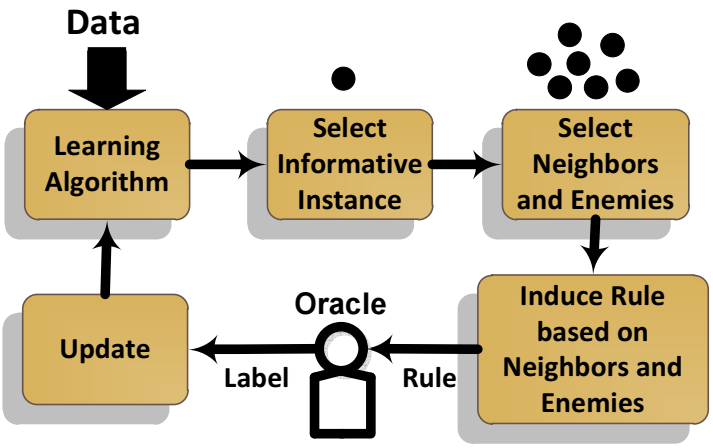


Figure 6.2: The main components of our method.

6.4 Choosing An Informative Instance

The first step in our method is to select the most informative instance x^* among the pool of unlabeled instances. As mentioned before, we use a variation of a density weighted method to measure the informativeness of an instance x. Our measure considers the similarity of x to the other unlabeled instances, as well as its dissimilarity to the previously labeled instances. Considering similarity to the other unlabeled instances allows us to select an instance that is representative of many other similar cases. Considering its dissimilarity to the previously labeled instances allows us to avoid querying similar instances repetitively, thus leading to a more effective method. Equation 6.1 shows our method for selecting the most informative instance x^*.

$$x^* = \arg\max_x \left[(1-\alpha) * \Phi(x) + \alpha * \frac{|\mathcal{L}| * \sum_{u=1}^{|\mathcal{U}|} M[x, x_u]}{|\mathcal{U}| * \sum_{l=1}^{|\mathcal{L}|} M[x, x_l]} \right] \tag{6.1}$$

In Equation 6.1, parameter α balances the contribution of density versus the base informativeness measure $\Phi(x)$. Here the second term shows the contribution of density with respect to both labeled and unlabeled data. The first term, $\Phi(x)$, refers to a base informativeness measure. The base informativeness measure here is an entropy measure as in Equation 6.2. This measure can be replaced by any other base informativeness measure. Here y_i ranges over all possible label values.

$$\Phi(x) = -\sum_{i} P_\theta(y_i|x) \log P_\theta(y_i|x) \tag{6.2}$$

After an informative instance has been selected, a generic query is constructed based on the informative instance and its similar instances.

6.5 Template Based Active Learning

After selecting the most informative instance, we construct a template query. Algorithm 4 shows further details of our model. The stopping criteria can be considered as either reaching a user defined number of iterations or reaching a specific accuracy improvement.

Lines 3-4 build a classifier $\mathcal{C}$ using $\mathcal{L}$. The classifier is then used to classify the unlabeled instances in $\mathcal{U}$. Line 5 finds the informative instance x^* using Equation 6.1. Line 6 computes the nearest neighbors and farthest enemies of x^* using the proximity matrix M. We consider the neighborhood size as a fixed fraction ϵ of the unlabeled dataset size. The unlabeled dataset gets smaller over time, therefore the neighborhood size shrinks over time which helps us to get more focused queries as more labeled data is provided. Line 7 computes features' relevancy based on classifying neighbors and enemies of x^*. To emphasis the fact that the neighbors are assigned a positive label and the enemies are assigned a negative label, we denote them by $\mathcal{N}^+$ and $\mathcal{E}^-$. Line 8 forms the template query as outlined in Algorithm 5. After posing the query to the

Algorithm 4 Template Based Active Learning

1: **procedure** FINDTEMPLATEQUERY($\mathcal{L}$, $\mathcal{U}$, M)
2: **while** stopping criteria is not met **do**
3: Train a classifier $\mathcal{C}$ using $\mathcal{L}$
4: Use $\mathcal{C}$ to compute $p(y|x)$ $\forall x \in \mathcal{U}$
5: Find x^* using Equation 6.1
6: Find $\mathcal{N}$ and $\mathcal{E}$ for x^*
7: r = features' relevance based on $\mathcal{N}^+ \bigcup \mathcal{E}^-$
8: $c, l \leftarrow$ formQuery(f, r, $\mathcal{N}$)
9: **for all** $x_j \in \mathcal{N}$ **do**
10: Update w_j as in Equation 6.3
11: **if** $w_j > \theta$ **then**
12: Add x_j to $\mathcal{L}$ with label l
13: **end if**
14: **end for**
15: **end while**
16: **end procedure**

oracle, lines 9-14 update the neighbors' weight w according to Equation 6.3.

$$w_j = c * \frac{1}{|\mathcal{N}|} \sum_{k}^{|\mathcal{N}|} M\left[j,k\right] \tag{6.3}$$

An instance's weight increases proportionately with its similarity to the query neighborhood and the oracle's confidence about query's label. At the end, the unlabeled instances with a weight above the threshold θ are added to the training dataset $\mathcal{L}$.

When forming a query Q, we adopt an approach similar to the one proposed by [Du and Ling, 2010] in which we assign each feature f_i to one of the three feature relevancy categories depending on its relevance weight r_i. The feature can be highly relevant, weakly relevant or irrelevant as in Equation 6.4.

$$f_i \text{ is } \begin{cases} \text{Highly Relevant} & \text{if } r_i \in [r_n + 2d, r_x] \\ \text{Weakly Relevant} & \text{if } r_i \in [r_n + d, r_n + 2d) \\ \text{Irrelevant} & \text{otherwise} \end{cases} \tag{6.4}$$

Here r_n and r_x denote the minimum and maximum relevance values and d denotes $(r_x - r_n)/3$.

After assigning each feature to its proper relevance category, we need to decide about the feature values in our template query Q. In Algorithm 5, lines 3-12 compute the feature values and augment Q with the assigned feature values. For highly relevant

Algorithm 5 Form Template Query

1: **procedure** FORMQUERY(f, r, $\mathcal{N}$)
2: $Q = \emptyset$
3: **for all** $f_i \in f$ **do**
4: **if** f_i is highly relevant **then**
5: set f_i as in Equation 6.5
6: $Q = Q \bigcup f_i$
7: **end if**
8: **if** f_i is weakly relevant **then**
9: set f_i to its range in $\mathcal{N}$
10: $Q = Q \bigcup f_i$
11: **end if**
12: **end for**
13: c, $l \leftarrow$ askQuery(Q)
14: **end procedure**

features, we will set the value to one specific value. Weakly relevant features can be generalized with several values. The irrelevant features can be generalized with any values (can be displayed as $*$ or don't care values), therefore we simply omit them from our queries. For highly relevant features if f_i is a numeric feature, then its value is set to the mean value of $\mathcal{N}$, otherwise it is set to its mode (see Equation 6.5).

$$f_i = \begin{cases} \frac{1}{|\mathcal{N}|} \sum_{x \in \mathcal{N}} f_i & \text{if } f_i \text{ numeric} \\ mode(f_i) & \text{if } f_i \text{ nominal} \end{cases} \tag{6.5}$$

For weakly relevant features, we use the actual range of values according to $\mathcal{N}$. The template query is the union of highly relevant and weakly relevant features. Note that we simply discard the irrelevant features from our query. After constructing template query Q, it is posed to the oracle to obtain a label l and a confidence value c. Then as described in Algorithm 4, the unlabeled neighbor instances are updated and are moved to the labeled dataset $\mathcal{L}$.

6.6 RIQY

In this section we explain our RIQY method in more detail. Similar to the template based method, after an informative instance has been selected, a RIQY generic query is constructed based on the informative instance and its similar instances. Al-

gorithm 6 outlines further details of our RIQY method.

Lines $3-4$ trains a classifier $\mathcal{C}$ based on the available training data $\mathcal{L}$, and computes the class probabilities for each unlabeled instance $x \in \mathcal{U}$. In line 5, based on the obtained class probabilities and the underlying distribution, we select the most informative instance x^* according to Equation 6.1.

Next, we find x^* similar instances and form a generic query based on those instances. We first select the nearest neighbors of x^* in line 6. The size of the neighborhood (or the enemy vicinity) is a fixed fraction ϵ of the unlabeled dataset $\mathcal{U}$. In other words, $|\mathcal{N}_{x^*}| = |\mathcal{U}| * \epsilon$. To get the enemies set $\mathcal{E}_{x^*}$, we sample from $\mathcal{U} - \mathcal{N}_{x^*}$. The size of the enemies set is the same as the neighbors set, i.e. $|\mathcal{E}_{x^*}| = |\mathcal{U}| * \epsilon$. We chose them to be of the same size to avoid any class imbalance problems. As over time the unlabeled dataset gets smaller and as more labeled data becomes available, the neighborhood size as well as the enemy vicinity becomes smaller. This allows us to pose more focused queries over time.

Next in line 8, we combine the nearest neighbors $\mathcal{N}_{x^*}$ and the enemies $\mathcal{E}_{x^*}$ into a single set Δ. Each instance of the nearest neighbors is assigned a label of 0 in Δ, while each instance belonging to the enemies is assigned a label of 1 in Δ, regardless of their original labels. We denote the re-labeled sets by $\mathcal{N}^+_{x^*}$ and $\mathcal{E}^-_{x^*}$.

