Activity recognition for health monitoring elderly using temporal probabilistic models

van Kasteren, T.L.M.

Publication date
2011

Document Version
Final published version

Activity Recognition for Health Monitoring Elderly using Temporal Probabilistic Models

Tim van Kasteren

Activity Recognition for Health Monitoring Elderly using Temporal Probabilistic Models

Tim van Kasteren

Activity Recognition for Health Monitoring Elderly using Temporal Probabilistic Models

Tim van Kasteren

Activity Recognition for Health Monitoring Elderly using Temporal Probabilistic Models

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor promoties
ingestelde commissie,
in het openbaar te verdedigen in de Aula der Universiteit
op woensdag 27 april 2011, te 15:00 uur

door

Timotheus Leonhard Martinus van Kasteren

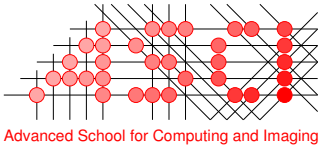
geboren te Alkmaar

Promotiecommissie

Promotor:	Prof. dr. ir. F. C. A. Groen
Copromotor:	Dr. ir. B. J. A. Kröse
Overige leden:	Prof. Dr. C. Ersoy
	Prof. Dr. D. M. Gavrila
	Prof. Dr. Ing. P. Havinga
	Prof. Dr. J. Treur
	Dr. I. Karkowski

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

This work has been supported by Cogniron and by the Centre for Intelligent Observation Systems (CIOS), which is a collaboration between UvA and TNO.



This work was carried out in the ASCI graduate school.
ASCI dissertation series number 205.

"Old age, believe me, is a good and pleasant thing.
It is true you are gently shouldered off the stage,
but then you are given such a comfortable
front stall as spectator. "

Jane Ellen Harrison

Contents

List of abbreviations

ADL activities of daily living

CRF conditional random field

DBN dynamic Bayesian network

EM Expectation Maximization

IADL instrumental activities of daily living

HMM hidden Markov model

HHMM hierarchical hidden Markov model

HSMM hidden semi-Markov model

MAP maximum a posteriori

RFID Radio frequency identification

SMCRF semi-Markov conditional random field

1

Introduction

Population aging is a serious problem affecting both the developed and the less developed countries of the world. In 2009, the elderly aged 60 and over made up 11 percent of the world population, and this is expected to reach 22 percent by 2050 [17]. This increase causes a decline in the ratio of workers to elderly and will cause economic stress in the political need to transfer financial resources to the elderly [134].

As people age, their healthcare expenses increase significantly [101], but with less financial resources to aid the elderly, many of them will not be able to make ends meet. Efficient and accurate caregiving is therefore a must to guarantee a healthy and affordable future for our elderly.

Automatic health monitoring provides an inexpensive way for obtaining the information needed to give efficient and accurate care. Besides being cost effective, elderly would also maintain their independence, allowing them to live longer in their own homes. This will result in a high quality of life for the elderly and delay the transition to costly care facilities [114].

A common method in healthcare for assessing the cognitive and physical well-being of elderly are activities of daily living (ADL), a collection of activities that are performed on a daily basis such as cooking, toileting and showering [3, 72]. Recent developments in sensing technology make it possible to easily equip existing homes with sensors, therefore allowing a continuous observation system. However, automatically interpreting this sensor data to recognize ADL is an unsolved problem.

1.1 Different Forms of Health Monitoring

To highlight the importance of using ADL for health monitoring we give an overview of the different forms of health monitoring available. We distinguish between alarm systems and long-term monitoring systems.

The most common type of alarm systems are buttons or pull-cords that send an alarm when used. These systems are widely commercially available and are installed in many homes and caregiving facilities [106]. Their effectiveness depends on the ability of the user to recognize an emergency and requires the user to press the button when an emergency takes place. A more advanced type of alarm system are fall detection systems that recognize an emergency automatically. The detection of a fall can be done using different kinds of sensors. For example, in work by Ohta et al. infrared sensors measure the duration of stay in a particular room and incidents of falling are detected by searching for excessively long durations of stay [121]. Nait-Charif et al. use cameras to track inhabitants through the house and use long inactive periods as indications of alarm [115]. Such advanced fall detection systems in which no wearable sensor needs to be carried around are now becoming commercially available. An example is the Unattended Autonomous Surveillance (UAS) system, developed at dutch research institute TNO [62].

Alarm systems detect emergencies and are very important to be able to offer aid as quickly as possible. On the other hand, long-term monitoring systems maintain information about the user over time and can therefore be used to track changes and detect anomalies. One type of information to monitor are physiological measurements such as heart rhythm and blood pressure. Systems for obtaining these measurements are now appearing as commercial health monitoring systems in the form of Intel's Health Guide [55] or Bosch's Health Buddy [10]. They prompt users to take measurements and provide a user friendly interface for doing so. Alternatively, measurements can be obtained automatically using clever sensing constructions such as measuring body weight while toileting, temperature while sleeping and an electrocardiography (ECG) while bathing [155]. The measurements can be made available to physicians for analysis and can be important indicators when something is wrong.

ADL are also suitable for long term monitoring and are commonly used indicators for monitoring the health of elderly [72]. Currently, healthcare professionals measure ADL manually by visiting the home of an elderly and observing them in performing activities. This results in a subjective measure and makes the assessment of activities such as taking medicine and eating very difficult. Caregivers, therefore, often rely on information provided by the elderly themselves, who may report biased information because of a desire to gain services or a fear of losing them [175].

Automatically recognizing ADL provides an objective indicator for caregiving. Irregular sleep patterns, changes in the frequency of toilet use and an increase in the duration of time it takes to complete an ADL are all important indicators of physical and cognitive health disorders [45, 154]. Occupational therapists can use this information to install assistive devices and daily living aids that reduce the time and energy it takes to perform an ADL. When applied on a large scale, it is expected that the wealth of information will be extremely useful to evidence based nursing, a form of nursing relying on scientific data. It is expected that such an approach makes it possible to treat certain diseases proactively, before any real damage is done [31].

Systems that recognize ADL from sensor data are now an active topic of research. Some systems dedicated to recognizing a limited number of activities such as toileting and sleeping behavior have been tried in practice [52]. But the majority of work focuses on the recognition of a large variety of ADL such as preparing breakfast, washing dishes and other kitchen activities [33, 102] or ironing, vacuuming and other housekeeping activities [149]. These systems are evaluated in a laboratory setting in which an office space is equipped with furniture to resemble a home environment. In this thesis, we evaluate our approaches using datasets recorded in real world settings and consisting of a large variety of ADL.

1.2 Patterns in Activities of Daily Living

The term 'activity' is used in the relevant literature to describe a large variety of concepts, ranging from simple physical activities to complex activities such as ADL. To allow a clear discussion, we define some terminology for distinguishing among action primitives, actions and activities. For the remainder of this thesis, we use the term activities to refer to ADL.

An action primitive is defined as a simple operation which typically takes up a few seconds. Examples are, opening a cupboard, sitting on a chair or flushing the toilet. Action primitives can typically only be performed in a very limited number of ways. For example, opening a cupboard is typically done using either the right hand or the left hand. Also the temporal order in which the action primitive is performed is rather strict, one first moves the hand towards the cupboard and then opens it by pulling it. It is not possible to open the cupboard without first moving the hand towards it. Finally, the variety in which action primitives differ per person is minimal, for example, almost everybody opens a cupboard using their hands.

A combination of action primitives forms an action, such as taking a plate from a cupboard or putting coffee powder in a coffee machine. The temporal ordering of actions corresponds to the order of the action primitives the action is made of, and

is generally rather strict. For example, taking a plate from a cupboard requires the action primitives of opening the cupboard, taking the plate and closing the cupboard. A different order of action primitives generally results in a failure to complete the action. The number of ways an action can be performed is limited and the duration of an action can range from several seconds to several minutes.

Activities are made up of a complex combination of actions and can last from several minutes to several hours. For example, sleeping, toileting and making coffee are all examples of activities. They can be performed in a large number of ways due to the complex combination of actions involved. We can distinguish between required actions and optional actions in performing an activity. For example, in making coffee a required action is adding coffee powder to the machine, an optional action is adding sugar when the coffee is done. An optional action can also be, getting the coffee powder from the cupboard, because it could still be on the counter after its last use. The temporal order in which activities can be performed varies strongly. When making coffee one can choose to first add coffee powder and then add water, or the other way around. Still for many activities there are certain constraints in the temporal order of execution. For example, adding water, turning on the machine and after several minutes adding the coffee powder does not typically result in very good coffee. Finally, activities also tend to vary strongly between individuals due to cultural differences or simply different habits. For example, one person might always use a coffee machine to make coffee, while another uses instant coffee.

The description of action primitives, actions and activities shows there exists a hierarchy in activities. Understanding the structure of this hierarchy can play an important role in the recognition of activities. This thesis includes approaches that take the entire hierarchy described above into account, as well as approaches in which only part of the hierarchy is considered.

1.3 Observing Action Primitives

To allow automatic health monitoring on a large scale, existing homes will need to be equipped with sensors. An overview of the most common sensing modalities used in activity recognition can be found in Chapter 2, Section 2.2. In this thesis, we use sensor networks to observe action primitives performed by inhabitants in their homes. Sensor networks provide a non-intrusive, privacy-friendly and easy-to-install solution to in-home monitoring. Sensors used are contact switches to measure whether doors and cupboards are open or closed; pressure mats to measure sitting on a couch or lying in bed; mercury contacts for movement of objects such as drawers; passive infrared sensors to detect motion in a specific area and float sensors to measure the toilet being flushed. These sensors observe a large number of action primitives performed in the home, however, not all action

primitives can be observed. For example, it is possible to observe a cupboard being opened, but not which item is taken from the cupboard. The sensing method we use is therefore able to observe only a limited set of action primitives.

1.4 Recognizing Activities

Recognizing activities from sensor data means we wish to determine which activity corresponds to an observed sequence of action primitives. To do this, the following issues need to be dealt with.

Ambiguity: A single action primitive can occur during different activities. For example, the action primitive of opening the refrigerator door can occur during the activity cooking and during the activity of getting a drink. A single action primitive can therefore be ambiguous with respect to the recognition of activities. Considering a sequence of action primitives would resolve that ambiguity, since the context of other action primitives specifies with what intention a particular action primitive was performed. However, the sensor network we use is only able to observe a limited set of action primitives. Therefore, a sequence of observed action primitives can still be ambiguous with respect to the recognition of activities.

Generalization: Activities can be performed in a large number of ways, which means it is hard to formulate a generalized description for an activity. The range of possibilities can become even broader due to noise, caused either by the human performing the activity (e.g. opening the wrong cupboard) or by the sensor network (e.g. a network packet is lost). This makes determining which sequence of action primitives corresponds to which activity even more difficult.

Segmentation: Data obtained from the sensor network is unsegmented, this means that the start and end time of activities is unknown. For an observed sequence of action primitives it is therefore not known how many activities were performed during the sequence and at what point one activity ends and another one starts. Therefore, a recognition method has to consider all possible combinations of segmentation to determine which interpretation fits best.

These issues make activity recognition a very challenging task. In this thesis, we use temporal probabilistic models to deal with these issues in a principled manner. By taking the uncertainty into account with respect to all of the above issues, such models can make an informed decision on which activity is taking place based on the evidence that the sensor data provides.

1.5 Research Focus and Questions

Recent developments in sensing technology make it possible to easily equip existing homes with sensor networks. Healthcare professionals agree that activities are what they want to monitor [72]. What is missing are the pattern recognition methods to recognize these activities automatically from sensor data. In this thesis, we answer the following questions with respect to that issue:

Which temporal probabilistic model is able to accurately recognize activities from sensor network data? Designing such models requires a trade-off between model complexity, computation time and the amount of example data needed.

We present temporal probabilistic models that accurately recognize activities from sensor data. A comparison of generative, discriminative, semi-Markov and hierarchical models provides an insightful analysis to the performance of these models in the application of activity recognition.

How can we re-use methods for activity recognition in multiple homes? Pattern recognition methods for activity recognition are generally developed with a particular home in mind. Home-specific properties are either automatically learned from data or manually created by a domain expert. Differences in the layout of homes and the behavior of inhabitants prevent us from applying these methods to other homes.

We developed a novel transfer learning method that allows us to use labeled data from other homes to learn the parameters of a temporal probabilistic model for a new home.

How can we evaluate the performance of pattern recognition methods to ensure an accurate performance in a real world setting? It is very difficult to capture the complex structure of activities, actions and action primitives realistically in a simulated setting. Recording real world sensor data would avoid this issue, but establishing a ground truth is expensive and difficult to collect accurately.

We created a sensor and annotation system for recording data sets at home. The sensor system can easily be installed in existing homes and the annotation system provides an inexpensive way of recording an accurate ground truth. Several real world data sets were recorded using this system and are publicly available to the research community providing a standardized benchmark for the community to compare their results on [67]. These datasets are now widely used within the community [4, 92, 100, 131, 137, 170].

1.6 Overview

Chapter 2 gives an overview of activity recognition systems in the literature. We discuss applications that use activities, sensors that are used in activity recognition, pattern recognition methods for recognizing activities from sensor data and datasets for activity recognition. We describe state of the art developments and discuss open problems and issues with respect to these topics.

Chapter 3 answers some fundamental pattern recognition questions with respect to activity recognition. Issues with respect to discretization and feature representation are discussed and solutions to these issues are proposed and empirically supported. We compare the recognition performance of two of the most basic temporal probabilistic models. The two models used present an important dichotomy in the field of temporal probabilistic models, namely the choice between generative and discriminative models. We explain the differences between these types of models and discuss their usability with respect to activity recognition on the basis of their recognition performance, the amount training data used and the required computation time for processing sensor data.

Chapter 4 discusses the consequences of the Markov assumption and presents semi-Markov models that relax this assumption. The Markov assumption is a powerful assumption which allows us to perform inference in temporal probabilistic models in linear time. However, this assumption is too strong for problems such as activity recognition in which there are temporal dependencies over long periods of time. Semi-Markov models provide a solution for modeling long term dependencies, but require computationally expensive inference algorithms. We compare conventional Markov models with their semi-Markov counterparts and discuss their impact on the computation time and recognition performance.