Then a rule induction classifier such as C4.5 [Quinlan, 1993] is used to generate a number of rules $\mathcal{R}$ from Δ in line 9. Through separation of the nearest neighbors

Algorithm 6 RIQY Active Learning method

procedure FINDRULEBASEDQUERY($\mathcal{L}$, $\mathcal{U}$, M)
  while stopping criteria is not met **do**
    Train a classifier $\mathcal{C}$ using $\mathcal{L}$
    Use $\mathcal{C}$ to compute $p(y|x)$ $\forall x \in \mathcal{U}$
    Find x^* using Equation 6.1
    Find $\mathcal{N}_{x^*}$
    Sample $\{\mathcal{U} - \mathcal{N}_{x^*}\}$ to get $\mathcal{E}_{x^*}$
    $\Delta = \mathcal{N}^+_{x^*} \bigcup \mathcal{E}^-_{x^*}$
    Form rule set $\mathcal{R}$ from Δ
    Select $\acute{\mathcal{R}}$ from $\mathcal{R}$ based on γ and a
    $c, l \leftarrow$ askOracle($\acute{\mathcal{R}}$)
    for all $x_j \in \Delta$ **do**
      if x_j is covered by some $r_k \in \acute{\mathcal{R}}$ **then**
        Update w_j as in Equation 6.6
        if $w_j > \theta$ **then**
          Add x_j to $\mathcal{L}$ with label l_k
        end if
      end if
    end for
  end while
end procedure

of x^* from its enemies by using a rule induction classifier, we can obtain the essential features for discriminating x^* and its similar cases from the rest of data in form of rules. Note that each rule $r \in \mathcal{R}$ is accompanied by its coverage value γ and its accuracy a. The coverage value shows how many instances are covered by a certain rule and the accuracy shows the discriminative power of a rule for distinguishing between positive and negative instances. We select the rules with a minimum accuracy $a_{\min}$ in order to avoid adding the incorrect instances. The rules are then sorted according to their coverage values and the top N rules with the highest coverage value are selected as $\acute{\mathcal{R}}$ in line 10. Note that normally we set N to 1 to present one rule a time to the oracle. But it is also possible to present several rules at a time in a batch mode.

Line 11 poses the rule set $\acute{\mathcal{R}}$ to the oracle and receives the corresponding labels l_i and confidences c_i for each rule $r_i \in \acute{\mathcal{R}}$. Lines 12-19 update each unlabeled instance $x_j \in \Delta$ that is covered by some rule $r_k \in \acute{\mathcal{R}}$, according to Equation 6.6.

$$w_j = c_k * \frac{1}{|X_{r_k}|} \sum_{k}^{|X_{r_k}|} M\left[j,k\right] \tag{6.6}$$

Here w_j shows the weight of instance x_j in our dataset. The set of all instances that are covered by a certain rule r_k is denoted by X_{r_k}. The instance's weight increases proportionately with its similarity to the query neighborhood and the oracle's confidence about query's label.

At the end, those instances covered by some rule and with a weight above the weight threshold θ are added to the labeled dataset $\mathcal{L}$. This process continues until we reach the stopping criteria. The stopping criteria can be either reaching a maximum number of queries or reaching a certain minimum classification accuracy improvement.

6.7 Experiments

In the following subsections, we show the results of our RIQY and template based algorithms on two different sets of data.

6.7.1 Template Based Method Experiments

In order to evaluate our algorithm, we tested our algorithm on a number of real world datasets. We used 6 datastes from the UCI repository [Frank and Asuncion, 2010] and 6 datasets from CASAS repository. The UCI datasets include the Australian credit approval dataset, ionosphere dataset, wine dataset, tic tac toe dataset, dermatology dataset and mines vs. rocks dataset. The dermatology and wine datasets are multi-class problems, while the rest of the datasets are binary classification problems.

The CASAS datasets include B3, C, K4, M, T1 and T2 datasets. The annotated

activities in those datasets are shown in Table 6.1.

	B3	C	K4	M	T1	T2
Hygiene	✓		✓	✓		
Leave Home	✓	✓	✓	✓	✓	✓
Cook	✓		✓	✓	✓	✓
Relax	✓		✓	✓	✓	✓
Take Med	✓	✓		✓		
Eat	✓	✓	✓	✓	✓	✓
Sleep	✓	✓	✓	✓		
Bathing	✓		✓			
Bed to toilet	✓	✓	✓	✓		
Work	✓	✓	✓	✓		

Table 6.1: The annotated activities in each dataset.

All the activity datasets represent multi-class classification problems. Each activity instance is represented by features composed of sensor locations and activity duration. Further information about these datasets is shown in Table 6.2.

For each dataset, we split the dataset into 3 disjoint datasets: a training dataset

Dataset	Num. of Features	Num. of Activities	Num. of Examples
B3	15	10	3361
C	10	6	511
K4	14	9	746
M	13	9	2270
T1	10	4	1431
T2	11	4	163

Table 6.2: CASAS datasets.

$\mathcal{L}$ ($1\% - 2\%$ of data), a test dataset $\mathcal{T}$ (25% of data) and a validation dataset $\mathcal{U}$ (the rest of the data). The feature selection method selects the individual features using information gain criteria. Based on our cross validation results, we found the following values for our parameters: $\theta = 0.5$, $\alpha = 0.5$, $\epsilon = 0.2$ for the UCI datasets and $\epsilon = 0.01$ for CASAS datasets.

In order to simulate the probability estimation of a human oracle based on a template query, we consider the instances that are covered by the query. The label of the query is then the majority label and the confidence is computed as the proportion of the majority instances. It should be noted that the datasets might contain noisy

instances which can be confusing for the oracle and can occasionally lead to a lower confidence.

As an example of our template queries, we chose a sample query based on the wine dataset. The query is as following:

"*What is the alcohol type if the color-intensity is [3.52 .. 4.70] and hue is 1 in average and OD280 is [3.59 .. 4.00] ?*"

Note that the number of features is reduced by 76% as a result of aggregating 17 instances. This can possibly account for 17 potential queries in traditional active learning methods. This simple example clearly shows how our approach can lead to shorter and more intuitive queries, while at the same time aggregating many similar instances together into a single query.

Figure 6.3 shows the average accuracy of our method on UCI datasets and Figure 6.4 shows the average accuracy of our method on CASAS datasets. We compare our method to a traditional active learning method based on uncertainty sampling. From those results, we can clearly see that our method outperforms the traditional uncertainty sampling method. Its performance can be attributed to its use of density distribution information and avoiding querying the outliers. It should be noted that the accuracy of a generic active learning method can depend on some other factors too. For example, if a dataset has mostly nominal values such as the credit approval dataset, then applying our template based active learning method might not lead to

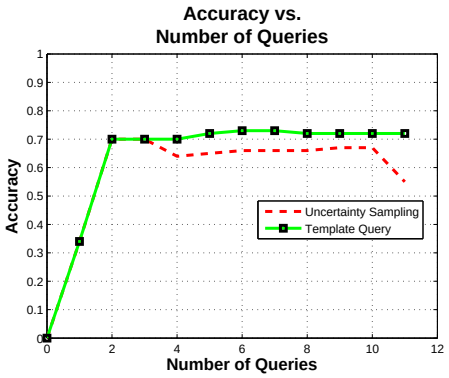
significant improvement over the traditional pool based method. The reason is that in such datasets the range of values is quite limited, therefore the queries can not be generalized beyond a certain limit.

Dataset	Features Reduction	Oracle Confidence	Added Examples
Credit	83.0%	74.9%	55.8
Ionosphere	64.7%	71.5%	22.5
Wine	53.8%	88.2%	11.3
Dermatology	61.7%	84.7%	23
Tic Tac	81.1%%	95.3%	63.7
Mines	63.3%	75.5%	12.2

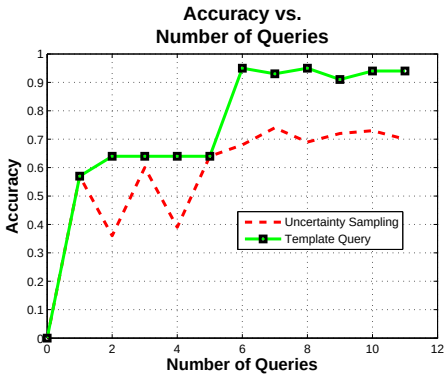
Table 6.3: Some statistics based on template based active learning.

Table 6.3 shows the average feature reduction, average oracle's confidence and average number of added instances to the labeled dataset for each dataset. For a traditional active learning method the number of added instances is always 1. For random synthesis method, this number is always fixed (it is user defined, here we set it as the initial $\epsilon * |\mathcal{U}|$). From those results and specially from the feature reduction results, we can clearly see how our method results in shorter queries.

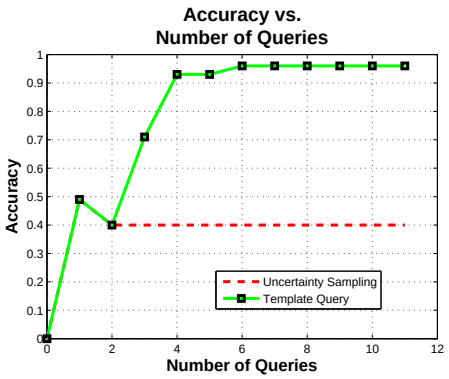
Figure 6.5 shows the number of added instances over 10 queries for an example



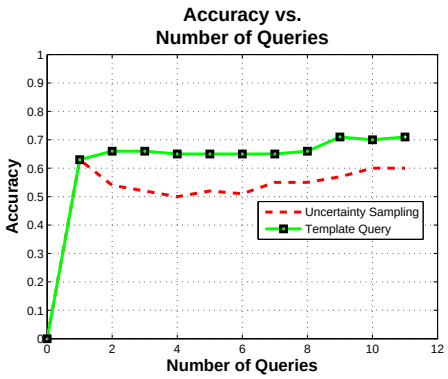
(a) Australian credit approval dataset.