Chapter 5 presents hierarchical models to model the internal structure of activities more accurately. The use of a hierarchy allows abstraction of the data on multiple levels. We compare two approaches to modeling activities using hierarchical models and compare the recognition performance of these models to the performance of the Markov and semi-Markov models.

Chapter 6 introduces a method for applying activity recognition models in multiple homes without the need for labeled training data from each home. The models used in activity recognition require labeled data to learn the model parameters. A model trained for one home cannot automatically be used in another home due to differences in the layout of the home and the behavior of the inhabitant. We present a transfer learning method that allows us to use labeled data from other homes to learn the model parameters for a new home.

Chapter 7 summarizes the conclusions drawn from this thesis and discusses future directions in activity recognition research.

2

Activity Recognition: Related Work

Traditionally computers are used in a desktop setting for office and study related tasks such as text processing and spreadsheets. Two developments are moving computers out of the desktop setting and into our daily lives. First, the miniaturization of computers and the introduction of mobile phones has made computers available to the user at anytime and anywhere. Second, novel computer systems are enriched with all sorts of sensors to measure properties of their environment. This novel computing paradigm is known under a variety of names namely: pervasive computing, ubiquitous computing, ambient intelligence and context awareness.

This new trend in computing allows for many different branches of applications that focus on various aspects of our lives. Applications for these systems rely on different kinds of contextual information, such as location, identity of the user, and the activity a user is performing [1]. The location of a user can be the country that he is currently in, but can also be the room number within a building that he is in. User identity can involve a single user or multiple users, and can be a source for other contextual information such as friends and relationships. In this work, we focus on the activity that a user is performing.

In this chapter, we provide an overview of related work in the area of activity recognition. We start, in Section 2.1, with a discussion of applications in which activity is used as contextual information. In Section 2.2, we describe the different sensing modalities that are used to perform activity recognition. Section 2.3

[1]The material in this chapter is partly drawn from [69, 70].

reviews pattern recognition methods for recognizing activities from sensor data. Section 2.4 gives an overview of various activity recognition datasets.

2.1 Applications using Activities

In this section, we discuss applications that rely on activity recognition. We discuss surveillance applications, office applications and health applications in general. These applications show that activity recognition is broadly applicable and is not only useful for health monitoring. Discussing these applications will show that there are different forms of activity recognition and this will allow us to clearly define which form of activity recognition we are interested in for health monitoring purposes.

2.1.1 Surveillance Applications

The increasing need for safety in our society has resulted in a large number of video cameras present in many public locations. Because it requires a lot of manpower to monitor all these cameras, there is a need for surveillance systems that can automatically recognize suspicious behavior. Such systems can either automatically sound an alarm, or attract the attention of an operator when suspicious activities are taking place. A large variety of potential applications exist that automatically detect unattended luggage [7, 103, 168], intruders of a building [23, 60], aggression on a train station [181], car theft or theft of handbags [35, 95], bank robberies [43] and drowning in a swimming pool [61, 90].

These applications can rely on activity recognition because they need to recognize activities in order to detect deviations from a regular pattern of behavior. It is normal for people to open their cars, to visit a bank or leave their luggage unattended for a short period of time. Detecting deviations from such normal behavior relies on knowledge of the normal execution of an activity. For example, smashing the window of a car to open the car door is clearly not normal behavior for a car owner. Alternatively, these applications can focus directly on detecting deviations from a normal pattern of behavior, rather than recognizing a set of activities. If sufficient amount of training data is available, a model can be trained to distinguish normal behavior from deviant behavior without the need for interpreting which activities were performed. This means that surveillance applications do not necessarily require the accurate recognition of individual activities.

Surveillance applications are generally installed in public places, therefore such systems should be able to deal with multiple people, who may or may not be interacting. They can be installed in an outdoor setting or indoors in a public

building and are likely to allow a lot more unexpected or irregular events than private spaces. Because the purpose of surveillance systems is to act quickly in case of an emergency, such systems should process the sensor data in an online manner. This means that recognition is performed as soon as the sensor data is observed. The alternative is to process data offline, which means a collection of data (e.g. a full day) is collected and processed at the end of the day. In terms of sensors, surveillance application typically use cameras for sensing the environment. Cameras are already installed in many locations. Furthermore, the use of cameras allows for a gradual transition from a system that assists a human operator, to a system that runs completely automatic using an intelligent algorithm.

2.1.2 Office Applications

The introduction of activity-aware systems in an office setting can result in cost reduction and a more efficient workflow. An example is an energy-efficient power-management system. Current energy saving devices, such as motion activated lights, tend to be disabled by users because they are annoying. Instead, an activity recognition system could use more targeted information to better match the expectation of the user. For example, recognizing when someone goes to a meeting [46, 176].

Meeting rooms also allow for interesting applications. On the one hand, activity recognition systems can provide support in the form of intelligent lighting and beamer control. On the other hand, a vision system can be used for automatic annotation of the meetings to allow powerful querying [18, 99, 144].

These application are clearly designed for indoors and generally need to be able to deal with multiple people to achieve their full potential. Although offices often have a public area, the applications described above are mainly targeted at locations in offices that are restricted to personnel. Such areas typically show very regular patterns of behavior (e.g. a standard lunch hour) and irregular events are less common than in public spaces. Due to the interactive nature of the described applications, sensor data should be processed in an online manner. In terms of sensors, both cameras and sensor networks are used in an office setting. Although cameras in an office room might raise some privacy issues, they are well accepted in hallways.

2.1.3 Health Applications

Besides the health monitoring applications discussed in Section 1.1, activities are also used in applications for assistive and persuasive technologies.

Assistive technology refers to applications designed to help people in living their daily lives. Activity recognition systems are used to assist people suffering from dementia, who tend to forget certain steps while performing an activity. For example, when making coffee they might install a coffee filter, but forget to add coffee powder. An activity recognition system can assist these people by recognizing the activity they are performing, recognizing which actions are performed and reminding them which action primitives to perform to complete the activity. Audio cues can be used to guide the person in performing any missing steps [50, 104] or a display can be used to show images of the actions that need to be performed [89, 113].

Persuasive technology motivates people to change their behavior, such as leading a healthier life style. One way is to provide users with well-timed information when they have to make decisions concerning their health. For example, to reduce the chances of obesity, a system can provide diet suggestions when it detects the user is preparing dinner. The appropriate timing of such a message is crucial to the success of such a system [56]. Another approach is to use a reward when the user is living a healthy life style. For example, in a study by Consolvo et al. [24], users were given an exercise program and a mobile phone showing the image of a virtual garden. If they spent enough time performing exercises from the exercise program, they received a visual reward in the form of flowers appearing in the virtual garden. The amount of time spent on exercises by users of the persuasive system was compared to participants that did not use the system. Participants using the system were shown to spend significantly more time on exercises than participants that did not use the system.

These applications can target both in-home situations as well as a combination of in-home and outdoors (e.g. completing an exercise program outdoors). Assistive applications rely on the recognition of which actions are performed and which actions are skipped in the execution of an activity. This requires a thorough understanding of how activities are performed. Both assistive and persuasive applications need to interact with the user and therefore require sensor data to be processed online. Although it is possible to apply such technology to multiple people, they are typically designed to assist or persuade a single person. Sensors used are cameras, sensor networks and wearables.

2.1.4 Conclusions

We have seen that the automatic recognition of activities allows a broad range of applications and that there are many different forms of activity recognition. Applications are either single person or multi person oriented. Multi person activity recognition is more challenging because a source separation task needs to be performed. This means the system should be able to determine which person triggered which sensors. In this thesis activity recognition is discussed in

the context of health monitoring elderly. For this purpose, we assume that a single person is present in the home at all times. Multiple person activity recognition would make the health monitoring system more broadly applicable, but single person households are likely to have a higher need for health monitoring. In a multi person household, the people might still be able to look after each other. Furthermore, it is easier to study the effects of recognition when only a single person is generating sensor events.

Activity recognition can be performed indoors or outdoors. An indoor setting is generally more constrained than an outdoor setting, making the chance for irregular events smaller. For the application of health monitoring, we assume all activities are performed indoors and recognize the activity of leaving the house as a single activity, regardless of the purpose the person is leaving the house with. Including outdoor activity recognition is something that can be considered at a later stage.

Sensors used in the applications are cameras, sensor networks and wearables and the data obtained from sensors can be processed in an online or offline fashion. Most applications use online processing because they rely on interaction with the user. For health monitoring purposes offline processing is sufficient, because we are monitoring for long term purposes. Sensor data can be collected for the duration of an entire day, after which the system recognizes which activities were performed during the day. A system developed for offline processing can generally be easily adjusted to perform online processing, but this does result in a decrease in recognition performance. All the results in this thesis are obtained using offline inference.

2.2 Sensors for Activity Recognition

The type of sensors used is an important aspect of designing an activity recognition system. The following three factors are of interest. First, because we are sensing inside the private house of a person, it is important to evaluate how intrusive the inhabitant experiences the sensors. Some sensors need to be worn on the body, which might be considered inconvenient by the user. Other sensors may simply be considered intrusive because they are constantly in sight, or because of the amount of privacy sensitive information they record. Second, the ease of installation is important. Sensors that are easy to install can increase the acceptance of the system and sensors that can be installed without the need for a technician can significantly reduce the cost of a system. Third, what the sensor measures is an important factor. If a sensor is able to sense all possible action primitives that make up activities, it would provide a good basis for the recognition of activities.

In this section we discuss the most commonly used sensing modalities discussed in the literature of activity recognition. We discuss cameras, wearables, RFID and wireless sensor networks and discuss the different factors described above for each of these modalities.

2.2.1 Cameras

Cameras provide high-dimensional image data which are very informative of the action primitives that make up an activity. However, automatically recognizing action primitives from video data is hard. There are many surveys available discussing the recognition of action primitives, actions and activities from image data [2, 42, 107, 129, 166]. A recent survey on computer vision concluded that the recognition of action primitives from video data is a very challenging task and therefore most work in computer vision currently is limited to the recognition of action primitives rather than actions or activities [107].

Instead of accurately recognizing every action primitive performed, certain features related to action primitives can be extracted from images and can be used as a basis for activity recognition. For example, the location of a person can be indicative of which activity is performed. In work by Duong et al. multiple cameras installed in the corners of the room observe a person performing activities. The room is divided into square cells of 1 square meter in size and a multi-camera tracking module detects movement and returns a list of cells visited by a person. Some cells contain an object of interest (e.g. fridge or stove) so that when a person visits a cell it is highly indicative that an action primitive related to the object in place is performed (e.g. opening the fridge or using the stove). No recognition of the action primitives is done, but the list of cells visited is used as input for activity recognition [33]. A similar approach uses a camera with a top down viewpoint, making it easier to divide the image into squared regions of interest and detecting the location of a person [119, 160].

Instead of using location, the object a person is using is also a good indicator of the action primitive a person performs. Typically a single camera is focused on a particular area of interest where activities are performed, such as the sink in a bathroom or the kitchen cooking area. In work by Wu et al. image data is processed to detect which objects a person uses. The detected objects are used to recognize activities such as making tea or taking medicine [177]. Messing et al. use various salient visual features extracted from the image and calculate the velocity of these features to track the movement and position of the hands. This data allows them to distinguish between actions such as drinking from a cup and pealing a banana [102]. Instead of using a static camera it is also possible that a camera is worn on the shoulder to observe the interaction between the hands of the user and various objects. In such an approach it is possible to observe activities in multiple locations using a single camera [98, 152].

The highly informative data obtained by cameras allows for many different kind of approaches to activity recognition. However, this highly informative data also contains a lot of privacy sensitive information. Both elderly and their caregivers have expressed concerns with respect to the use of cameras for home health monitoring [6, 175].

2.2.2 Wearables

'Wearables' refers to the collection of sensors that one wears on the body or in their clothes. Obtained sensor data is either processed on the sensing board itself or transferred wirelessly to a computer or mobile phone for further processing. The most common wearable sensors are 2- or 3-axis accelerometers to measure the magnitude and direction of acceleration [77, 116, 122, 182]. Often additional sensors are included such as light sensors, humidity sensors, heart rate sensors, gyroscopes and compasses [84, 127, 161]. Sensors are typically stored compactly in a single device that users can carry with them. There are also approaches in which the sensors are distributed over the body to allow specific measurements on each limb [93]. While designing a sensor system, there is a trade-off between user comfort and usability of the obtained sensor data. A study on the acceptance of the location of wearables considered the wrist, hip, and chest as potential locations. They found that the hip and the chest were generally considered as the most acceptable locations [122]. Another study compared the performance of recognizing various types of movements such as running, walking and standing, with respect to the location of the wearables [97]. Sensor hardware was placed on the wrist, the belt, on a necklace, in the right trouser pocket, in the shirt pocket, and in a bag. Sensors on the wrist and in the bag generally outperformed the sensors placed in the other locations. To increase the ease of use, sensors can be integrated into a mobile phone [12] or in clothing [34]. In a jacket specifically designed for health monitoring, sensors were integrated into the clothing to perform electrocardiography (ECG) monitoring and capture accelerometer data for recognizing user activities [143].

The data obtained from wearable sensors mainly provides information about the pose and movement of a person. This data is typically used to recognize types of movement, such as running or walking. Due to the limited information about the location of the user and the objects used, it is very difficult to recognize activities. Some activities that have very distinctive body movement, such as vacuum-cleaning the house and drinking from a cup, are recognized using wearables [127]. To increase the range of activities that can be recognized, the use of wearables can be combined with additional sensing equipment. For example, installing ultrasonic listeners in an environment, combined with an ultrasonic tracker on the body, allows indoor localization [37]. Stiefmeier et al. used this technique and placed trackers on the hands to recognize maintenance activities

such as repairing a bicycle [148].

Although the recognition of activities using wearables is challenging, they are commonly used in fall detection [29]. A potential downside of wearables is that people might forget to wear the sensing unit or find it uncomfortable to wear them. The installation of a wearable sensor (i.e. wearing the wearable) needs to be performed on a daily basis and can be experienced as easy when the sensors are stored in a compact box, or tedious when sensors have to be distributed over the body.

2.2.3 RFID

Radio frequency identification (RFID) is a technology for reading information from a distance from so called RFID-tags. There are two types of tags, passive RFID-tags and active RFID-tags. Passive tags extract energy from the radio frequency signal emitted by an RFID reader and use that energy to send a stream of information back. The amount and contents of information can vary, but usually contains at least a unique identification string. Active tags are equipped with a battery used as a source of power for the circuitry of the tag and its antenna. This makes it possible to read the tags from a much larger distance than passive tags [28].

Chip manufacturer Intel used RFID technology to develop a sensing method used specifically for activity recognition. Their product, named the iBracelet [80], is a RFID reader in the form of a bracelet which the user wears on a single hand or on both hands. By tagging a large number of objects in the house with passive RFID tags the iBracelet is able to observe which objects users are holding in their hands. Objects are very indicative of the action primitive a user is performing, therefore making activity recognition possible [38]. The task of tagging all objects in the household may seem tedious, but it is expected that passive RFID tags will replace barcodes in the future. Because the tags can contain detailed information about the product they are tagged to, this would allow the iBracelet to easily observe any product used [108].

A difficulty in using the iBracelet is that it often fails to detect a tag if it is only briefly in the range of the RFID reader. In a comparison between various sensing modalities used for activity recognition, the iBracelet performed by far the worst due to this limitation [88, 150]. RFID readers have been combined with wearable accelerometers to measure both the objects used and the movement of a person. This way both sensors can compensate for the shortcomings of each other, resulting in better activity recognition performance [51, 149].

2.2.4 Wireless Sensor Networks

A wireless sensor network consists of a collection of small network nodes. Each node is capable of performing some processing, gathering sensor measurements and communicating wirelessly with other nodes in the network [147]. Nodes are designed to be as small as possible and therefore usually have a working memory of only several kilobytes. Specifically engineered operating systems have been developed for dealing with such conditions, such as TinyOS [49]. Although the term 'wireless sensor network' is broadly applicable, in this work we use the term to describe a network of simple sensors with binary output which are installed in fixed locations.

The nodes generally run on batteries and therefore a lot of research is devoted to energy efficiency. The communication between these nodes typically requires little bandwidth and is relatively insensitive to latency, so that energy efficient communication protocols are possible [27]. It is also possible to save power by shutting down parts of the node when these are not in use [145]. The use of ad hoc routing protocols allows the nodes to dynamically form a temporary network without any pre-existing network infrastructure or centralized administration [13]. Such routing protocols also allow further power saving schemes, by shutting down nodes strategically and avoiding them in the network route [179].

A large variety of simple sensors can be incorporated in the network nodes. In a house setting, it is possible to use various sensors to measure different things: contact switches for open-close states of doors and cupboards; pressure mats to measure sitting on a couch or lying in bed; mercury contacts for the movement of objects such as drawers; passive infrared sensors to detect motion in a specific area; float sensors to measure the toilet being flushed; temperature sensors to measure when the stove is used; humidity sensors to measure when the shower is used [71, 176] and accelerometers to detect when a large object is used [96, 156]. Basically any kind of sensor can be combined with a network node as long as its output can be converted to a binary output. For example, in work by Fogarty et al. microphones were attached to water pipes in the basement to record whether water was flowing [39]. Such an approach requires a minimal installation effort and allows the system to use an informative property such as when water is being used.

Wireless sensor networks can be used to determine whether an object is used. However, in contrast to RFID tags, it is difficult to equip small objects, such as a tooth brush or a dinner plate, with a sensing node. Wireless sensor networks can therefore only sense a limited set of action primitives. For example, it is possible to observe that a cupboard is opened, but not which item is taken from the cupboard.

Nonetheless, the use of wireless sensor networks in a home setting offers many

advantages compared to the other sensing modalities. First, the majority of the sensors can be installed out of sight of the inhabitants, therefore limiting the intrusiveness of the sensors. Second, the data recorded is anonymous and contains very limited privacy-sensitive information. Third, installation of the sensors can be done quickly without any need for the installation of power and network cables.

2.2.5 Conclusion

None of the sensing modalities discussed above is able to provide information that can perfectly identify every action primitive that is performed in a house. In the case of cameras, the processing of raw image data is still too challenging for detecting all possible action primitives. Wearables simply provide too little relevant information for the recognition of activities. RFID readers often fail to observe tags due to their limited sensing range. Wireless sensor networks can only sense a limited set of action primitives. These limitations are complementary to a point, however, and the most informative solution would therefore be to combine several sensing modalities. However, due to issues with intrusiveness and ease of installation, this is not a suitable solution for home health monitoring. Instead, in this thesis, we use a wireless sensor network consisting of simple sensors with binary output to observe the action primitives of volunteers in their own house. The non-intrusiveness, privacy-friendly nature and ease of installation make them ideally suited for the use in home health monitoring.

2.3 Pattern Recognition Methods for Activity Recognition

The use of our proposed wireless sensor network gives us binary temporal data from a number of sensing nodes that observe the action primitives performed in a home setting. Recognizing a predefined set of activities from sensor data is a classification task. In such tasks, features are extracted from the space and time information present in the sensor data and used to classify the data. Feature representations are used to map the data to another representation space with the intention of making the classification problem easier to solve.

In this section, we discuss the most commonly used pattern recognition methods in activity recognition. We start with a description of feature representations, then discuss temporal data-mining methods, logic based methods, temporal probabilistic models and common sense mining. We describe these methods in detail and explain how they address the challenges in activity recognition, namely the issues of ambiguity, generalization and segmentation.

2.3.1 Feature Representations

The sensor data obtained from the wireless network is multivariate and binary valued, with a continuous time dimension. Such temporal data can either be represented as a time series or as a sequence of events.

Let a time series be a sequence of data points measured in succession and discretized using a constant time interval. The data points of a time series are often referred to as timeslices and in our case each timeslice consists of a binary vector of sensor readings. We assume that an activity is performed at all times, and that no two activities can be performed simultaneously, so that each timeslice is associated with a discrete activity label indicating which activity is performed during that timeslice. The task of a pattern recognition method is to determine which activity label should be assigned to a particular timeslice. Choosing a proper time interval for discretization is important to the recognition process. Too small an interval will incorporate noise of the signal, while too large an interval will smooth out important details of the signal. For example, a passive infrared sensor measures changes in infrared radiation and is generally used to pick up the movement of a human being. We would like the discretization of such movement to correspond closely to the action primitives that are performed. Discretizing with a small interval might cause a single action primitive to be broken down into multiple action primitives, because of small pauses in the movement of the human being. On the other hand, too large an interval would discretize multiple action primitives into a single action primitive. In the case of activity recognition from wireless sensor networks, no existing work provides any theoretical or empirical support to favor any interval length. Besides discretization certain pre-processing steps are often applied to transform raw time series data to a different representation, which makes recognition of patterns easier [41]. For wireless sensor network data, there is no known transformation that yields a better performance in activity recognition, than using the raw sensor data.

Temporal data can also be represented as a sequence of events. In the case of binary sensor data, every change in state of a binary sensor generates an event. We can assign a single symbol, to both represent the change in sensor state from off to on, and from on to off, or use two different symbols for each type of change in state. The use of events can avoid the need for discretization. If we assume that the end point of an activity and the start point of the next activity always coincide with a sensor event, an activity recognition algorithm only needs to process the events and determine whether an activity ended, and if so which other activity started. However, this might be an unrealistic assumption and we might want an activity to start and stop at any point in time (i.e. even if no sensor event took place). The pattern recognition method will therefore need to be able to deal with continuous time. Furthermore, since the elapsed time between two events is not constant, a pattern recognition method will probably need to incorporate

the duration of an activity to avoid unrealistically long activities.

2.3.2 Temporal Data Mining Methods

Temporal data mining methods are unsupervised pattern recognition methods that automatically find recurring patterns in sequential data. We discuss two approaches that are relevant to activity recognition.

The first approach automatically finds templates in noisy sensor data. For example, it is possible to automatically detect recurring words in audio speech data [120]. Such techniques can be applied to automatically detect activities in sensor data. Typically no prior knowledge is provided to the algorithm about any template of interest, so that an exhaustive search easily becomes computationally intractable. The search space can be limited by first transforming the raw sensor data to a discrete symbolic representation and searching for similar pairs of subsequences in the symbolic data [21]. In work by Thad et al., this approach was used to automatically detect activities in sensor data obtained with wearable sensors. A probabilistic model was trained using the Expectation Maximization (EM) algorithm to automatically detect recurrent activity patterns [161].

The second approach looks for correlations between events in a large collection of data. By finding sensors that often both change state within a certain time window, it is possible to automatically identify activity patterns. This approach is used in a technique called T-patterns [94]. In this technique, we assume that each event is independently and randomly distributed over time. From a given collection of data, we can calculate the observed average frequency of each event. Two events A and B form a T-pattern if after an occurrence of A at time t there is an interval $[t + d_1, t + d_2]$, with $(d_2 \geq d_1 \geq 0)$, that contains at least one more occurrence of B than would be expected by chance. When detecting T-patterns in data every possible pair of symbols is tested, using every possible interval, from the largest to the smallest one. If a T-pattern is found, the pattern itself is added to the data sequence as a single event. This allows the detection of hierarchical T-patterns (i.e. T-patterns formed out of T-patterns). In work by Tavenard et al., T-patterns were applied to automatically detect movement patterns in sensor network data [159].

The advantage of these temporal data mining methods is that they automatically find recurrent patterns without the need for any labeled data. However, because there is no labeled data to indicate which patterns are of interest to the user it is unsure whether the patterns found are of any use. These data mining techniques therefore often speak of activity discovery, rather than activity recognition.

2.3.3 Logic-Based Methods

Activity recognition in the logic community was first defined as plan recognition, which aims to infer the plan of an agent from observations of the action primitives of the agent. For the purpose of our discussion, we can consider the definition of a plan to be equivalent to our definition of activity. The most popular work in the area of plan recognition was done by Kautz [73] who formalized the problem using McCarthy's circumscription theory and aims to explain a sequence of observations using a minimal number of plans. For example, consider observing an action primitive A_1 which can be part of plan P_1 or P_2. If we then observe an action primitive A_2 which can only be part of plan P_2 or P_3 it follows that A_1 and A_2 are both part of P_2 [19].

Kautz assumes in his model that the person (agent) performing the activities acts rationally, therefore his model is not able to deal with noise. We have already explained how humans can generate noisy sensor readings by opening a wrong cupboard. People suffering from Alzheimer's disease are even more prone to generating noisy sensor data, because they often fail to complete an activity, or continue to complete a different activity than they initially started performing. This has led to a large body of work in which logic models perform activity recognition and assume an irrational agent. Bouchard et al. [11] used lattice theory and action description logic to recognize patients performing incoherent activities. Their framework is able to dynamically generate new plausible plans which were not defined beforehand.

Chen et al. [20] use event calculus to recognize activities and assist inhabitants in performing the correct action at the correct time and place. The event calculus ontology consists of events and fluents. An event in this case is the observation of an action primitive by the sensor network (e.g. add(water, kettle)), while a fluent is any property of a world that can change over time (e.g. inside(water, kettle)). Predicates are then used to define relations between ontology entities (e.g. initiates(add(water, kettle), inside(water, kettle))). Activities are defined by combining several predicates. An inhabitant profile is used to define normal behavior of the inhabitant, if the system detects deviations from this normal behavior it can activate a reminding service indicating which event should occur. For example, if an inhabitant typically completes the activity of making tea at a certain hour each day and the necessary action primitives are at one point not observed at that hour, it can suggest to use the kettle and add water.

Rather than recognizing normally performed activities, Rugnone et al. [138] use temporal logic to recognize deviations from normal activity behavior. Their syntax consists of temporal operators such as S (since) and H (historically), which are used in combination with a temporal interval to indicate the time constraint within which an activity is typically performed. For example, the sentence 'the subject has been sleeping for the past 8 hours' can be expressed as $H^{[8]}\gamma_1$, where

γ_1 represents 'the subject sleeps'. The rules of the system are entered manually by health care professionals. A visual editor is proposed which allows a more intuitive method for inputting rules. Deviations are detected by violations of predefined constraints. These constraints are hard-coded using explicit threshold value (e.g. no longer than 5 minutes). This approach allows the user to specify abnormal behavior without the need for specifying normal behavior.

Landwehr et al. [81] are inspired by transformation-based learning, primarily used in natural language processing. Because the structure in natural language is more rigid compared to the structure of activities they extend existing transformation-based learning with more flexible relational transformation based learning. By manually adding relations expressing time constraints, it is possible to disambiguate the observed sensor data. For example, $close(id, obj, t)$ means object obj was observed within t seconds of event id. Using a greedy search algorithm, rules that minimize the recognition error are iteratively added to the system.

The advantage of these logic-based methods is that it is very easy to manually incorporate domain knowledge, therefore reducing the need for annotated training data. Even if the method is able to automatically learn how to recognize activities from data, the rules on which this classification is based are still easy to interpret. An important disadvantage of these methods, however, is that if any ambiguity remains about which activity is performed the methods are unable to express which of the activities is most likely.

2.3.4 Temporal Probabilistic Models

Here we consider discrete-time temporal probabilistic models for modeling sequential data. A sequence of observations denoted as $x_{1:T}$ denotes a collection of data observed from sensors. The goal is to infer a matching sequence of hidden states $y_{1:T}$, which in our context correspond to the performed activities. Temporal probabilistic models solve this problem by finding the sequence of states that maximizes the probability of the activities given the sensor readings, $p(y_{1:T} \mid x_{1:T})$. There are two types of models, generative and discriminative probabilistic models. Generative models fully define the joint probability $p(y_{1:T}, x_{1:T})$ and can be used to generate (sample) data from this distribution, or to perform inference given a novel sequence of observations. Discriminative models are solely used for inference and therefore define the conditional probability $p(y_{1:T} \mid x_{1:T})$ directly. We will further highlight the differences between generative and discriminative models in the upcoming chapters.

The most popular generative temporal probabilistic model is the hidden Markov model (HMM). It is based on the first-order Markov assumption, which states that the current state y_t only depends on the previous state y_{t-1}. This assumption

makes it possible to perform inference in the HMM in linear time. A popular discriminative temporal probabilistic model is the conditional random field (CRF). Applying the Markov assumption to this model gives us the linear-chain CRF. The HMM and linear-chain CRF have been successfully applied in activity recognition [53, 71, 124, 174]. We will discuss these models and their related work in Chapter 3.