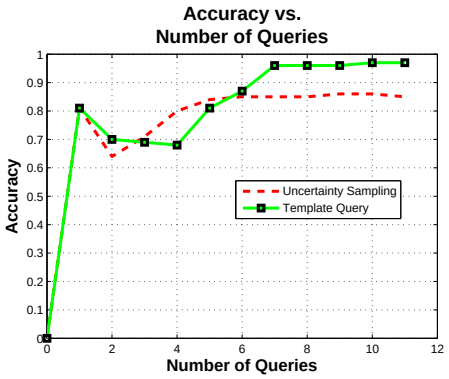
(b) Ionosphere dataset.



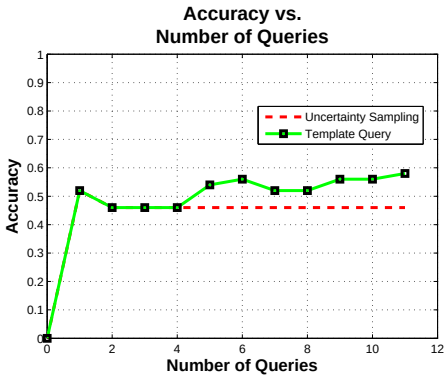
(c) Wine dataset.



(d) Tic Tac Toe dataset.

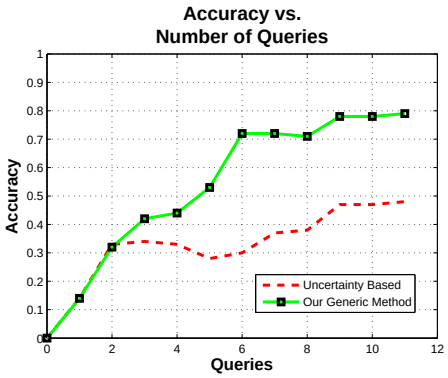


(e) Dermatology dataset.

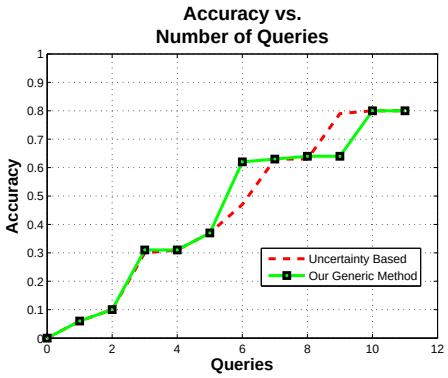


(f) Mines vs. Rocks dataset.

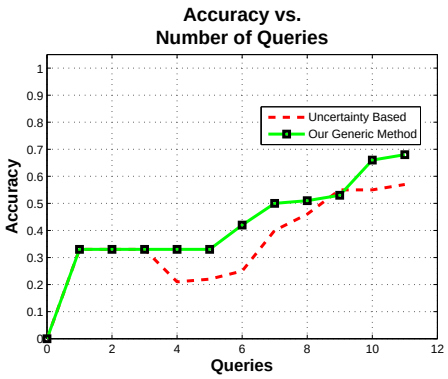
Figure 6.3: Accuracy for different UCI datasets vs. number of queries.



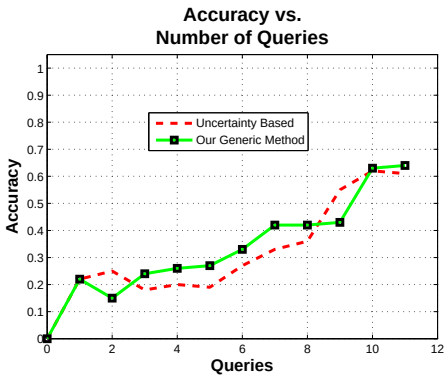
(a) B3 dataset.



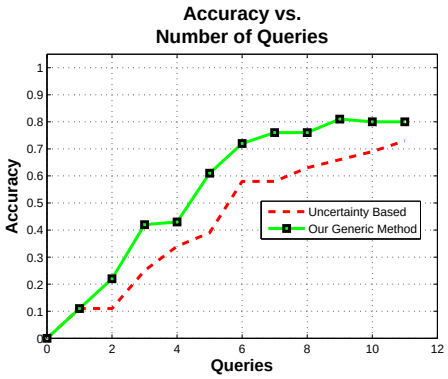
(b) C dataset.



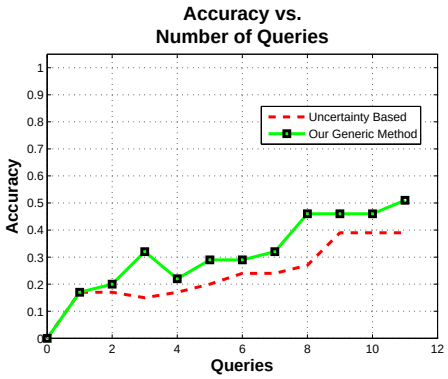
(c) K4 dataset.



(d) M dataset.



(e) T1 dataset.



(f) T2 dataset.

Figure 6.4: Accuracy for different CASAS datasets vs. number of queries.

dataset (wine dataset). As mentioned before, the neighborhood size is shrank over time to get more focused queries. It should be noted that not all instances in the neighborhood will be added to the dataset. Depending on the oracle's confidence and the similarity of the instance to the rest of instances, an instance might or might not be added to the labeled dataset. Depending on the size of dataset, the number of added instances will be different.

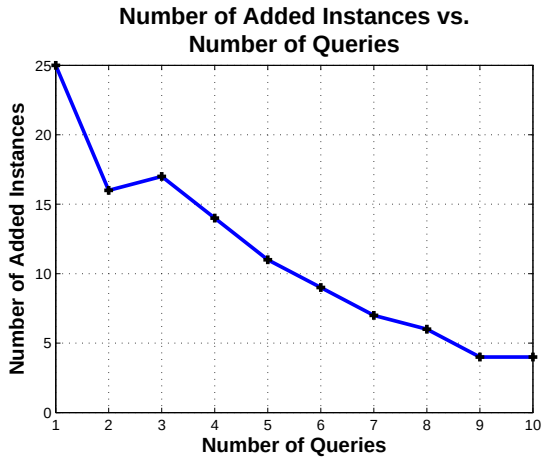


Figure 6.5: Number of added instances over 10 queries for the wine dataset.

Overall our results demonstrate how can we successfully construct a template query by exploiting the underlying density distribution and the similarity between a group of unlabeled instances. It also shows how our method can construct shorter and more intuitive queries, while also leading to higher accuracies.

6.7.2 RIQY Method Experiments

We perform a number of experiments on two sets of real world data in order to evaluate our RIQY active learning algorithm. The data in our experiments consists of 6 real world datasets from the the UCI machine learning repository [Frank and Asuncion, 2010] and 6 human activity recognition datasets from the CASAS repository.

The first set of data includes 6 real world datasets from the UCI repository [Frank and Asuncion, 2010]. These datasets include the Germany credit approval dataset, the Wisconsin breast cancer dataset, the heart disease dataset, the wine dataset, the ionosphere dataset and the chess dataset. All the datasets are binary classification problems, except for the wine dataset. Further details of those datasets is shown in Table 6.4.

The activity recognition datasets consists of data collected from 6 different smart apartments, similar to our experiments in previous section. We again refer to those datasets as B3, C, K4, M, T1 and T2.

We split each dataset into three disjoint parts: a small labeled dataset $\mathcal{L}$ (about 1%-2% of data), a test dataset $\mathcal{T}$ (about 25% of data) and an unlabeled dataset $\mathcal{U}$ (the rest of data). All the results are averaged over 3 runs in order to reduce the experiment variation. The initial classification step is performed using a support vector machine

Dataset	Num. of Features	Attributes Type	Num. of Examples
Credit approval	20	Numeric-Nominal	1000
Ionosphere	34	Numeric	351
Wine	13	Numeric	178
Heart Disease	13	Numeric-Nominal	270
Breast Cancer	9	Nominal	699
Chess	36	Nominal	3196

Table 6.4: UCI datasets.

from the LibSVM library [Chang and Lin, 2001]. Using a cross validation method, we found the following values for our parameters: $\alpha = 0.5$, $\theta = 0.5$, $a_{\min} = 0.85$ and $\epsilon = 0.2$ for the UCI dataset and $\epsilon = 0.01$ for CASAS datasets. A C4.5 decision tree from the RapidMiner tool was used for constructing rules [Mierswa et al., 2006]. We set the minimal information gain to 0.1. We confined the depth of the tree to a maximum of 10 attributes to prevent very long queries, though in most cases the generated query is shorter.

To simulate the effect of a human oracle in determining the label of a specific rule, we consider all the instances that are covered by a specific rule. The label of

the rule is then reported as the majority label and the confidence is reported as the majority fraction. It should be noted that our datasets as real world datasets contain noisy examples, which can lead to lower confidences. Similarly, the noisy examples can also confuse a domain expert in the real world. Though by infusing many similar cases together into a generic query, the effect of noisy examples is reduced, still in some cases this might lead to a lower confidence.

After performing the initial preprocessing steps such as proximity computation, we ran our algorithm on both sets of UCI and CASAS datasets.

First we show two example queries from the wine dataset and the B3 activity dataset. In the B3 query, the actual sensor numbers have been replaced by their location.