There exist many extensions to these models in which specific properties of the data are modeled explicitly. In particular, semi-Markov models explicitly model the duration of a state. The accurate modeling of the duration of an activity can be an important property in distinguishing between activities when sensor data alone does not provide sufficient discrimination. For example, sensors in the bathroom might observe a person present in the bathroom, but cannot observe whether this person uses the shower or a tooth brush. Taking a shower typically takes longer than brushing teeth and therefore the amount of time spent in the bathroom is indicative of the activity performed. In Chapter 4 we present the hidden semi-Markov model (HSMM) and semi-Markov conditional random field (SMCRF) applied to activity recognition and discuss their related work.

Hierarchical models extend on conventional sequential models by explicitly modeling the hierarchical structure in the data. We present a novel hierarchical model for activity recognition in Chapter 5. This model is based on the hierarchical structure of activities that was introduced in the introduction chapter of this thesis.

The advantage of temporal probabilistic models is that for many of these models there exist efficient algorithms that can perform inference in linear time with respect to the length of the sequence. The disadvantage is that a large amount of annotated training data is needed to learn the model parameters accurately.

2.3.5 Common Sense Mining

Probabilistic models require large amounts of annotated data to learn their model parameters. This need for annotated sensor data can be eliminated by using external information on how activities are performed, for example by using information obtained from the world wide web. This form of data collection has only been used with RFID, because RFID provides a precise sequence of used objects which can be mapped directly to an online description [125, 126, 157, 178]. A probabilistic model is used which calculates the probability of a sequence of objects indicating an activity. The list of objects involved in a particular activity are mined from the web using websites containing recipes, user manuals and 'how-to' articles. These websites contain explicit step-by-step instructions on how activities are performed. The objects are filtered out by querying WordNet for each word and keeping terms that have object or substance as hypernyms. The probability that an object is used during an activity is calculated using the

number of search results returned by Google search engine [126]. For example, in determining whether a cup is used when making tea, the object probability $p(object \mid activity)$ is calculated as

$$p(\text{cup} \mid \text{making tea}) = \frac{gc(\text{making tea cup})}{gc(\text{making tea})}, \tag{2.1}$$

where $gc(s)$ is a function return the number of Google search results for the words in string s.

Wyatt et al. extended this work by not only considering recipes and how-to sites, but querying a search engine for relevant pages. The webpages found by the search engine are classified as relevant or not relevant using a support vector machine. They compare their mining method to learning model parameters from training data. Their approach sometimes outperforms the learned models, but overall the learned models do better [178].

Pentney et al. mine their activity definitions from the Open Mind Indoor Common Sense (OMICS) database. OMICS is a common sense knowledge base to which all users on the web can contribute. Examples of information stored in OMICS include 'you eat when you are hungry' and 'after dinner you wash the dishes'. Information is stored using predicates (e.g. people('eat', 'are hungry'). Because anyone can add information to OMICS, the authors use the number of search results returned by a search engine to measure how reliable a particular proposition is [125].

Tapia et al. showed that a technique called shrinkage can be used to improve estimates of parameters learned from the web. First, a tree-like ontology of objects is created using WordNet. The parameters of an object are then updated by calculating a weighted sum of the leaf nodes of the object in the tree structure. This shrinkage technique is shown to significantly improve performance in the recognition of activities [157].

2.3.6 Conclusion

Using time series to represent the sensor data raises questions with respect to the length of the time interval to discretize the data with, and which transformation method should be used to best represent the data. The use of events avoids these questions, but requires the pattern recognition to process timestamps to obtain a notion of time.

We have seen that temporal data mining methods perform activity discovery, rather than activity recognition. Logic based methods can be used for recognition and make it possible to easily include domain knowledge, which reduces the need for annotated data. However, when the rules of logic cannot decide on a single

activity there is no measure for expressing which interpretation is most likely. Temporal probabilistic models provide a way for dealing with such ambiguous cases, but require large amounts of annotated training data to learn the model parameters. Common sense mining can be used to learn model parameters from online data sources, but has only been applied to RFID data because it requires precise and unambiguous measurements of the used utensils.

In this thesis, we chose to use models that rely on a time series representation. Several well understood models, such as the hidden Markov model, exist for this type of representation and this provides a good starting point for applying temporal probabilistic models to activity recognition.

2.4 Datasets for Activity Recognition

To evaluate the performance of pattern recognition methods for activity recognition, datasets are needed. Ideally real world datasets are used which are fully annotated with a ground truth. To this end many research groups have created their own test houses equipped with sensors to record datasets and perform experiments. In pervasive computing, houses equipped with sensors are called smart homes.

In this section, we review the smart home projects of various research groups, discuss the publicly available datasets they produced and describe a sensor and annotation system used by our research team to record our own datasets.

2.4.1 Smart-Home Projects

The Aware Home[1] is a project at the Georgia Institute of Technology. A house consisting of two identical and independent living spaces was built especially for the project. Cameras are present in the house for observing the inhabitants and a smart floor is developed which senses people walking across the floor [74].

At the University of Florida, the Gator-Tech smart-house[2] was built as a laboratory for research on assisting elderly. The house has many technological features such as a smart mailbox which senses the arrival of mail and a smart refrigerator which monitors food usage [47].

MIT's PlaceLab[3] is the most comprehensive smart-home in terms of sensing technology. The house contains hundreds of sensing components installed in nearly every part of the home. Cameras, microphones, contact switches, temperature

[1] Aware Home: `http://awarehome.imtc.gatech.edu/`
[2] Gator-Tech: `http://www.icta.ufl.edu/gt.htm`
[3] PlaceLab: `http://architecture.mit.edu/house_n/placelab.html`

and humidity sensors were all installed during the construction of the home. The home is occupied by volunteer subjects who agree to live in the home for varying lengths of time [57].

2.4.2 Publicly Available Datasets

Publicly available datasets are important for a research community to create standardized testbeds for evaluating the performance of pattern recognition methods. Unfortunately, due to privacy issues and high annotation costs not many datasets for activity recognition are made publicly available.

Out of the datasets that are publicly available the majority are recorded in a laboratory setting. Typically a kitchen is equipped with a large number of sensors and several subjects are asked to perform a set of activities. Although these datasets contain interesting examples of how the same activity is performed by different subjects. They do not provide a good testbed for evaluating real world activity recognition, in which subjects have the freedom to execute any activity whenever they want.

A dataset recorded in the MIT PlaceLab does meet this criterion and consists of four hours of fully annotated activities performed by a researcher of their team. The researcher was free to perform any activity in any sequence and at any pace. Cameras installed throughout the house recorded every move, and these recordings were later used to annotate which activity was performed at which point in time [57]. In a second recorded dataset a couple stayed in the PlaceLab for over two months. Unfortunately, there was only enough funding to annotate one of the two person's activities [88].

MIT also developed a portable sensing kit allowing them to easily install wireless sensor networks in existing homes [58]. Two datasets were recorded using this kit. A large number of sensors was installed throughout the houses and recorded data for two weeks. Subjects annotated their activities by using a PDA which asked them every 15 minutes which activity they were performing. Unfortunately, this annotation method resulted in poor quality labeling of the data, therefore the researchers used the sensor data to further determine which activity was performed at which point in time [158].

2.4.3 Our Datasets

The publicly available datasets either contain too little data or contain incomplete or poor quality annotation. In this subsection we describe the sensor and annotation system that we used to record and annotate our own datasets. We

first describe the sensor system, then the annotation method and finally we give a description of the houses and the datasets recorded in them.

Sensors Used

In this work we use wireless sensor networks to observe the behavior of inhabitants inside their homes. After considering different commercially available wireless network kits, we selected the RFM DM 1810 [135] (Fig. 2.1(a)), because it comes with a very rich and well documented API and the standard firmware includes an energy efficient network protocol. Special low-energy consuming radio technology together with an energy saving sleeping mode result in a long battery life. The node can reach a data transmission rate of 4.8 kb/s, which is enough for the binary sensor data that we need to collect. The kit comes with a base station which is attached to a PC through USB. A new sensor node can be easily added to the network by a simple pairing procedure, which only involves pressing a button on both the base station and the new node.

The RFM wireless network node has an analog and digital input. It sends an event when the state of the digital input changes or when some threshold of the analog input is violated. We equipped nodes with various kinds of binary sensors:

reed switches measure whether doors are open or closed. We installed them on doors within the homes, cupboards in the kitchen and on appliances such as the refrigerator, freezer and microwave.

pressure mats measure someone sitting on a couch or lying in bed. We installed them in the bed, couch and on chairs.

mercury contacts detect movement of objects (e.g. drawers) to which the sensor is attached. We installed them in drawers in the kitchen.

passive infrared detect motion in a specific area. We installed them in the kitchen and the shower.

float sensors to measure the fluid level in a basin. We installed them in the toilet basins and in bathroom and kitchen sinks.

Annotation

Annotation was performed by the monitored person and is either done using a hand written diary or using a bluetooth headset combined with speech recognition software. In both methods, the starting and end time of an activity is recorded.

(a) Wireless network node RFM DM 1810 to which sensors can be attached.

(b) Bluetooth headset used for annotating datasets.

Fig. 2.1: Sensing and annotation set used for recording datasets.

In the hand written diary method, the name of the activity together with its start and end time are written down on a pieces of paper distributed throughout the house in locations where activities are generally performed. The time at which the activity started and ended was read off a watch carried by the subject. A disadvantage of this approach is that writing down the activity and time is rather time consuming and interrupts the subject in performing the activity.

Annotation using the bluetooth headset method requires the subject to walk around the house with a headset at all times. When an activity is performed the subject speaks into the headset which activity is performed and whether it starts or ends. We used the Jabra BT250v bluetooth headset (Fig. 2.1(b)) for annotation. It has a range up to 10 meters and battery power for 300 hours standby or 10 hours active talking. This is more than enough for a full day of annotation, the headset was recharged at night. The headset contains a button which we used to trigger the software to add a new annotation entry.

The software for storing the annotation is written in C and combines elements of the bluetooth API with the Microsoft Speech API[4]. The bluetooth API was needed to catch the event of the headset button being pressed and should work with any bluetooth dongle and headset that uses the Widcomm[5] bluetooth stack.

The Microsoft Speech API provides an easy way to use both speech recognition and text to speech. When the headset button is pressed the speech recognition engine starts listening for commands it can recognize. We created our own speech grammar, which contains a limited combination of commands the recognition

[4]For details about the Microsoft Speech API see: http://www.microsoft.com/speech/

[5]For details about Widcomm see: http://www.broadcom.com/products/bluetooth_sdk.php

engine could expect. By using very distinctive commands such as 'begin use toilet' and 'begin take shower', the recognition engine had multiple words by which it could distinguish different commands. This resulted in near perfect recognition results during annotation. The recognized sentence is then read back to the user, using the text-to-speech engine. Any errors that do occur can be immediately corrected using a 'correct last' command.

Because annotation is provided by the user on the spot this method is very efficient and accurate. The use of a headset together with speech recognition results in very little interference while performing activities. This results in annotation data which requires very little post processing, making this a very inexpensive method.

Houses

A total of three datasets was recorded in three different houses. Details about the datasets, the houses they were recorded in and their inhabitants, can be found in Table 2.1. Floorplans for each of the houses, indicating the locations of the sensors, can be found in Figure 2.2. These datasets are publicly available for download from `http://sites.google.com/site/tim0306/`.

Datasets made public

The increasing interest in pervasive computing applications has caused many research groups to create their own smart homes or develop easily installable sensor kits. Due to privacy issues and expensive annotation costs not many real world datasets are made available to the public. To this end, we developed our own sensor system and annotation method and recorded three datasets each consisting of several weeks of data recorded in a real world setting. These datasets have been made publicly available and have been used to evaluate the recognition performance of various recognition methods such as the naive Bayes

House	House A	House B	House C
Age	26	28	57
Gender	Male	Male	Male
Setting	Apartment	Apartment	House
Rooms	3	2	6
Duration	25 days	13 days	18 days
Sensors	14	22	21

Tab. 2.1: Information about the datasets recorded in three different homes using a wireless sensor network.

(a) House A

(b) House B

(c) House C, First floor

(d) House C, Second floor

Fig. 2.2: Floorplan of houses A, B and C, the red boxes represent wireless sensor nodes (Created using `http://www.floorplanner.com/`).

classifier [59], classification using evidence theory [100, 164] and context lattice [180] and for evaluating a transfer learning method [184]. However, in many of these works the experimental setup used differs from the setup used in our work. Changes in the number of activities recognized, the feature representation used or the performance measure used for evaluation, make it difficult to compare our work directly to the results published in those works. For that reason, we have more recently published a benchmark dataset package in which we not only encourage other authors to compare their performance directly to ours using the same experimental setup, but we have also released the necessary Matlab code for reproducing the experiments [67]. Such a benchmark will make quantitative comparisons between papers easier because the published results all rely on the same experimental setup and code.

2.5 Conclusions

This chapter gave an overview of related work on activity recognition and all the elements that make up such systems. We highlighted the importance and broad use of activities in various surveillance, office and health applications.

The potential of using cameras, wearables, RFID and wireless sensor networks in a home setting for activity recognition was discussed. Each of these sensing modalities was evaluated with respect to their intrusiveness, ease of installation and informativeness with respect to activity recognition. Our choice for wireless sensor networks was explained and the challenges in recognizing activities were highlighted.

Various pattern recognition methods for activity recognition were discussed. Temporal probabilistic models provide a good framework for representing uncertainty with respect to interpreting the data.

Several smart home projects were described and the need for and lack of real world datasets was explained. Because very few real world datasets are publicly available, we recorded three datasets ourselves. These datasets have been made publicly available and are now actively used by other researchers.

3

Markov models

3.1 Introduction

Activity recognition is a challenging task for several reasons: the start and end time of the activities are unknown, activities can be performed in a large number of ways and because there is ambiguity and noise in the observed sensor data. Temporal probabilistic models provide a framework for dealing with the uncertainty caused by these issues. They are able to make an informed decision about which activity is taking place based on the evidence that the sensor data provides.