Wine query "*What is the alcohol type if alcohol percentage is* 12–14 *and Proanthocyanidins is* 0.4–1.2 *and color intensity is* 1.2–7.1*?*"

B3 query "*What is the activity label if duration is* 5–101 *minutes and* 78%–100% *of the sensors are work area sensors and* 1%–2% *of the activated sensors are bedroom sensors?*"

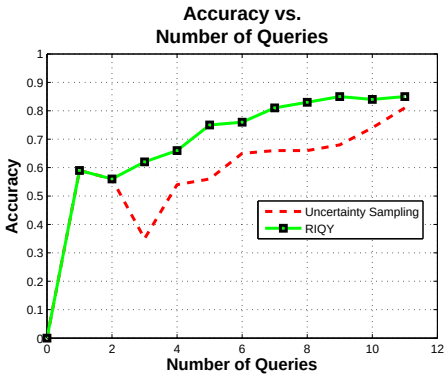
The first query reduces the number of features by 76.9% and it aggregates a total of 13 instances. The second query reduces the number of features by 80% and it aggregates a total of 113 instances. One can clearly see how our method results in

shorter and more intuitive queries.

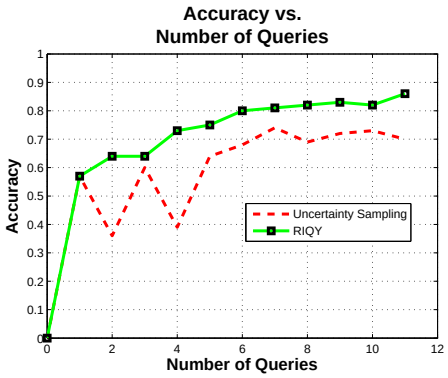
Next, we computed the classification accuracy of our method. We set the number of selected rules N to 1. In order to compare our method to a traditional active learning method, we also performed the same experiments using an uncertainty sampling method. The results of our experiments on UCI datasets can be seen in Figure 6.6. Similarly, the results on the WSU datasets are shown in Figure 6.7. We can see that our method outperforms the uncertainty sampling method in most cases by reaching higher accuracy rates with fewer queries. This can be attributed to utilizing the actual underlying distribution of data and avoiding querying the outliers.

It should be noted that the results of applying our RIQY method might vary from dataset to dataset, depending on the type of data and whether it can be generalized easily or not. Also the noisy instances and the regularity of the dataset can play a role in determining the classification accuracy. For example, dataset T2 represents a smart apartment where the residents did not have a regular schedule. Therefore, the algorithm is not able to take much advantage of the similarities between various instances of the same activity. Despite all these, we still can see that in most cases our method offers a higher accuracy rate by using fewer and shorter queries.

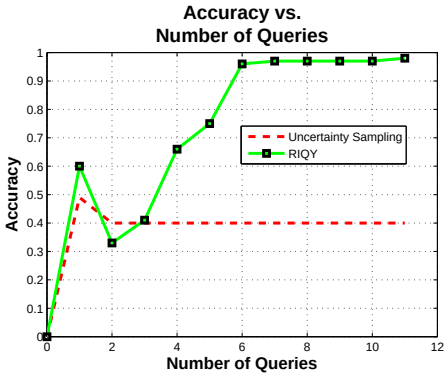
Table 6.5 and Table 6.6 show the average feature reduction, the average oracle confidence and the average number of added instances for the UCI and WSU datasets. Note that the average number of added instances is 1 for the uncertainty sampling



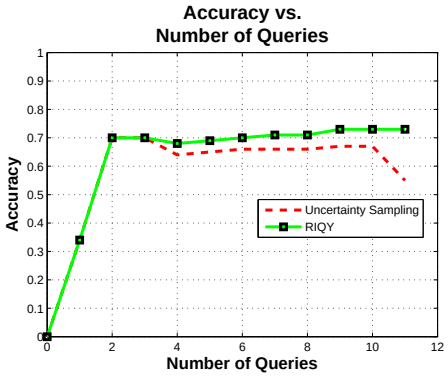
(a) Heart disease dataset.



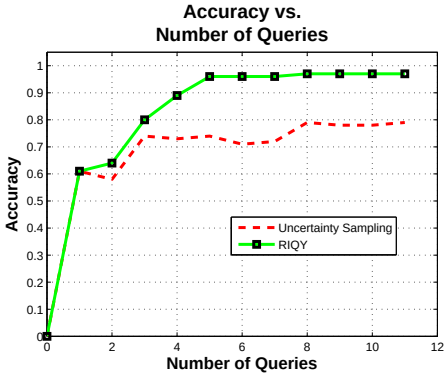
(b) Ionosphere dataset.



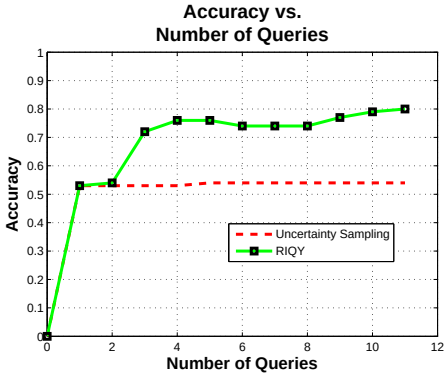
(c) Wine dataset.



(d) German credit approval dataset.

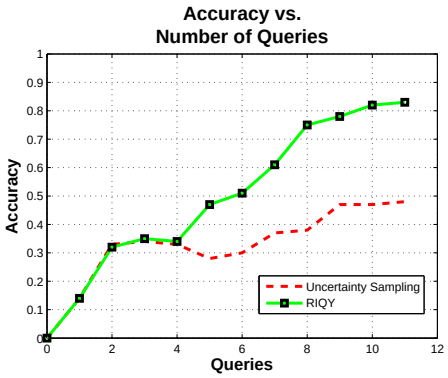


(e) Wisconsin breast cancer dataset.

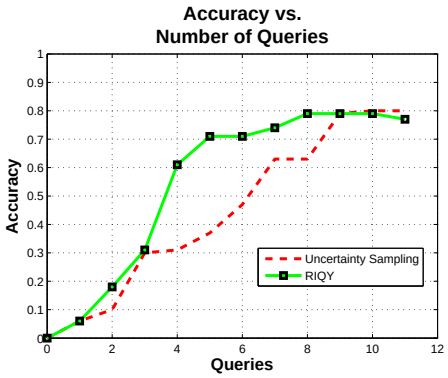


(f) Chess dataset.

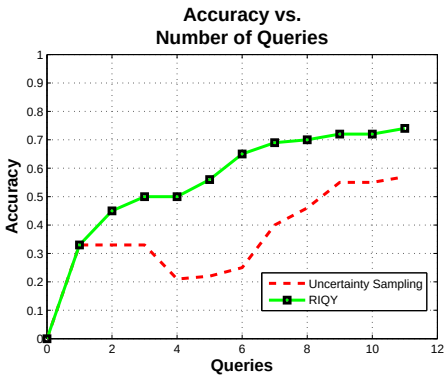
Figure 6.6: Accuracy for different UCI datasets vs. the number of queries.



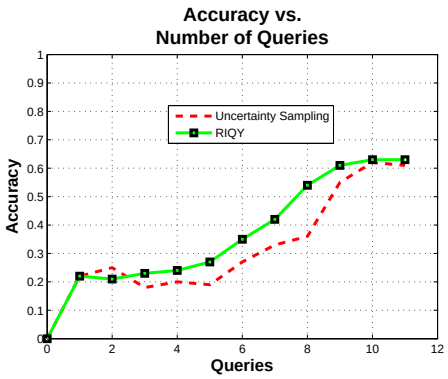
(a) B3 dataset.



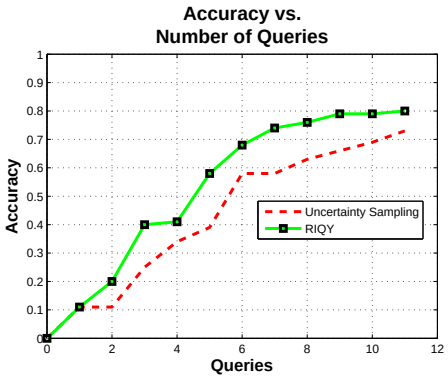
(b) C dataset.



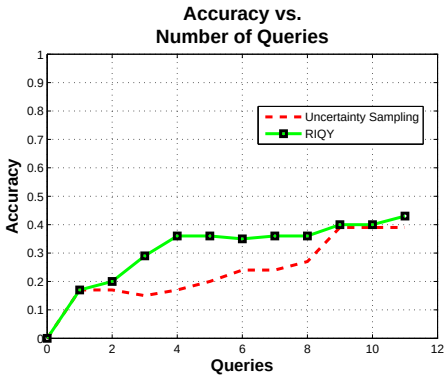
(c) K4 dataset.



(d) M dataset.



(e) T1 dataset.



(f) T2 dataset.

Figure 6.7: Accuracy for different CASAS datasets vs. the number of queries.

method. From these tables we can see how in average the number of features is reduced by our RIQY method. This makes it possible to pose shorter queries.

Dataset	Features Reduction	Oracle Confidence	Added Instances
Credit	84.5%	74.4%	51
Ionosphere	90.8%	76.4%	23
Wine	77.6%	75.0%	10
Heart	76.9%	77.6%	14
Cancer	68.8%%	94.5%	34
Chess	88.6%	88.0%	200

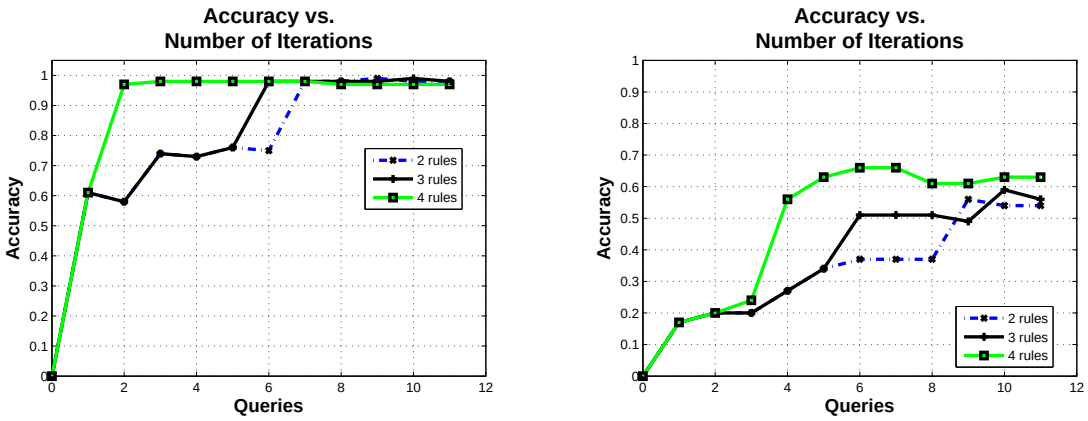
Table 6.5: Some statistics based on our RIQY active learning method for UCI datasets.

In another experiment, we explored the effect of increasing the number of posed rules (N) to the oracle. Figure 6.8 shows the results of our experiment on two example datasets: the Wisconsin breast cancer dataset and the T2 dataset. We can see that increasing the number of posed rules to the oracle can lead to higher accuracy rates. Note that as we add the rules in the order of descending coverage, posing the last rule might not have the same effect as posing the first rule. Also though increasing the number of posed rules to the oracle increases the total number of queries, however many irrelevant features are not included in each rule. Therefore despite posing more

Dataset	Features Reduction	Oracle Confidence	Added Instances
B3	88.0%	64.6%	66
C	85.0%	74.9%	12
K4	86.0%	71.1%	16
M	81.0%	64.9%	44
T1	73.6%%	88.9%	40
T2	90.3%	46.4%	2

Table 6.6: Some statistics based on our RIQY active learning method for CASAS datasets.

queries, the total number of features to be handled by the oracle is still small.



(a) Wisconsin breast cancer dataset.

(b) T2 dataset.

Figure 6.8: Accuracy vs. number of posed rules.

In summary, the above results show how we can construct more generic active learning queries based on rule induction. Our results confirm that such a method can lead to higher classification accuracy rates with fewer and shorter queries.

6.8 Summary

In this chapter, we showed two methods for constructing more generic active learning queries based on rule induction and heuristic feature selection. Our methods are able to construct shorter and more intuitive queries that are easier for a human oracle to answer, allowing us to better utilize our human resources. Our methods also allow the learning algorithm to achieve a higher accuracy rate using fewer queries.

CHAPTER 7. CONCLUSION

In this dissertation, we highlight challenges that face current activity discovery and recognition methods. These challenges make it difficult and sometimes even impractical for smart environment technologies to be deployed in real world situations. In order to address these challenges and to achieve a more scalable solution, we proposed a number of novel data mining and machine learning methods.

It should be noted that though we have studied our algorithms in the context of activity recognition in smart environments, these methods are not limited to activity recognition problems. One can use our proposed methods in a variety of different machine learning and data mining problems. As an example, in case of our active learning algorithms, we showed how our methods can be useful for other real world problems from the UCI data repository.

Our sequence mining methods, DVSM [Rashidi et al., 2010] and COM [Rashidi and Cook, 2010c], are the first data mining methods to deal with discontinuous and varied order events across a non-transactional dataset. Non-transactional datasets pose more challenges compared to transactional datasets, as they do not have a clear boundary between episodes or transactions.

Our stream data mining method, StreamCOM [Rashidi and Cook, 2010b], is

the first stream data mining method to handle a non-transactional data stream at multiple granularity. It also exploits the discontinuous and varied order model of COM mining method, which can be quite useful for mining complex data streams.

Our transfer learning solution composes of several novel algorithm for transferring activity models across different residents or different physical spaces [Rashidi and Cook, 2009a, 2010a,d,e]. Also our domain selection method is the first proposed method for selecting the most promising smart homes for activity transfer.

Finally, we showed two novel active learning methods, template based active learning and RIQY, which can construct more generic and intuitive queries with promising results. We showed how our active learning methods can be applied to a variety of real world problems, besides activity recognition.

7.1 Suggestions For Future Work

As previously mentioned, our sequence and stream mining algorithms have the potential to be applied to wide range of applications such as web click stream analysis or DNA sequence analysis. Ultimately, we hope that our model can be used in a fully functional system deployed in a real home. A great extension to the system can be an anomaly detection component to detect anomalies in observed data over time.

For our transfer learning methods, it should be noted that we made a number of

simplifying assumptions to deal with the complex nature of activities in the real world. In this work, we ignored the complications that might arise when multiple residents are present. Besides, some activities are not always consistent and show a change of pattern over time (e.g. dinner time shifted). This can cause a low recognition rate after some time. A future goals can be to address multi-resident issues by using entity detection methods [Crandall and Cook, 2008]. As part of the future work, one also can detect changes in patterns over time. Another direction is to extend our method to be able to transfer activities across spaces with different functionalities.

Our active learning methods can be extended to handle more sophisticated data types, such as graphs and sequences. One also can explore the effects of a noisy oracle and to propose methods for preventing the resulting accuracy degradation. It would be also interesting to perform actual user studies with human oracles to further explore the advantages of using our method over similar traditional active learning methods.

Bibliography

G. Abowd and E. Mynatt. *Smart Environments: Technology, Protocols, and Applications*, chapter Designing for the human experience in smart environments., pages 153–174. Wiley, 2004.

Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, SIGMOD '93, pages 207–216, New York, NY, USA, 1993. ACM. ISBN 0-89791-592-5. doi: http://doi.acm.org/10.1145/170035.170072. URL http://doi.acm.org/10.1145/170035.170072.

A. Ahmadi, D.D. Rowlands, and D.A. James. Investigating the translational and rotational motion of the swing using accelerometers for athlete skill assessment. In *IEEE Conference on Sensors*, pages 980 –983, 2006.

Mehran Asadi and Manfred Huber. Effective control knowledge transfer through learning skill and representation hierarchies. In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2054–2059, 2007.

Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 429–435, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: http://doi.acm.org/10.1145/775047.775109. URL http://doi.acm.org/10.1145/775047.775109.

T.S. Barger, D.E. Brown, and M. Alwan. Health-status monitoring through analysis of behavioral patterns. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1):22–27, Jan. 2005. ISSN 1083–4427.

T. Batu, L. Fortnow, R. Rubinfeld, W. D. Smith, and P. White. Testing that distributions are close. In *Symposium on Foundations of Computer Science*, page 259, 2000.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, 7:2399–2434, 2006.

Shai Ben-david, John Blitzer, Koby Crammer, and Presented Marina Sokolova. Analysis of representations for domain adaptation. In *Neural Information Processing Systems*. MIT Press, 2007.

G. Demiris B.K. Hensel and K.L. Courtney. Defining obtrusiveness in home tele-

health technologies: A conceptual framework. *Journal of the American Medical Informatics Association*, 13:428–431, 2006.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Empirical Methods in Natural Language Processing*, pages 120–128, 2006.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics*, pages 187–205, 2007.

M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, CVPR '97, pages 994–, Washington, DC, USA, 1997. IEEE Computer Society. ISBN 0-8186-7822-4. URL http://portal.acm.org/citation.cfm?id=794189.794420.

O. Brdiczka, J. Maisonnasse, and P. Reignier. Automatic detection of interaction groups. In *Proceedings of the 7th international conference on Multimodal interfaces*, pages 32–36, 2005.

Oliver Brdiczka, James L. Crowley, and Patrick Reignier. Learning situation models in a smart home. *Trans. Sys. Man Cyber. Part B*, 39:56–63,

February 2009. ISSN 1083-4419. doi: 10.1109/TSMCB.2008.923526. URL http://portal.acm.org/citation.cfm?id=1656702.1656709.

Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *International Conference on Machine Learning*, pages 59–66, 2003.

Fazli Can. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems*, 11(2):143–164, 1993.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Joong Hyuk Chang and Won Suk Lee. Finding recent frequent itemsets adaptively over online data streams. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 487–492, New York, NY, USA, 2003. ACM.

Gong Chen, Xindong Wu, and Xingquan Zhu. Sequential pattern mining in multiple streams. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 585–588, Washington, DC, USA, 2005a. IEEE Computer Society.

Ming-yu Chen, Michael Christel, Alexander Hauptmann, and Howard Wactlar. Putting active learning into multimedia applications: dynamic definition and refinement of concept classifiers. In *13th annual ACM international conference on Multimedia*, MultiMedia 2005, pages 902–911, New York, NY, USA, 2005b. ACM. ISBN 1-59593-044-2.

Yen-Liang Chen, Shih-Sheng Chen, and Ping-Yu Hsu. Mining hybrid sequential patterns and sequential rules. *Information Systems*, 27(5):345–362, 2002. ISSN 0306-4379.

James Cheng, Yiping Ke, and Wilfred Ng. A survey on algorithms for mining frequent itemsets over data streams. *Knowledge and Information Systems*, 16(1):1–27, 2008.

Dirk Colbry. Execution mutioni with quantitative temporal bayesian networks. In *In 6th International Conference on AI Planning and Scheduling*, 2002.

D. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48(05):480–485, 2009.