The hidden Markov model (HMM) is a classic temporal probabilistic model that has been studied for years and is very well understood. It has been successfully applied in many sequential data modeling problems such as speech recognition, handwritten digit recognition and biological sequence analysis. More recently, the conditional random field (CRF) has been shown to outperform the HMM in many of these areas when sufficient labeled data is available [44, 78, 79]. Activity recognition shares many characteristics with these problems and therefore the HMM and CRF seem suitable candidates for modeling activities using sensor data.

To apply these models, the sensor data is typically discretized into timeslices of a constant length. Often, resulting timeslice values are further transformed to obtain a feature representation that is better suited for classification. Due to the novelty of the field of activity recognition, there are many open questions with respect to these pre-processing steps:

[1] The material in this chapter is largely drawn from [63, 68, 71].

- **With what granularity should the sensor data be discretized?** The length of the timeslices used during discretization has important consequences for the recognition performance of a model. Too short timeslices will include too many irrelevant details that can hurt the performance of the model, while too large timeslices can smooth out important details of the signal.
- **What feature representation should we use for the sensor data?** A good feature representation reduces the variability of the data within a particular class (in our case an activity) and makes the data easily distinguishable among classes. The choice of feature representation can therefore have a great impact on the recognition performance of a model.

After preprocessing the data, it can be applied to the probabilistic model. We distinguish between a learning phase and an inference phase. During learning, the model parameters are estimated from a collection of training data. By performing probabilistic inference, we are able to determine the most likely sequence of activities the model recognizes from a novel sequence of sensor data. In terms of modeling, we answer the following questions:

- **Which model is best suited for activity recognition?** The choice of the model is complicated and depends on issues of recognition performance, computation time and amount of data needed to learn the model parameters. CRFs have outperformed HMMs in other domains, will the same hold for activity recognition?
- **How much training data is needed to accurately learn the model parameters?** Probabilistic models rely of labeled data to learn the model parameters. Collecting labeled data is expensive and therefore it is interesting to see how the amount of available training data relates to the recognition performance.

In this chapter, we first discuss related work with respect to HMMs and CRFs in Section 3.2. Section 3.3 introduces the notation used. Section 3.4 introduces the HMM and its learning and inference methods, while Section 3.5 does the same for CRFs. In Section 3.6, we highlight the similarities and differences between these models. Section 3.7 describes our experiments and presents our results and in Section 3.8, we discuss these results. Finally, in Section 3.9, we sum up the conclusions of this chapter.

3.2 Related Work

The first application of the HMM to a challenging task such as speech recognition was implemented in the 1970s. A paper by Lawrence Rabiner [130], provides a clear overview and tutorial on the theory of HMMs and their application

to speech recognition. Since then, the HMM has shown to be an enormously versatile model with applications in handwriting recognition [128], computer vision [42] and computational biology [76].

In 2001, CRFs were introduced and quickly grew in popularity as they were shown to outperform HMMs [78]. CRFs directly model the conditional probability distribution used during inference, which makes them more robust to violations of the modeling assumptions [153]. We explain the reason for this robustness in detail in Section 3.6.2. A consequence of modeling the conditional distribution directly is that there is often no closed-form solution for estimating the model parameters from data. Instead the parameters are estimated iteratively using a numerical method by maximizing the likelihood. Calculating the likelihood involves calculating the conditional probability for all timeslices, making it a relatively expensive operation. A lot of work has focused on finding efficient learning algorithms for CRFs. Various first-order derivative numerical optimization techniques were proposed, but eventually second-order derivative methods were used to find the optimal parameters [142, 169].

Hidden Markov models are part of a family of models called generative models, while conditional random fields are part of the discriminative family of models. Although discriminative models usually outperform generative models when large amounts of training data are available, generative models have been shown to outperform discriminative models when training data is limited [117]. Generative and discriminative models can be combined into a hybrid model in which a subset of the model parameters is learned using the generative approach and another subset of parameters is learned using the discriminative approach. Depending on the amount of training data available, hybrid models can outperform both generative and discriminative model [133].

The application of probabilistic models to activity recognition from wireless sensor networks started with the naive Bayes model [158]. The naive Bayes classifier is a generative probabilistic model that does not model temporal correlations, instead, it models each timeslice as a separate data point. Patterson et al. applied the HMM to activity recognition from RFID data and showed that modeling temporal correlations results in a strong performance gain [124]. Wilson et al. modeled activity recognition in a home with multiple inhabitants. Their approach is similar to Patterson's in that the HMM was used to model temporal correlations, however, a particle filter was used to assign sensor readings to each inhabitant in the home, after which the activity recognition was performed for each individual separately [173].

Conditional random fields were applied to gesture recognition [109, 171] and role recognition in a game environment [167]. These tasks are similar to activity recognition in that they are defined by a combination of simple actions and that the start and end point of gestures and roles are not known beforehand. CRFs have been applied to recognize activities such as going to work and visiting a

friend from GPS data in an outside setting [86]. Hu et al. applied CRFs to activity recognition in a home setting using wireless sensor networks. They compared the performance of CRFs to the naive Bayes classifier, showing CRFs outperform naive Bayes significantly [53]. However, because the naive Bayes classifier does not model temporal correlations and the CRF does, it is unclear whether the gain in performance is caused merely by these differences in modeling assumptions.

Hybrid models were applied to recognize activities from wearable sensor data. Experimental results showed a slight increase in performance for the hybrid model compared to the discriminative model [84].

In these works, the activity recognition is performed using various kinds of sensors. In terms of wireless sensor networks, none of these works provides any experimental comparison with respect to the timeslice length for discretization and the feature representation for classification. Although the HMM and CRF have both been applied to activity recognition, these models have not been compared on a single dataset. In this chapter, we compare the recognition performance of these models on three real world datasets.

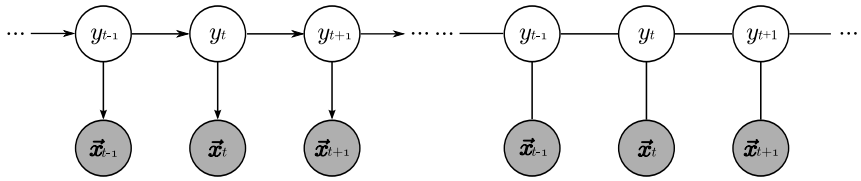
3.3 Notation and Discretization

The data obtained from the sensors is discretized into T timeslices of length Δt. A single feature value is denoted as x_t^i, indicating the value of feature i at timeslice t, with $x_t^i \in \{0, 1\}$. Feature values can either represent the raw values obtained directly from the sensor, or can be transformed according to a fixed function. In Section 3.7.3 we present various feature representations that can be used.

In a house with N installed sensors, we define a binary observation vector $\vec{x}_t = (x_t^1, x_t^2, \ldots, x_t^N)^T$. The activity at timeslice t, which is the state that the system is in, is denoted with $y_t \in \{1, \ldots, Q\}$ for Q possible states. Our task is to find a mapping between a sequence of observations $\mathbf{x}_{1:T} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T\}$ and a sequence of labels $\mathbf{y}_{1:T} = \{y_1, y_2, \ldots, y_T\}$ for a total of T timeslices.

3.4 Hidden Markov Model

Our goal is to recognize which activities took place given a sequence of sensor data. That is, we wish to find the sequence of activities $\mathbf{y}_{1:T}$ that best explains the sequence of observations $\mathbf{x}_{1:T}$. In temporal probabilistic models, this problem corresponds to finding the sequence $\mathbf{y}_{1:T}$ that maximizes the probability $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T})$.



(a) HMM (b) CRF

Fig. 3.1: The graphical representation of first-order HMM and a linear-chain CRF. The shaded nodes represent observable variables, while the white nodes represent hidden ones.

In this section, we present the HMM and describe how we can use this model to find the sequence of activities that best explains the sequence of observations. We start with a model definition in which we explain the probability distributions that make up the model and introduce the set of model parameters underlying these distributions. Then we explain how the model is used to find the sequence $\mathbf{y}_{1:T}$ that maximizes $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T})$. Finally, we explain how the model parameters are learned from a collection of training data.

3.4.1 Model Definition

The HMM is a generative probabilistic model which specifies the joint probability of the activities and sensor data $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T} \mid \theta)$, called the likelihood function. This is a function of the model parameters θ, which is optimized during learning.

There are two dependence assumptions that define this model, represented with the directed arrows in Figure 3.1(a).

- The hidden variable at time t, namely y_t, depends *only* on the previous hidden variable y_{t-1} (first order Markov assumption [130]).
- The observable variable at time t, namely $\vec{x}_t$, depends *only* on the hidden variable y_t at that time slice.

The joint probability distribution therefore factorizes as follows

$$p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) = p(y_1) \prod_{t=1}^{T} p(\vec{x}_t \mid y_t) \prod_{t=2}^{T} p(y_t \mid y_{t-1}). \tag{3.1}$$

Each of the factors represent a probability distribution:

- Initial state distribution $p(y_1)$, represents the probability of starting in state y_1 and is a multinomial distribution with parameter values denoted as $p(y_1 = i) \equiv \pi_i$.

- Observation distribution $p(\vec{x}_t \mid y_t)$, represents the probability that the state y_t would generate observation vector $\vec{x}_t$.
- Transition distribution $p(y_t \mid y_{t-1})$, represents the probability of going from one state to the next.

In order to model the distribution of the observation vector exactly, we would need to consider all possible combinations of values in the vector's dimensions. This would require 2^N parameters per activity, where N is the number of features. Which easily results in a large number of parameters, even for small N, and requires accordingly large numbers of training elements. Therefore, we apply the naive Bayes assumption, which means that we model each feature independently of the other features. Applying this assumption requires only N parameters for each activity. This is a very strong model assumption which most likely does not represent the true distribution of the data (i.e. it is very likely that two sensors are dependent on each other with respect to a particular activity). However, naive Bayes has been shown to give very good results in many domains, even when the independence assumption is violated [136]. Each feature is modeled by an independent Bernoulli distribution, where μ_{ni} is the parameter of the nth feature for state i. So that $p(\vec{x}_t \mid y_t) = \prod_{n=1}^{N} p(x_n \mid y_t)$, with $p(x_n \mid y_t = i) = \mu_{ni}^{x_n}(1 - \mu_{ni})^{(1-x_n)}$.

The transition distribution is modeled by Q multinomial distributions, one for each activity, where individual transition probabilities are denoted as $a_{ij} \equiv p(y_t = j \mid y_{t-1} = i)$. The HMM is therefore fully specified by the parameters $A = \left\{a_{ij}\right\}$, $B = \{\mu_{ni}\}$ and $\pi = \{\pi_i\}$.

3.4.2 Inference

The inference problem for the HMM consists of finding the single best state sequence that maximizes $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})$. Although the number of possible paths grows exponentially with the length of the sequence, the best state sequence can be found efficiently using the Viterbi algorithm. Using dynamic programming, we can discard a number of paths at each time step, resulting in a computational complexity of $O(TQ^2)$ for the entire sequence, where T is the total number of timeslices and Q the number of states [130]. This allows us to solve the segmentation problem in linear time. Further details on the Viterbi algorithm can be found in Appendix A.1.

3.4.3 Parameter Learning

The model parameters are learned by finding the maximum likelihood parameters. Given a collection of training data $\mathbf{x}_{1:T}, \mathbf{y}_{1:T}$, we want to find those para-

meters that maximize $p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T} \mid \theta)$. This is equivalent to finding the maximum likelihood parameters of each of the factors that make up this joint likelihood.

We assume that our training data is always fully labeled and therefore we can optimize the parameters of our distributions in closed form. The initial state distribution is a multinomial distribution whose parameters are calculated by

$$\pi_i = \delta(y_1, i) \tag{3.2}$$

where $\delta(i, j)$ is the Kronecker delta function, giving 1 if $i = j$ and 0 otherwise.

The observation probability $p(x_t^n \mid y = i)$ is a Bernoulli distribution whose maximum likelihood parameter estimation is given by

$$\mu_{ni} = \frac{\sum_{t=1}^{T} x_t^n \delta(y_t, i)}{\sum_{t=1}^{T} \delta(y_t, i)} \tag{3.3}$$

where T is the total number of data points.

The transition probability $p(y_t = j \mid y_{t-1} = i)$ is a multinomial distribution whose parameters are calculated by

$$a_{ij} = \frac{\sum_{t=2}^{T} \delta(y_t, j)\delta(y_{t-1}, i)}{\sum_{t=2}^{T} \delta(y_{t-1}, i)} \tag{3.4}$$

where T is equal to the number of time slices.

We apply Laplace smoothing to the estimation of all parameters, this ensures that no parameter value will be 0 [40].

3.5 Conditional Random Fields

We have seen how in the HMM, a generative model, parameters are learned by maximizing the joint likelihood $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T} \mid \theta)$. Conditional random fields are discriminative models, in which we learn the model parameters by optimizing the conditional likelihood $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}, \theta)$, rather than the joint likelihood. Conditional random fields represent a general class of discriminative models. A CRF using the first-order Markov assumption is called a linear-chain CRF and most closely resembles the HMM in terms of structure. In this section we give the definition of the linear-chain CRF and describe its inference and learning algorithms.

3.5.1 Model Definition

We define our linear-chain CRF as a discriminative analog of our previously defined HMM (see Fig. 3.1(b)). This means that the same dependence assumptions hold, that is:

- The hidden variable at time t, namely y_t, depends *only* on the previous hidden variable y_{t-1} (first order Markov assumption [130]).
- The observable variable at time t, namely $\vec{x}_t$, depends *only* on the hidden variable y_t at that time slice.

These assumptions are represented in the model using feature functions, so that the conditional distribution is defined as

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

where K is the number of feature functions used to parameterize the distribution, λ_k is a weight parameter and $f_k(y_t, y_{t-1}, \vec{x}_t)$ a feature function. The product of the parameters and the feature function $\lambda_k f_k(y_t, y_{t-1}, \vec{x}_t)$ is called an energy function, and the exponential representation of that term is called a potential function [9]. Unlike the factors in the joint distribution of HMMs, the potential functions do not have a specific probabilistic interpretation and can take any positive real value.