Diane Cook and Sajal Das. *Smart Environments: Technology, Protocols and Applications*. Wiley Series on Parallel and Distributed Computing. Wiley-Interscience, 2004.

D.J. Cook, M. Youngblood, III Heierman, E.O., K. Gopalratnam, S. Rao, A. Litvin,

and F. Khawaja. Mavhome: an agent-based smart home. In *IEEE International Conference on Pervasive Computing and Communications*, pages 521–524, March 2003.

Koby Crammer, Michael Kearns, and Jennifer Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9:1757–1774, 2008. ISSN 1532-4435.

A.S. Crandall and D. Cook. Attributing events to individuals in multi-inhabitant environments. In *Intelligent Environments, 2008 IET 4th International Conference on*, pages 1 –8, 2008.

Dorothy W Curtis, Esteban J Pino, Jacob M Bailey, Eugene I Shih, Jason Waterman, Staal A Vinterbo, Thomas O Stair, John V Guttag, Robert A Greenes, and Lucila Ohno-Machado. Smart:an integrated wireless system for monitoring unattended patients. *Journal of the American Medical Informatics Association*, 15(1):44–53, 2008.

Hal Daumé. Frustratingly easy domain adaptation. In *Association for Computational Linguistics*, 2007.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *The Royal Statistical Society*, 39(1):1–38, 1977.

Thomas G. Dietterich. Ensemble methods in machine learning. In *MCS '00: Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15, 2000.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89: 31–71, January 1997.

Christos Dimitrakakis and Samy Bengio. Boosting hmms with an application to speech recognition, 2004.

Todor Dimitrov, Josef Pauli, and Edwin Naroska. Unsupervised recognition of adls. In Stasinos Konstantopoulos, Stavros Perantonis, Vangelis Karkaletsis, Constantine Spyropoulos, and George Vouros, editors, *Artificial Intelligence: Theories, Models and Applications*, volume 6040 of *Lecture Notes in Computer Science*, pages 71–80. Springer Berlin / Heidelberg, 2010.

F. Doctor, H. Hagras, and V. Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35(1): 55–65, Jan. 2005.

Pinar Donmez, Jaime G. Carbonell, and Paul N. Bennett. Dual strategy active learning. In *Proceedings of the 18th European conference on Machine Learn-*

ing, ECML '07, pages 116–127, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-74957-8. doi: http://dx.doi.org/10.1007/978-3-540-74958-5_14. URL http://dx.doi.org/10.1007/978-3-540-74958-5_14.

Gregory Druck, Gideon Mann, and Andrew McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 595–602, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-164-4. doi: http://doi.acm.org/10.1145/1390334.1390436. URL http://doi.acm.org/10.1145/1390334.1390436.

Jun Du and Charles X. Ling. Asking generalized queries to domain experts to improve learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:812–825, 2010.

Lixin Duan, Ivor W. Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *International Conference on Machine Learning*, pages 289–296, 2009.

Thi V. Duong, Hung H. Bui, Dinh Q. Phung, and Svetha Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*,

CVPR '05, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2372-2. doi: http://dx.doi.org/10.1109/CVPR.2005.61. URL http://dx.doi.org/10.1109/CVPR.2005.61.

Jesus Favela, Monica Tentori, Luis A. Castro, Victor M. Gonzalez, Elisa B. Moran, and Ana I. Martínez-García. Activity recognition for context-aware hospital applications: issues and opportunities for the deployment of pervasive networks. *Mob. Netw. Appl.*, 12:155–171, March 2007. ISSN 1383-469X. doi: http://dx.doi.org/10.1007/s11036-007-0013-5. URL http://dx.doi.org/10.1007/s11036-007-0013-5.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag. ISBN 3-540-59119-2.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Mach. Learn.*, 28:133–168, September 1997. ISSN 0885-6125. doi: 10.1023/A:1007330508534. URL http://portal.acm.org/citation.cfm?id=263100.263123.

Hua fu Li, Suh yin Lee, and Man kwan Shan. An efficient algorithm for mining frequent itemsets over the entire history of data streams. In *In Proc. of First International Workshop on Knowledge Discovery in Data Streams*, 2004.

Minos Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Querying and mining data streams: you only get one look a tutorial. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 635–635, New York, NY, USA, 2002. ACM. ISBN 1-58113-497-5. doi: http://doi.acm.org/10.1145/564691.564794.

Zoubin Ghahramani. Learning dynamic bayesian networks. In *Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks, "E.R. Caianiello"-Tutorial Lectures*, pages 168–197, London, UK, 1998. Springer-Verlag. ISBN 3-540-64341-9. URL http://portal.acm.org/citation.cfm?id=647638.733212.

Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, and Philip S. Yu. *Mining Frequent Patterns in Data Streams at Multiple Time Granularities*, chapter 3. MIT Press, 2003.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, pages 518–529, San Francisco, CA,

USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-615-7. URL http://portal.acm.org/citation.cfm?id=645925.671516.

Karthik Gopalratnam and Diane J. Cook. Online sequential prediction via incremental parsing: The active lezi algorithm. *IEEE Intelligent Systems*, 22(1):52–58, 2007.

T. Gu, Z. Wu, X. Tao, H. Pung, , and J. Lu. epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *International Conference on Pervasive Computing and Communication*, 2009.

Yuhong Guo and Dale Schuurmans. Discriminative Batch Mode Active Learning. In *Advances in Neural Information Processing Systems*, volume 20, pages 593–600, 2008.

Rahul Gupta and Sunita Sarawagi. Domain adaptation of information extraction models. *SIGMOD*, 37(4):35–40, 2008.

Asterisk Guru. Asterisk guru. www.asteriskguru.com/tutorials, 2009.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Machine Learning Research*, 3:1157–1182, 2003.

Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Freespan: frequent pattern-projected sequential pattern mining.

In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 355–359, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. doi: http://doi.acm.org/10.1145/347090.347167. URL http://doi.acm.org/10.1145/347090.347167.

Jiawei Han, Yixin Chen, Guozhu Dong, Jian Pei, Benjamin W. Wah, Jianyong Wang, and Y. Dora Cai. Stream cube: An architecture for multi-dimensional analysis of data streams. *Distrib. Parallel Databases*, 18(2):173–197, 2005.

David J. Hand and Keming Yu. Idiot's bayes—not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.

J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

Tamara L. Hayes, Misha Pavel, Nicole Larimer, Ishan A. Tsay, John Nutt, and Andre Gustavo Adami. Distributed healthcare: Simultaneous assessment of multiple individuals. *IEEE Pervasive Computing*, 6(1):36–43, 2007. ISSN 1536-1268.

Edwin O. Heierman, III and Diane J. Cook. Improving home automation by discovering regularly occurring device usage patterns. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 537, 2003.

Sumi Helal, William Mann, Hicham El-Zabadani, Jeffrey King, Youssef Kaddoura,

and Erwin Jansen. The gator tech smart house: A programmable pervasive space. *Computer*, 38(3):50–60, 2005.

Steven C. H. Hoi, Rong Jin, Jianke Zhu, and Michael R. Lyu. Batch mode active learning and its application to medical image classification. In *23rd international conference on Machine learning*, ICML 2006, pages 417–424, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: http://doi.acm.org/10.1145/1143844.1143897. URL http://doi.acm.org/10.1145/1143844.1143897.

Derek Hao Hu, Xian-Xing Zhang, Jie Yin, Vincent Wenchen Zheng, and Qiang Yang. Abnormal activity recognition based on hdp-hmm models. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1715–1720, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. URL http://portal.acm.org/citation.cfm?id=1661445.1661721.

T. Huynh and B. Schiele. Towards less supervision in activity recognition from wearable sensors. In *Wearable Computers, 2006 10th IEEE International Symposium on*, pages 3 –10, 2006.

Rebecca Hwa. Sample selection for statistical parsing. *Computational Linguistics*, 30:253–276, September 2004.

Holger Schwenk International. Using boosting to improve a hybrid hmm/neural network speech recognizer, 1999.

Jabber. Open instant messaging and presence. http://www.jabber.org, 2009.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Daiki Kawanaka, Takayuki Okatani, and Koichiro Deguchi. Hhmm based recognition of human activity. *IEICE - Trans. Inf. Syst.*, E89-D:2180–2185, July 2006. ISSN 0916-8532. doi: 10.1093/ietisy/e89-d.7.2180. URL http://portal.acm.org/citation.cfm?id=1184860.1185053.

Eamonn Keogh, Jessica Lin, and Wagner Truppel. Clustering of time series subsequences is meaningless: Implications for previous and future research. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, page 115, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1978-4.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Very Large Data Bases*, pages 180–191, 2004.

Solomon Kullback and Richard A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on*

Machine Learning, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://portal.acm.org/citation.cfm?id=645530.655813.

Ken Lang. NewsWeeder: learning to filter netnews. In *International Conference on Machine Learning*, pages 331–339, 1995.

V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X.

L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using relational markov networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 773–778, 2005.

Ja Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, 1991.

Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *Proceedings of the fifth ACM SIGKDD international conference*

on Knowledge discovery and data mining, KDD '99, pages 337–341, New York, NY, USA, 1999. ACM. ISBN 1-58113-143-7. doi: http://doi.acm.org/10.1145/312129.312274. URL http://doi.acm.org/10.1145/312129.312274.

K. K. Loo, Ivy Tong, Ben Kao, and David Cheung. Online algorithms for mining inter-stream associations from large sensor networks. In *PAKDD*, 2005.

Konrad Lorincz, David J. Malan, Thaddeus R. F. Fulford-Jones, Alan Nawoj, Antony Clavel, Victor Shnayder, Geoffrey Mainland, Matt Welsh, and Steve Moulton. Sensor networks for emergency response: Challenges and opportunities. *IEEE Pervasive Computing*, 3(4):16–23, 2004. ISSN 1536-1268.

Konrad Lorincz, Bor-rong Chen, Geoffrey Werner Challen, Atanu Roy Chowdhury, Shyamal Patel, Paolo Bonato, and Matt Welsh. Mercury: a wearable sensor network platform for high-fidelity motion analysis. In *ACM Conference on Embedded Networked Sensor Systems*, pages 183–196, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-519-2.

Ping Luo, Fuzhen Zhuang, Hui Xiong, Yuhong Xiong, and Qing He. Transfer learning from multiple source domains via consensus regularization. In *Information and Knowledge Management*, pages 103–112, 2008.

J. B. MacQueen. Some methods for classification and analysis of multivariate observa-

tions. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

Dhruv Mahajan, Nipun Kwatra, Sumit Jain, Prem Kalra, and Subhashis Banerjee. A framework for activity recognition and detection of unusual activities. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2004.

Ketil Malde, Eivind Coward, and Inge Jonassen. Fast sequence clustering using a suffix array algorithm. *Bioinformatics*, 19(10):1221–1226, July 2003.

Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*, pages 346–357. VLDB Endowment, 2002.

Christopher D. Manning and Hinrich Schtze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. ISBN 0-262-13360-1.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems*, pages 1041–1048, 2009.

Alice Marascu and Florent Masseglia. Mining sequential patterns from data streams: a centroid approach. *Journal of Intelligent Information Systems*, 27(3):291–307, 2006. ISSN 0925-9902.

Florent Masseglia, Fabienne Cathala, and Pascal Poncelet. The psp approach for mining sequential patterns. In *PKDD '98: Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 176–184, London, UK, 1998. Springer-Verlag.

U. Maurer, A. Smailagic, D.P. Siewiorek, and M. Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *International Workshop on Wearable and Implantable Body Sensor Networks, 2006. BSN 2006*, pages 4–116, April 2006.

Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *International Conference on Machine Learning*, ICML '98, pages 350–358, 1998.

Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. *Advanced Lectures on Machine Learning*, pages 118–183, 2003.

Florian Michahelles and Bernt Schiele. Sensing and monitoring professional skiers. *IEEE Pervasive Computing*, 4(3):40–46, 2005. ISSN 1536-1268.

Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: Rapid prototyping for complex data mining tasks. In Lyle Ungar, Mark Craven, Dimitrios Gunopulos, and Tina Eliassi-Rad, editors, *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery*

and data mining, pages 935–940, New York, NY, USA, August 2006. ACM. ISBN 1-59593-339-5. doi: http://doi.acm.org/10.1145/1150402.1150531. URL http://rapid-i.com/component/option,com_docman/task,doc_download/gid,25/Itemid,62/.

G. W. Milligan and M. C. Cooper. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 1985.

Ion Muslea, Steven Minton, and Craig A. Knoblock. Selective sampling with redundant views. In *17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, AAAI 2000, pages 621–626. AAAI Press, 2000. ISBN 0-262-51112-6. URL http://portal.acm.org/citation.cfm?id=647288.721119.

Hieu T. Nguyen and Arnold Smeulders. Active learning using preclustering. In *International Conference on Machine Learning*, pages 79–89, 2004.

Sang-Kyun Noh, Yong-Min Kim, DongKook Kim, and Bong-Nam Noh. *Computational Science and Its Applications*, volume 3981 2006 of *Lecture Notes in Computer Science*, chapter Network Anomaly Detection Based on Clustering of Sequence Patterns, pages 349–358. Springer Berlin Heidelberg, May 2006.

Georg Ogris, Thomas Stiefmeier, Paul Lukowicz, and Gerhard Troster. Using a complex multi-modal on-body sensor system for activity spotting. In *IEEE International Symposium on Wearable Computers*, pages 55–62, 2008.

Nuria Oliver, Eric Horvitz, and Ashutosh Garg. Layered representations for human activity recognition. In *Proceedings of the 4th IEEE International Conference on Multimodal Interfaces*, ICMI '02, pages 3–, Washington, DC, USA, 2002. IEEE Computer Society. ISBN 0-7695-1834-6. doi: http://dx.doi.org/10.1109/ICMI.2002.1166960. URL http://dx.doi.org/10.1109/ICMI.2002.1166960.

Paulito Palmes, Hung Keng Pung, Tao Gu, Wenwei Xue, and Shaxun Chen. Object relevance weight pattern mining for activity recognition and segmentation. *Pervasive Mob. Comput.*, 6:43–57, February 2010. ISSN 1574-1192. doi: http://dx.doi.org/10.1016/j.pmcj.2009.10.004. URL http://dx.doi.org/10.1016/j.pmcj.2009.10.004.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

Rina Panigrahy. An improved algorithm finding nearest neighbor using kd-trees. In *Proceedings of the 8th Latin American conference on Theoretical informatics*, LATIN'08, pages 387–398, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 3-540-78772-0, 978-3-540-78772-3. URL http://portal.acm.org/citation.cfm?id=1792918.1792952.

Spiros Papadimitriou, Anthony Brockwell, and Christos Faloutsos. Adaptive, hands-

off stream mining. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 560–571. VLDB Endowment, 2003.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns by prefix-projected growth. In *Proceedings of the 17th International Conference on Data Engineering*, pages 215–224, Washington, DC, USA, 2001. IEEE Computer Society.

Jian Pei, Jiawei Han, and Wei Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *Journal of Intelligent Information Systems*, 28(2): 133–160, 2007. ISSN 0925-9902.

Alex Sandy Pentland. Healthwear: Medical technology becomes wearable. *Computer*, 37(5):42–49, 2004. ISSN 0018-9162.

Aritz Pérez, Pedro Larraòaga, and Iòaki Inza. Bayesian classifiers based on kernel density estimation: Flexible classifiers. *International Journal of Approximate Reasoning*, 50(2):341–362, 2009.

Mike Perkowitz, Matthai Philipose, Kenneth Fishkin, and Donald J. Patterson. Mining models of human activities from the web. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 573–582, New York, NY,

USA, 2004. ACM. ISBN 1-58113-844-X. doi: http://doi.acm.org/10.1145/988672.988750. URL http://doi.acm.org/10.1145/988672.988750.

Matthai Philipose, Kenneth P. Fishkin, Mike Perkowitz, Donald J. Patterson, Dieter Fox, Henry Kautz, and Dirk Hahnel. Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57, 2004. ISSN 1536-1268.

M. Pollack. Intelligent technology for an aging population: The use of ai to assist elders with cognitive impairment. *AI Magazine*, 26(2):9–24, 2005.

D. Pyle. *Do Smart Adaptive Systems Exist?*, chapter Data Preparation and Preprocessing, pages 27–53. Springer, 2005.

J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-238-0.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in speech recognition*, pages 267–296, San Francisco, CA, USA, 1990. Morgan Kaufmann Publishers Inc. ISBN 1-55860-124-4. URL http://portal.acm.org/citation.cfm?id=108235.108253.

Rajat Raina, Andrew Y. Ng, and Daphne Koller. Constructing informative priors us-

ing transfer learning. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 713–720, 2006.

Chedi Raïssi, Pascal Poncelet, and Maguelonne Teisseire. Need for speed: Mining sequential pattens in data streams. In *BDA05: Actes des 21iemes Journees Bases de Donnees Avancees*, October 2005.

Chedy Raïssi and Marc Plantevit. Mining multidimensional sequential patterns over data streams. In *DaWaK '08: Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*, pages 263–272, Berlin, Heidelberg, 2008. Springer-Verlag.

Parisa Rashidi and Diane J. Cook. Adapting to resident preferences in smart environments. In *AAAI Workshop on Preference Handling*, pages 78–84, 2008.

Parisa Rashidi and Diane J. Cook. Activity recognition based on home to home transfer learning. In *AAAI Workshop on Plan, Activity, and Intent Recognition*, 2010a.

Parisa Rashidi and Diane J. Cook. Mining sensor streams for discovering human activity patterns over time. *Data Mining, IEEE International Conference on*, 0: 431–440, 2010b. ISSN 1550-4786. doi: http://doi.ieeecomputersociety.org/10.1109/ICDM.2010.40.

Parisa Rashidi and Diane J. Cook. Transferring learned activities in smart environments. In *5th International Conference on Intelligent Environments*, volume 2 of *Ambient Intelligence and Smart Environments*, pages 185–192, 2009a.

Parisa Rashidi and Diane J. Cook. Mining and monitoring patterns of daily routines for assisted living in real world settings. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI '10, pages 336–345, New York, NY, USA, 2010c. ACM. ISBN 978-1-4503-0030-8. doi: http://doi.acm.org/10.1145/1882992.1883040. URL http://doi.acm.org/10.1145/1882992.1883040.

Parisa Rashidi and Diane J. Cook. Multi home transfer learning for resident activity discovery and recognition. In *KDD International Workshop on Knowledge Discovery from Sensor Data*, pages 53–63, 2010d.

Parisa Rashidi and Diane J. Cook. Transferring learned activities and cues between different residential spaces. *Journal of Pervasive and Mobile Computing (PMC)*, 2010e. In Press.

Parisa Rashidi and Diane J. Cook. the resident in the loop: Adapting the smart home to the user. *IEEE Transactions on Systems, Man, and Cybernetics journal, Part A*, 39(5):949–959, 2009b.

Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment.

IEEE Transactions on Knowledge and Data Engineering, 99(PrePrints), 2010. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.148.