The partition function $Z(\mathbf{x}_{1:T})$ is a normalization term that ensures that the distribution sums up to one and obtains a probabilistic interpretation [153]. It is calculated by summing over all possible state sequences

$$Z(\mathbf{x}_{1:T}) = \sum_{\mathbf{y}} \left\{ \prod_{t=1}^{T} \exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \vec{x}_t) \right\}. \tag{3.5}$$

The feature functions $f_k(y_t, y_{t-1}, \vec{x}_t)$ for the CRF can be grouped as observation feature functions and transition feature functions. In defining the feature functions, we use a multi-dimensional index to simplify the notation, rather than the one-dimensional index used above. This gives the following feature function definitions:

Observation: $f_{inv}(x_t^n, y_t) = \delta(y_t, i) \cdot \delta(x_t^n, v)$

Transition: $f_{ij}(y_t, y_{t-1}) = \delta(y_t, i) \cdot \delta(y_{t-1}, j)$

where v is the value of the feature.

In Section 3.6, we will see that the differences in formulation between CRF and HMM are largely notational differences. The real difference lies in how the parameters are optimized.

3.5.2 Inference

Inference in a CRF is done using the Viterbi algorithm, similarly as with HMMs. The algorithm has a computational complexity of $O(TQ^2)$ [153]. A generalized

version of the Viterbi algorithm that works for both HMMs and CRFs can be found in Appendix A.1.

3.5.3 Parameter Learning

The parameters $\theta = \{\lambda_1, \ldots, \lambda_K\}$ of CRFs are learned by maximizing the conditional log likelihood $l(\theta) = \log p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}, \theta)$ given by

$$l(\theta) = \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \vec{x}_t) - \log Z(\mathbf{x}_{1:T}) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2}$$

where the final term is a regularization term, penalizing large values of λ to prevent overfitting. The constant σ is set beforehand and determines the strength of the penalization [153].

The function $l(\theta)$ is concave, which follows from the convexity of $\log Z(\mathbf{x}_{1:T})$ [153]. A useful property of convex functions in parameter learning is that any local optimum is also a global optimum. Quasi-Newton methods such as BFGS have been shown to be suitable for CRFs [142, 169]. These methods approximate the Hessian, the matrix of second derivatives, by analyzing successive gradient vectors. Because the size of the Hessian is quadratic in the number of parameters, storing the full Hessian is memory-intensive. We therefore use a limited-memory version of BFGS [14, 87]. The partial derivative of $l(\theta)$ with respect to λ_i, is given by

$$\frac{\partial l}{\partial \lambda_i} = -\frac{\lambda_i}{\sigma^2} + \sum_{t=1}^{T} f_i(y_t, y_{t-1}, \vec{x}_t) - \sum_{t=1}^{T} \sum_{y_t, y_{t-1}} p(y_t, y_{t-1} \mid \vec{x}_t) f_i(y_t, y_{t-1}, \vec{x}_t)$$

3.6 Generative vs. Discriminative Models

Generative models such as the HMM are defined by a factorization of the joint distribution $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})$. Discriminative models such as CRFs are defined by the conditional distribution $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T})$. We can rewrite the conditional distribution in terms of the joint distribution as

$$p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}) = \frac{p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})}{\sum_{\mathbf{y}} p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})}. \tag{3.6}$$

In this section, we first show that we can formulate the conditional distribution of CRFs in terms of the joint distribution of the HMM using the formula above. This will show how closely related these models actually are. We then show the

differences between these models due to their different optimization criteria used during parameter estimation. That is, parameters of the HMM are learned by maximizing the joint distribution, while the parameters of CRFs are learned by maximizing the conditional distribution.

3.6.1 From HMMs to CRFs

In Section 3.4.1, we introduced the factorized joint distribution of the HMM. Here, we rewrite the joint distribution using a more generalized notation

$$p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) = C \exp\left\{ \sum_t \sum_{i,j \in S} \lambda_{ij} \delta(y_t, i) \delta(y_{t-1}, j) + \sum_t \sum_{i \in S} \sum_{n \in N} \sum_{v \in V} \mu_{inv} \delta(y_t, i) \delta(x_t^n, v) \right\} \tag{3.7}$$

in which C is a normalization constant and $\delta(a, b)$ is a Kronecker delta function, giving 1 if $a = b$ and 0 otherwise. Every HMM can be written in this form by setting $\lambda_{ij} = \log p(y_t = i \mid y_{t-1} = j)$ and $\mu_{inv} = \log p(x_t^n = v \mid y_t = i)$, in which we have defined $p(y_1 \mid y_0) = p(y_1)$ for the sake of notational simplicity.

The factors $p(y_t = i \mid y_{t-1} = j)$ and $p(x_t^n \mid y_t)$ of the HMM are probabilities and therefore constrained to values between 0 and 1. However, in the energy function framework of CRFs the exponents of these terms are potential functions and can take any positive real value. The parameters λ_{ij} and μ_{inv} can therefore take any real value (positive or negative). This greater flexibility requires us to add a normalization constant C to ensure the joint distribution is still guaranteed to sum to 1. Despite this greater flexibility, it can be shown that Formula 3.7 describes exactly the class of HMMs as formulated by 3.1 [153].

To simplify the notation, we replace the Kronecker delta functions by feature functions $f_{ij}(y_t, y_{t-1}, \vec{x}_t) = \delta(y_t, i)\delta(y_{t-1}, j)$ and $f_{inv}(y_t, y_{t-1}, \vec{x}_t) = \delta(y_t, i)\delta(x_t^n, v)$. By using a one-dimensional index for all feature functions and their corresponding parameters, we obtain the joint probability in the following form

$$p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) = C \prod_{t=1}^{T} \exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \tag{3.8}$$

where C is the normalization term and K is the total number of parameters.

By applying Formula 3.6, we can calculate the conditional distribution.

$$p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}) = \frac{p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})}{\sum_{\mathbf{y}} p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})} \tag{3.9}$$

$$= \frac{C \prod_{t=1}^{T} \exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)}{\sum_{\mathbf{y}} C \prod_{t=1}^{T} \exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)} \tag{3.10}$$

$$= \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t), \tag{3.11}$$

where $Z(\mathbf{x}_{1:T})$ is the observation specific normalization function

$$Z(\mathbf{x}_{1:T}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t). \tag{3.12}$$

This gives us the same function for CRFs as we defined in Section 3.5.1. Besides the difference in notation, it differs from the formula of the HMM for two reasons, 1) we calculate the conditional distribution instead of the joint distribution and 2) we use potential functions instead of probabilities.

3.6.2 Differences in learning parameters

We have seen how closely related the HMM and CRF are. When doing inference for a novel sequence of observations both models maximize the conditional distribution $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T})$ to find the optimal sequence of states that fit to the observations. The true difference between HMM and CRF therefore lies in their optimization criteria used during parameter estimation. That is, parameters of the HMM are learned by maximizing the joint distribution, while the parameters of CRFs are learned by maximizing the conditional distribution.

The difference between these two learning method becomes clear when the models are based on incorrect modeling assumptions [117]. Models often have incorrect modeling assumptions because either we do not know the true underlying distribution or because the choice of distribution results in fewer parameter values, therefore requiring less data to learn the model parameters accurately. For example, while defining our HMM, we explained how the use of the naive Bayes assumptions is most likely wrong, but is sensible to prevent an exponential growth in parameters.

Example

To illustrate the impact of incorrect modeling assumptions further, consider an example involving classes A and B. The data points x for each class are indepen-

dently and identically distributed (i.i.d.) and both classes y are equally likely to occur ($p(y = A) = p(y = B) = 0.5$). We assign a novel data point to one of the classes by calculating the posterior according to Bayes rule

$$p(y \mid x) = \frac{p(x \mid y)p(y)}{p(x)}. \tag{3.13}$$

The data of class A ($p(x \mid y = A)$) is distributed as a Gaussian distribution with mean 3 and standard deviation 1. Data of class B ($p(x \mid y = B)$) is distributed as a Gaussian with mean 5 and standard deviation 2 (Fig. 3.2(a)).

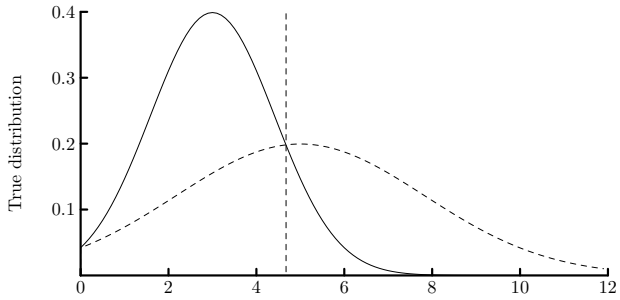
We model the data using a generative and a discriminative model, both using a Gaussian distribution with a fixed standard deviation of 1 to model the observation distribution $p(x \mid y)$. Because the standard deviation is fixed, the means of the Gaussians are the only parameters to learn. In the case of the generative model we learn the means by maximizing the joint probability $p(x, y)$. This is equivalent to maximizing the observation distribution $p(x \mid y)$ independently. The maximum likelihood parameters are learned by calculating the mean of the data points available for training. Figure 3.2(b) shows the resulting Gaussian distribution learned from data. We see that a generative model manages to learn the correct mean of 3 and 5 for the two distributions, but because of the incorrect modeling assumption (standard deviation is fixed at 1) a classification error arises.

In the case of the discriminative model, the parameters are learned by maximizing the posterior probability $p(y \mid x)$ directly. No closed form solution is available for maximizing this function and therefore a numerical method such as gradient ascent can be used. Figure 3.2(c) shows the resulting Gaussian distribution learned from data. We see that the model learned an incorrect mean for class B (mean of 6.35), however, this incorrect mean does result in optimal classification.

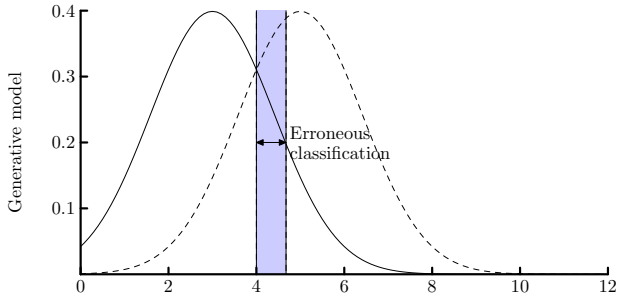
This example shows that discriminative models are more robust in dealing with violations of the modeling assumptions.

3.7 Experiments

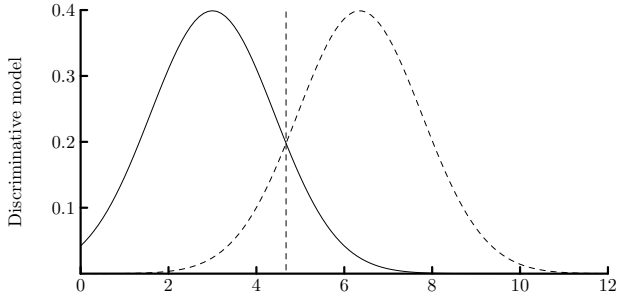
In this section, we present the experiments and their results for the comparison of the HMM and the CRF. We describe the datasets used, the experimental setup, a number of feature representations and the goal and results of the experiments. A total of three experiments are done. The first experiment is aimed at finding the ideal timeslice length for discretizing the sensor data. In the second experiment, we compare the recognition performance of the feature representations and models presented in this chapter. Finally, in the third experiment, we determine how much sensor data is needed to accurately learn the model parameters.



(a) True distributions.



(b) Distributions obtained using generative model.



(c) Distributions obtained using discriminative model.

Fig. 3.2: Example of incorrect model assumptions in which discriminative models learn parameters that correctly classify the data, while generative models learn parameters that result in a classification error.

House A	House B	House C
Other	Other	Other
Leaving	Leaving	Leaving
Toileting	Toileting	Eating
Showering	Showering	Toileting
Brush teeth	Brush teeth	Showering
Sleeping	Sleeping	Brush teeth
Breakfast	Dressing	Shaving
Dinner	Prep. Breakfast	Sleeping
Snack	Prep. Dinner	Dressing
Drink	Drink	Medication
	Dishes	Breakfast
	Eat Dinner	Lunch
	Eat Breakfast	Dinner
	Play piano	Snack
		Drink
		Relax

Tab. 3.1: List of activities for each home.

3.7.1 Datasets used

Our experiments are done using the datasets we recorded (introduced in Section 2.4.3). There are three datasets corresponding to three different homes. A list of activities that were annotated for each dataset can be found in Table 3.1. In these experiments, we consider each dataset separately, and use all annotated activities of each dataset for evaluation.

3.7.2 Experimental Setup

We split our data into a test and training set using a 'leave one day out' approach. In this approach, one full day of sensor readings is used for testing and the remaining days are used for training. We cycle over all the days and report the average performance measure.

We evaluate the performance of our models using precision, recall and F-measure. These measures can be calculated using the confusion matrix shown in Table 3.2. The rows show the ground truth labels as provided by a human annotator, while the columns show the labels inferred by the model. The diagonal of the matrix contains the true positives (TP), while the sum of a row gives us the total of ground truth labels (TT) and the sum of a column gives us the total of inferred labels (TI). We calculate the precision and recall for each class separately and then take the average over all classes.

Ground truth \ Inferred	1	2	3	
1	TP_1	ϵ_{12}	ϵ_{13}	TT_1
2	ϵ_{21}	TP_2	ϵ_{23}	TT_2
3	ϵ_{31}	ϵ_{32}	TP_3	TT_3
	TI_1	TI_2	TI_3	$Total$

Tab. 3.2: Confusion matrix showing the true positives (TP), total of ground truth labels (TT) and total of inferred labels (TI) for each class.

It is important to use these particular measures because we are dealing with unbalanced datasets. In unbalanced datasets, some classes appear much more frequent than other classes. Our measure takes the average precision and recall over all classes and therefore considers the correct classification of each class equally important. To further illustrate the importance of these measures, we also include the accuracy in our results. The accuracy represents the percentage of correctly classified timeslices, therefore more frequently occurring classes have a larger weight in this measure.

$$\text{Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TI_i} \tag{3.14}$$

$$\text{Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TT_i} \tag{3.15}$$

$$\text{F-Measure} = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{3.16}$$

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} TP_i}{Total} \tag{3.17}$$

The significance testing between two cases A and B is done at a confidence interval of 95% using a one-tail student t-test and using matching paired days. This means that performance in case A for a particular day of the cross validation is compared to the performance in case B for that exact same day of the cross validation. We do this because there are large differences in the activities performed throughout the various days. It can be said that some days are more challenging in terms of recognition than others. However, a single day will be equally challenging for both models. Therefore, to allow a fair comparison, it is important to match the performance of individual days while calculating the significance.

(a) Raw (b) Changepoint (c) Last-fired

Fig. 3.3: Different feature representations.

3.7.3 Feature Representation

The raw data obtained from the sensors can either be used directly, or be preprocessed into a different representation form. We experiment with three different feature representations:

Raw: The raw sensor representation uses the sensor data directly as it was received from the sensors. It gives a 1 when the sensor is firing and a 0 otherwise (Fig. 3.3(a)).

Changepoint: The change point representation indicates when a sensor event takes place. That is, it indicates when a sensor changes value. More formally, it gives a 1 when a sensor changes state (i.e. goes from zero to one or vice versa) and a 0 otherwise (Fig. 3.3(b)).

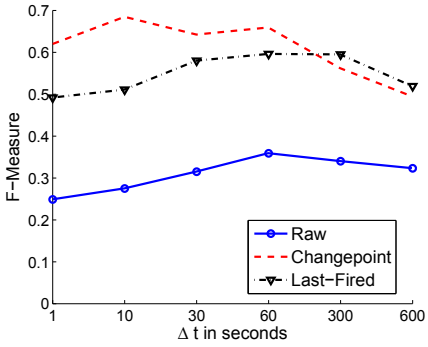
Last-fired: The last-fired sensor representation indicates which sensor fired last. The sensor that changed state last continues to give 1 and changes to 0 when another sensor changes state (Fig. 3.3(c)).

3.7.4 Experiment 1: Timeslice Length

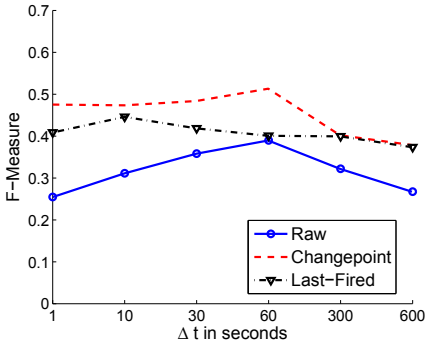
Here, we present our findings for determining the ideal timeslice length for discretizing the sensor data. Experiments were run using only the HMM, because for very small timeslice lengths (e.g. 1 second) the CRF would take several days to find the optimal parameters. We experimented using all the feature representations, to rule out any bias towards any of the representations.

During discretization both the sensor data and the ground truth activity labels are discretized using the same timeslice length. Especially when using large timeslice lengths, it is possible that one activity ends somewhere halfway during a timeslice, and another activity starts. This means multiple activities appear in the same timeslice. In this case, we use the activity that takes up most of the timeslice.

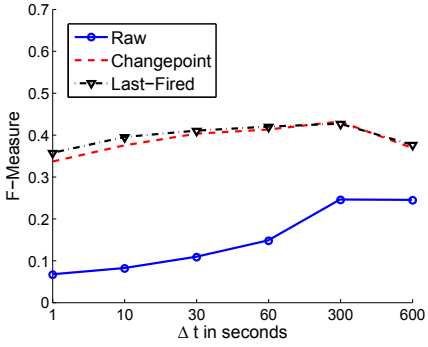
The discretized ground truth labels are used to learn the model parameters.



(a) House A



(b) House B



(c) House C

Fig. 3.4: F-Measure performance of the HMM for the three houses using different timeslice length to discretize the data.

Length		House A	House B	House C
$\Delta t =$	1 **s**	0.0	0.0	0.0
$\Delta t =$	10 **s**	0.2	0.2	0.2
$\Delta t =$	30 **s**	0.6	0.6	0.9
$\Delta t =$	60 **s**	1.3	1.1	1.7
$\Delta t =$	300 **s**	5.9	4.0	8.1
$\Delta t =$	600 **s**	10.6	17.4	13.7

Tab. 3.3: Discretization Error percentages.

Using this discretized ground truth for calculating the performance measures of the model would result in a bias towards large timeslices. In larger timeslices, the shorter activities would not survive the discretization process, thus making the classification task easier. Instead, we evaluate the performance using the original ground truth (not discretized) which was obtained with a one second accuracy.

The F-measure values for the various timeslice lengths are plotted in Figure 3.4. We see that for each house no timeslice length achieves consistently the best performance for all feature representations. Furthermore, we see that across all three homes, no timeslice length significantly outperforms the other timeslice lengths for a particular feature representation. Although the difference in performance is not significant, we see that the timeslice lengths of $\Delta t = 30$ seconds and $\Delta t = 60$ seconds give an overall good performance.

Table 3.3 shows the discretization error for each timeslice length. This error shows the percentage of discretized ground truth timeslices that are incorrectly labeled compared to the original non-discretized ground truth. This shows how accurately the discretized ground truth represents the actual ground truth. We see that the discretization error for timeslice lengths of $\Delta t = 300$ seconds and $\Delta t = 600$ is rather large, while for smaller timeslice lengths the error remains around 1% or lower.

3.7.5 Experiment 2: Feature Representation and Model

This experiment shows the performance of the HMM and the CRF for the different feature representations. Data was discretized using the timeslice length of $\Delta t = 60$ seconds. The discretized ground truth was used to calculate the performance measures. The results for House A are shown in Table 3.4, House B in Table 3.5 and House C in Table 3.6. Feature representations were used standalone and combined. Combining the feature representations was done by concatenating the feature matrices.

We first compare the results in terms of feature representations using only the F-measure. We see that out of the standalone representation, the raw representa-

		Precision	Recall	F-Measure	Accuracy
HMM	**Raw**	38 ± 20	46 ± 20	41 ± 20	59 ± 29
	Change	70 ± 16	74 ± 13	72 ± 14	92 ± 6
	Last	55 ± 17	70 ± 13	61 ± 15	90 ± 8
	Raw&Change&Last	64 ± 17	78 ± 11	70 ± 14	94 ± 4
	Raw&Change	47 ± 20	56 ± 20	51 ± 20	61 ± 29
	Raw&Last	63 ± 16	77 ± 12	69 ± 13	94 ± 4
	Change&Last	67 ± 18	79 ± 12	72 ± 15	94 ± 4
CRF	**Raw**	59 ± 19	56 ± 17	57 ± 17	90 ± 8
	Change	74 ± 17	68 ± 16	70 ± 16	91 ± 6
	Last	66 ± 16	66 ± 14	66 ± 15	96 ± 2
	Raw&Change&Last	72 ± 16	74 ± 13	73 ± 14	97 ± 3
	Raw&Change	75 ± 16	72 ± 13	73 ± 14	94 ± 5
	Raw&Last	67 ± 15	68 ± 14	67 ± 14	96 ± 3
	Change&Last	72 ± 15	74 ± 13	73 ± 14	97 ± 2

Tab. 3.4: Experiment 2, House A: Different feature representations for HMMs and CRFs.

		Precision	Recall	F-Measure	Accuracy
HMM	**Raw**	36 ± 12	45 ± 13	39 ± 13	63 ± 25
	Change	45 ± 17	60 ± 15	51 ± 16	80 ± 14
	Last	36 ± 16	45 ± 20	40 ± 17	48 ± 26
	Raw&Change&Last	37 ± 8	49 ± 16	42 ± 10	80 ± 13
	Raw&Change	26 ± 11	32 ± 11	28 ± 10	43 ± 25
	Raw&Last	33 ± 11	42 ± 14	37 ± 12	74 ± 22
	Change&Last	40 ± 9	53 ± 15	44 ± 9	80 ± 15
CRF	**Raw**	36 ± 17	41 ± 13	38 ± 15	78 ± 26
	Change	47 ± 10	50 ± 10	49 ± 9	92 ± 7
	Last	47 ± 12	48 ± 11	47 ± 11	88 ± 15
	Raw&Change&Last	32 ± 18	38 ± 17	34 ± 17	77 ± 20
	Raw&Change	32 ± 20	32 ± 19	32 ± 19	62 ± 24
	Raw&Last	29 ± 19	33 ± 18	30 ± 19	73 ± 19
	Change&Last	38 ± 17	40 ± 16	39 ± 17	85 ± 12

Tab. 3.5: Experiment 2, House B: Different feature representations for HMMs and CRFs.

		Precision	Recall	F-Measure	Accuracy
HMM	**Raw**	13 ± 8	17 ± 8	15 ± 8	26 ± 22
	Change	41 ± 8	50 ± 12	45 ± 8	76 ± 17
	Last	42 ± 11	53 ± 16	46 ± 12	80 ± 15
	Raw&Change&Last	42 ± 12	54 ± 13	47 ± 12	70 ± 24
	Raw&Change	42 ± 10	51 ± 12	46 ± 10	73 ± 22
	Raw&Last	42 ± 12	53 ± 11	46 ± 11	71 ± 27
	Change&Last	36 ± 14	47 ± 19	40 ± 16	48 ± 26
CRF	**Raw**	12 ± 11	16 ± 10	13 ± 10	38 ± 21
	Change	35 ± 20	37 ± 19	36 ± 19	78 ± 18
	Last	34 ± 17	37 ± 17	35 ± 17	84 ± 12
	Raw&Change&Last	44 ± 14	48 ± 11	46 ± 12	82 ± 19
	Raw&Change	41 ± 15	43 ± 13	42 ± 14	80 ± 24
	Raw&Last	42 ± 16	44 ± 13	43 ± 14	77 ± 22
	Change&Last	50 ± 11	52 ± 12	51 ± 11	89 ± 14

Tab. 3.6: Experiment 2, House C: Different feature representations for HMMs and CRFs.

tion gave very poor F-measure performance in all homes and with both models. The change and the last representations both perform significantly better than the raw representation in all houses and for both models. The changepoint representation performs on average better than the last representation, but this is not a significant difference. When looking at the combinations of feature representations, we see that the 'raw&change&last' and the 'change&last' representations generally give good performance and in some cases on average perform better than the standalone change and last representations. The same holds for the raw representation combined with either the change or the last representation, however, these differences in performance are not significant.

Next, we compare the results of the models. Comparing the HMM and CRF in terms of F-measure gives a very mixed result. For some representations, the HMM is better, for others the CRF. In Houses A and B, the changepoint representation gives the best or close to the best performance for both models. For House C, the same holds for the HMM, but for the CRF the 'change&last' representation works best in that house. In some cases, there are significant differences in performance between the HMM and the CRF for a particular feature representation. For example, the CRF using the 'raw&change' representation applied to House A performs significantly better than the HMM using the same representation. Such differences occur both in favor of the HMM as well as in favor of the CRF.

When comparing the model performances in terms of precision and recall, we see that the CRF on average performs better than the HMM in terms of precision, while the HMM on average performs better than the CRF in terms of recall.

True \ Model	Other	Leaving	Toileting	Showering	Brush teeth	Sleeping	Breakfast	Dinner	Snack	Drink
Other	915	309	517	401	36	196	61	861	91	820
Leaving	30	19282	12	7	6	0	0	0	0	0
Toileting	46	4	259	13	15	19	0	2	1	6
Showering	7	1	13	229	0	0	0	1	0	0
Brush teeth	5	3	12	3	7	0	0	2	0	0
Sleeping	3	0	44	0	4	10778	0	0	0	0
Breakfast	11	0	3	0	0	1	31	22	9	10
Dinner	13	0	0	0	0	0	12	225	27	10
Snack	6	0	0	1	0	1	2	12	20	0
Drink	5	0	1	1	0	0	1	10	2	29

Tab. 3.7: Experiment 2, House A: Confusion matrix for the HMM using the last-fired features.

True \ Model	Other	Leaving	Toileting	Showering	Brush teeth	Sleeping	Breakfast	Dinner	Snack	Drink
Other	3586	271	16	55	0	178	0	94	0	7
Leaving	8	19319	8	2	0	0	0	0	0	0
Toileting	59	10	220	9	0	63	0	4	0	0
Showering	182	6	6	57	0	0	0	0	0	0
Brush teeth	10	3	17	2	0	0	0	0	0	0
Sleeping	0	0	27	0	0	10802	0	0	0	0
Breakfast	23	0	0	0	0	3	53	0	3	5
Dinner	110	3	2	0	0	0	6	161	3	2
Snack	15	3	0	0	0	0	15	3	6	0
Drink	19	2	3	0	0	0	3	2	0	20

Tab. 3.8: Experiment 2, House A: Confusion Matrix for CRF using last-fired features.

Finally, when comparing the models in terms of accuracy, we see that the CRF performs on average better than the HMM in almost all cases, and does so significantly in a number of them, for example, in the case of the raw and 'raw&change' representation in House A.

To further understand the difference in performance between the two models, we compare the confusion matrices. The confusion matrix for House A, using the last-fired representation in combination with the HMM is shown in Table 3.7 and in combination with the CRF in Table 3.8. We see that the activities that take up most timeslices are generally recognized better by the CRF, while the less frequent activities are recognized better by the HMM. Interestingly enough, the tooth brushing activities was not inferred once by the CRF. The HMM recognized 7 instances of the tooth brushing correctly, but misclassified it many times. Overall, in both models, we see that most of the confusion occurs among the activities breakfast, dinner, snack and drink. These activities are all performed in the same room, namely the kitchen. Similarly, we see that there is confusion among the activities toileting, showering and brush teeth which are all performed in the bathroom.

		House A	House B	House C
Learning	HMM	1.3s	0.6s	1.0s
	CRF	1890.1s	1188.3s	3708.8s
Inference	HMM	3.9s	2.4s	3.2s
	CRF	4.7s	3.5s	5.2s

Tab. 3.9: Experiment 2: Computation times in seconds for learning and inference in HMMs and CRFs.

Table 3.9 shows the computation times for performing learning and inference for each of the models when using the changepoint representation. We see that the time to learn the model parameters in CRF is a lot higher than learning in HMMs. In the case of House C, the amount of learning time for the CRF exceeds one hour, while the parameters of the HMM are learned in one second. This is because the parameters of the HMM can be learned in closed form, while for the CRF, we have to use numerical optimization methods. Since learning is generally performed offline, this does not have to limit the practical use of the CRF.