Barry Reisberg, Sanford Finkel, John Overall, Norbert Schmidt-Gollas, Siegfried Kanowski, Hartmut Lehfeld, Franz Hulla, Steven G. Sclan, Hans-Ulrich Wilms, Kurt Heininger, Ian Hindmarch, Mark Stemmler, Leonard Poon, Alan Kluger, Carolyn Cooler, Manfred Bergener, Laurence Hugonot-Diener, Philippe H. Robert, and Hellmut Erzigkeit. The Alzheimer's disease activities of daily living international scale (adl-is). *International Psychogeriatrics*, 13(02):163–181, 2001.

Jiadong Ren and Cong Huo. Mining closed frequent itemsets in sliding window over data streams. In *Innovative Computing Information and Control, 2008. ICICIC '08. 3rd International Conference on*, pages 76 –76, 18-20 2008.

V. Rialle, C. Ollivet, C. Guigui, and C. Hervé. What do family caregivers of alzheimer's disease patients desire in smart home technologies? contrasted results of a wide survey. *Methods of Information in Medicine*, 47:63–9, 2008.

J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G. Dietterich. To transfer or not to transfer. In *NIPS Inductive Transfer: 10 Years Later*, 2005.

Daniel M. Roy and Leslie P. Kaelbling. Efficient bayesian task-level transfer learning.

In *IJCAI'07: Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2599–2604, 2007.

Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *International Conference on Machine Learning*, pages 441–448, 2001.

A. Ruotsalainen and T. Ala-Kleemola. Gais: A method for detecting discontinuous sequential patterns from imperfect data. In *Proceedings of the International Conference on Data Mining*, pages 530–534, 2007.

A. Salarian, H. Russmann, FJG. Vingerhoets, C. Dehollain, Y. Blanc, PR. Burkhard, and K. Aminian. Gait assessment in Parkinson's disease: Toward an ambulatory system for long-term monitoring. *IEEE Transactions on Biomedical Engineering*, 51(8):1434–1443, 2004.

Dairazalia Sánchez, Monica Tentori, and Jesús Favela. Activity recognition for the smart hospital. *IEEE Intelligent Systems*, 23(2):50–57, 2008. ISSN 1541-1672.

H. Saneifar, S. Bringay, A. Laurent, and M. Teisseire. S2mp: Similarity measure for sequential patterns. In *AusDM*, volume 87, pages 95–104, 2008.

Robert E. Schapire and Yoram Singer. Improved boosting algorithms using

confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999. ISSN 0885-6125.

Bernt Schiele. Unsupervised discovery of structure in activity data using multiple eigenspaces. In *2nd International Workshop on Location and Context Awareness*. Springer, 2006.

M. Schmitter-Edgecombe, E. Woo, , and D. Greeley. Memory deficits, everyday functioning, and mild cognitive impairment. *Proceedings of the Annual Rehabilitation Psychology Conference*, 2008.

S. R. Searle, editor. *Linear models for unbalanced data*. John Wiley & Sons, Inc., New York, NY, USA, 1987.

K. Sequeira and M. Zaki. Admit: anomaly-based data mining for intrusions. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 386–395, 2002.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Empirical Methods in Natural Language Processing*, pages 1070–1079, 2008.

H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *workshop on Computational learning theory*, pages 287–294, 1992.

Cristian Sminchisescu, Atul Kanaujia, Zhiguo Li, and Dimitris Metaxas. Conditional random fields for contextual human motion recognition. In *Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2*, ICCV '05, pages 1808–1815, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2334-X-02. doi: http://dx.doi.org/10.1109/ICCV.2005.59. URL http://dx.doi.org/10.1109/ICCV.2005.59.

C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, Aug 2000. ISSN 0162-8828.

Catherine A. Sugar, Gareth, and M. James. Finding the number of clusters in a data set: An information theoretic approach. *Journal of the American Statistical Association*, 98:750–763, 2003.

Gita Sukthankar and Katia Sycara. Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, AAMAS '06, pages 638–645, New York, NY, USA, 2006. ACM. ISBN 1-59593-303-4. doi: http://doi.acm.org/10.1145/1160633.1160746. URL http://doi.acm.org/10.1145/1160633.1160746.

S. Szewcyzk, K. Dwan, B. Minor, B. Swedlove, and D. Cook. Annotating smart environment sensor data for activity learning. *Technol. Health Care*, 17:161–169, August 2009. ISSN 0928-7329. URL http://portal.acm.org/citation.cfm?id=1605363.1605365.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Adison Wesley, 2005.

Emmanuel M. Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*, pages 158–175, 2004a.

Emmanuel M. Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. *Pervasive Computing*, pages 158–175, 2004b.

Matthew E. Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8:2125–2167, 2007.

Wei-Guang Teng, Ming-Syan Chen, and Philip S. Yu. A regression-based temporal pattern mining scheme for data streams. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 93–104. VLDB Endowment, 2003. ISBN 0-12-722442-4.

S. Thrun. Is learning the nth thing any easier than learning the first? *In Advances in Neural Information Processing Systems*, 8:640–646, 1996.

Katrin Tomanek and Fredrik Olsson. A web survey on the use of active learning to support annotation of text data. In *HLT '09: Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 45–48, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Simon Tong and Daphne Koller. Support vector machine active learning with application sto text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 999–1006, 2000.

Alireza Vahdatpour, Navid Amini, and Majid Sarrafzadeh. Toward unsupervised activity discovery using multi-dimensional motif detection in time series. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1261–1266, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc.

Douglas L. Vail, Manuela M. Veloso, and John D. Lafferty. Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 235:1–235:8, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5. doi: http://doi.acm.org/10.1145/1329125.1329409. URL http://doi.acm.org/10.1145/1329125.1329409.

T. van Kasteren and B. Krose. Bayesian activity recognition in residence for elders. In

3rd IET International Conference on Intelligent Environments, 2007. IE 07, pages 209–212, Sept. 2007.

T.L.M. van Kasteren, G. Englebienne, and B.J.A. Krose. Recognizing activities in multiple contexts using transfer learning. In *AAAI AI in Eldercare Symposium*, 2008.

Wadley VG, Okonkwo O, Crowe M, and Ross-Meadows LA. Mild cognitive impairment and everyday function: Evidence of reduced speed in performing instrumental activities of daily living. *American Journal of Geriatric Psychiatry*, 16(5):416–424, May 2008.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, Apr 1967. ISSN 0018-9448.

Jianyong Wang and Jiawei Han. Bide: Efficient mining of frequent closed sequences. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 79, Washington, DC, USA, 2004. IEEE Computer Society.

Manfred K. Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active learning with support vector machines in the drug discovery process. *Journal of Chemical Information and Computer Sciences*, 43, 2003.

R. Wray and J.E. Laird. Variability in human modeling of military simulations. In *Conference of Behaviour Representation in Modeling and Simulation*, pages 160–169, 2003.

C. Wren and E. Munguia-Tapia. Toward scalable activity recognition for sensor networks. In *Proceedings of the Workshop on Location and Context-Awareness*, pages 218–235, 2006.

Danny Wyatt, Matthai Philipose, and Tanzeem Choudhury. Unsupervised activity recognition using automatically mined common sense. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 1*, pages 21–27. AAAI Press, 2005.

Zhao Xu, Kai Yu, Volker Tresp, Xiaowei Xu, and Jizhi Wang. Representative sampling for text classification using support vector machines. In *European Conference on IR research*, ECIR'03, pages 393–407, 2003.

Zuobing Xu, Ram Akella, and Yi Zhang. Incorporating diversity and density in active learning for relevance feedback. In *European conference on IR research*, ECIR'07, pages 246–257, 2007.

Jiong Yang and Wei Wang. Towards automatic clustering of protein sequences. In *IEEE Computer Society Bioinformatics Conference, 2002*, pages 175–186, 2002.

Jun Yang, Rong Yan, and Alexander G. Hauptmann. Cross-domain video concept detection using adaptive svms. In *International Conference on Multimedia*, pages 188–197, 2007.

Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan. Sensor-based abnormal human-activity detection. *IEEE Transactions on Knowledge and Data Engineering*, 20(8):1082–1090, 2008. ISSN 1041-4347.

Tang Yiping, Jin Shunjing, Yang Zhongyuan, and You Sisi. Detection elder abnormal activities by using omni-directional vision sensor: Activity data collection and modeling. In *SICE-ICASE, 2006. International Joint Conference*, pages 3850–3853, 2006.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *International Conference on Machine Learning*, pages 609–616, 2001.

Mohammed J. Zaki. Spade: an efficient algorithm for mining frequent sequences. *Mach. Learn.*, 42:31–60, January 2001. ISSN 0885-6125. doi: 10.1023/A:1007652502315. URL http://portal.acm.org/citation.cfm?id=370660.370671.

Mohammed J. Zaki, Neal Lesh, and Mitsunori Ogihara. Planmine: Sequence mining

for plan failures. In *In 4th International Conference of Knowledge Discovery and Data Mining*, pages 369–373, 1998.

Tong Zhang and Frank J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *International Conference on Machine Learning*, pages 1191–1198, 2000.

Liyue Zhao, Xi Wang, Gita Sukthankar, and Rahul Sukthankar. Motif discovery and feature selection for crf-based activity recognition. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR '10, pages 3826–3829, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4109-9. doi: http://dx.doi.org/10.1109/ICPR.2010.932. URL http://dx.doi.org/10.1109/ICPR.2010.932.

Vincent W. Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1029–1038, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8.

Vincent Wenchen Zheng, Derek Hao Hu, and Qiang Yang. Cross-domain activity recognition. In *Ubicomp '09: Proceedings of the 11th international conference on Ubiquitous computing*, pages 61–70, 2009.