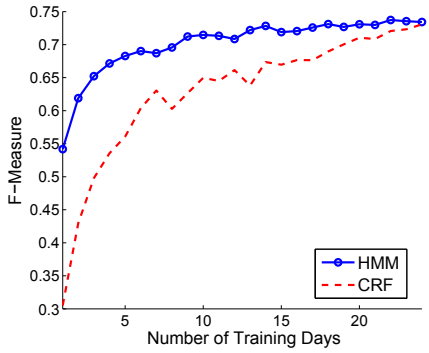
3.7.6 Experiment 3: Amount of Training Data

In this experiment we measure the effect of the amount of training data used to learn the model parameters. Different training days provide different amounts of information for each activity, therefore the days used for training were randomly sampled from the days available for training. We performed cross validation over all the days available, by using a single day for testing and sampling the number of training days needed from the remaining days. For each size, 10 samples of training days were collected and the model performance was averaged over these samples, and over all test days used for cross validation. Figure 3.5 shows the F-Measure for both models and for all three homes. We see that the HMM on average performs better than the CRF, but that the difference in performance becomes smaller as more training data is available.

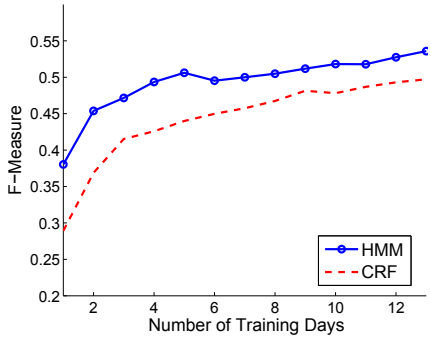
3.8 Discussion

Timeslice Length

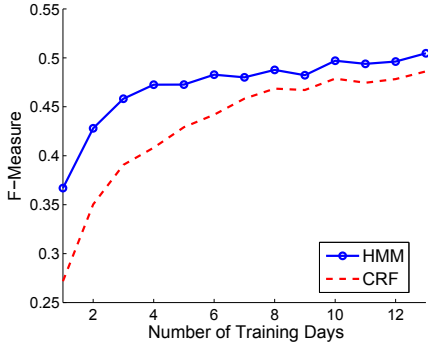
Our results from Experiment 1 showed that the recognition performance is not very strongly affected by the length of the time interval used for discretization. Timeslice lengths of $\Delta t = 30$ seconds and $\Delta t = 60$ seconds generally give a good



(a) House A



(b) House B



(c) House C

Fig. 3.5: Experiment 3: F-Measure performance for the HMM and CRF using various sizes of training data.

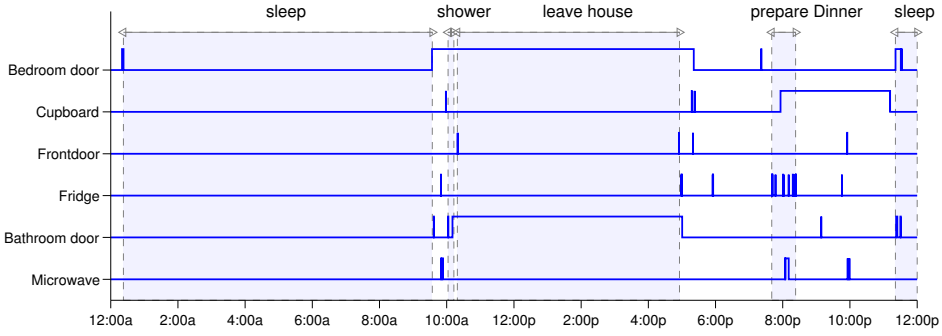
performance and result in a small discretization error. Besides the performance and error, the memory usage and computation time also play an important role in discretization. Smaller timeslice lengths result in higher memory usage and computation time. In Experiment 2, we saw that learning the parameters of the CRF, using a discretization of $\Delta t = 60$ seconds, can take up more than one hour. Using a smaller timeslice length would linearly increase this computation time. Taken all these factors into account a timeslice length of $\Delta t = 60$ seconds is best suitable for our experiments and we will use that discretization interval for the remainder of this thesis.

Feature Representations

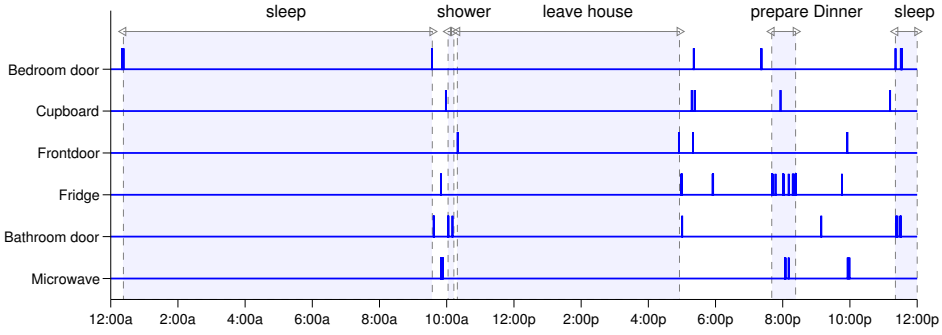
Our results showed large differences in performance between the different feature representations. The reason for these differences is best understood by comparing the representations using an example. Figure 3.6 shows each feature representation for a selection of sensors of a single day of sensor data. Each plot also shows at which time which activity was being performed. In the raw feature representation plot (Fig. 3.6(a)), we see that the bedroom door and bathroom door sensors sometimes continue to fire even though no relevant activity is taking place. This is because people do not always carefully close doors behind them. From a pattern recognition point of view this severely reduces the discriminative value of that sensor. If the bedroom door sensor fires half of the day without any sleeping activity taking place, the significance of that sensor is greatly reduced for recognizing the sleeping activity. A raw sensor representation can, however, provide important information with respect to the classification. For example, people tend to be very strict in closing their doors when performing toileting and bathing activities. If a toilet flushes and the toilet door is closed, this is a strong indication that the toileting activity is taking place. On the other hand, if the toilet flushes and the toilet door is open, it is more likely that somebody is, say, emptying a bucket of water used for cleaning the house.

In the example of the changepoint representation (Fig. 3.6(b)), we see that the issue of continuous sensor firing is resolved. We see that certain sensors consistently coincide with the start and end point of an activity. For the bedroom door sensor, this is the case with respect to the sleep activity, and for the front door sensor, this is the case with the leave house activity. We also see that after these sensor events, there are long periods in which no sensor fires at all. Also in areas where no activity is being performed, we see long periods in which there is no sensor activity. This makes the observation of no sensors firing at all a very ambiguous one in this representation.

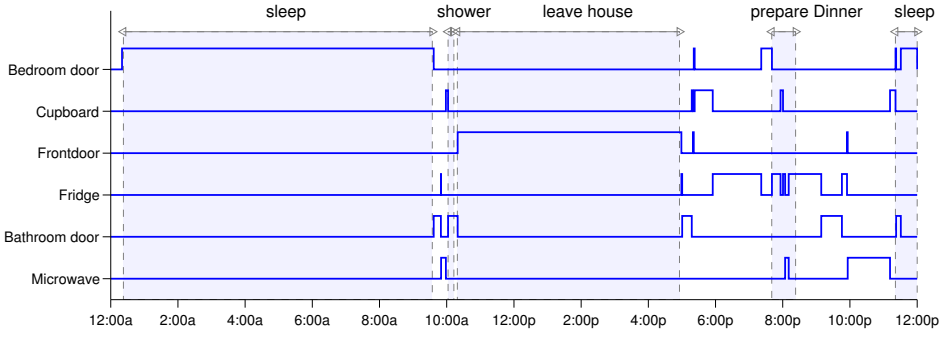
This ambiguity is resolved in the last-fired sensor representation (Fig. 3.6(c)). The motivation for this representation is that the last sensor that fired is very indicative of the location of the person in the home. A person that remains in a



(a) Raw sensor representation



(b) Changepoint sensor representation



(c) Last-fired sensor representation

Fig. 3.6: Visualization of sensor data using the (a) raw, (b) changepoint and (c) last-fired sensor representation. The shaded area indicates an activity was performed, the label of the activity is shown at the top.

particular room of a house is only able to trigger sensors within that room. As soon as the person leaves that room, it is likely that he or she quickly triggers one of the other sensors outside that room. This representation works best when a large number of sensors are installed so that there is a high chance that a sensor fires when a person resides in a room, and there is a high chance that a sensor fires outside the room once the person leaves the room.

The outcome of the experiments show that the changepoint and the last representations perform significantly better than the raw representation. Combinations of representations can sometimes result in a slight increase in performance, but never significantly outperforms the changepoint and the last representations.

Generative and Discriminative Models

The difference in performance between the generative HMM and the discriminative CRF is mainly visible in terms of precision and recall. This is caused by the way each model learns its model parameters. The parameters of the HMM are learned for each activity separately, while the parameters of the CRF are learned by optimizing the sequence of training data as a whole. The optimization method of the CRF therefore favors the correct classification of more frequent classes over the correct classification of infrequent classes when this would result in misclassification of the frequent class. This also explain that the activity of tooth brushing was not recognized at all when using the CRF. A further consequence of this optimization method is that CRFs generally score better on the accuracy measure. The accuracy represents the percentage of correctly classified timeslices. Because CRFs weigh the recognition of frequent classes higher than infrequent classes, they are more likely to recognize a high number of timeslices. We introduced the F-measure as an evaluation measure because we value the recognition of each activity equally important. The reason the F-measures have rather high standard deviations is because there are large differences between the days used in cross validation. For example, on a weekday a person spends a lot of time out of the house, while during the weekend many different activities are performed in the house. This makes weekends more challenging in terms of recognition. Although, this difference will also be noticeable in the accuracy measure, it will be less severe because there will still be many timeslices in which the person is idle or performing another activity.

Our results show that the ability of the CRF to deal with violations of the modeling assumptions does not result in a significant increase in the F-measure performance. Rather, the model shifts its recognition focus to the more frequent classes, at the cost of misclassifying the infrequent classes. Furthermore, we have seen that the parameter estimation for the CRF takes much longer than learning the parameters of the HMM.

For the application of health monitoring, the long learning time does not seem to be a very significant issue, since parameter estimation only has to be done once during the installation of an activity recognition system. However, ignoring infrequent classes for classification seems to be of larger concern. In practice, this could mean the classifier ignores the recognition of taking in medication to obtain a more accurate classification of the amount of time spent sleeping. On the other hand, the HMM misclassifies many instances of the frequent classes, to achieve a few correct classifications of the infrequent classes. It is not clear which information care givers value more and the F-measure performance has its limitations in that sense. Our results do not clearly show whether misclassification occurs because an activity is recognized to take up more time than it actually did (e.g. the subject slept for eight hours, but it is recognized as nine hours) or whether complete new instances of activities are recognized (e.g. making dinner was recognized while the subject was pacing idly through the kitchen). Future work should therefore focus on determining which information about activities is most informative to care givers and come up with a performance measure which matches the accuracy of this information.

Amount of Training Data

We have seen that the difference in recognition performance between the CRF and the HMM decreased as more training data was available. These results are similar to work by Ng and Jordan [117]. They compared the classification performance of generative naive Bayes model and the discriminative logistic regression model in various classification tasks. They found that generative models perform better when little training data is available, while discriminative models do better when training data is plenty.

3.9 Conclusions

In this chapter, we explored techniques for preprocessing sensor data, presented two probabilistic models for activity recognition and introduced four performance measures for evaluating the performance of these approaches. Preprocessing the sensor data consists of a discretization step and a transformation step. Sensor data was discretized using a timeslice of constant length and experiments were run to determine the effect of the timeslice length on the recognition performance. Three feature representations for transforming the sensor data were introduced, the raw representation, the changepoint representation and the last-fired sensor representation.

Two models for activity recognition were presented, the HMM and the CRF. The HMM is a generative model and the CRF a discriminative model. A comparison

between generative and discriminative models was provided to highlight the differences and similarities between these two types of models. The most important difference between the models lies in the way the model parameters are estimated. For HMMs, the model parameters are learned by maximizing the likelihood of all the distributions that make up the model, this can be done in closed form. In the case of CRFs, the model parameters are learned by maximizing the conditional likelihood, which is done using numerical optimization methods. As a result CRFs are more robust to violations of the modeling assumptions.

We presented four measures for evaluating the performance of our approaches. A multi-class precision and recall measure was presented and these two measures were combined in the F-measure. Additionally an accuracy measure was presented which represents the percentage of correctly classified timeslices.

Our experiments on three real world datasets showed that the recognition performance is not very strongly affected by the length of the time interval used for discretization. When taking constraints with respect to memory, computation time and discretization error into account, a time interval of 60 seconds provides the proper balance between an accurate representation of the data and a reasonable amount of data needed for this representation.

Experiments on the feature representations compared each feature representation separately. The results show that the changepoint and the last representations perform significantly better than the raw representation. Comparisons were also made between each possible combination of feature representations. Combinations of representations can sometimes result in a slight increase in performance, but never significantly outperforms the separate use of the change and last representations.

The comparison between the performance of the models showed that the ability of the CRF to deal with violations of the modeling assumptions does not result in a significant increase in F-measure performance compared to the HMM. This is because the CRF shifts its recognition focus to activities that occur more frequently in the dataset, at the cost of misclassifying infrequent activities. For this reason the CRF generally performs better than the HMM in terms of accuracy measure, since in this measure more frequent activities have a larger weight. On the other hand, the HMM misclassifies many timeslices of frequent activities to obtain a few correct classifications of infrequent activities. The results therefore show that the CRF generally performs better in terms of precision, while the HMM performs better in terms of recall. Experiments focusing on the effect of the amount of training data used show that the HMM on average performs better than the CRF when little training data is available, but that the difference in performance becomes smaller as more training data is available.

4

Semi-Markov models

4.1 Introduction

In the previous chapter, we used the hidden Markov model (HMM) and the conditional random field (CRF) to recognize activities. The experiments showed that most confusion of activities occurs between activities performed in the same room. Activities performed in one room typically involve largely the same sensors. This makes it hard to distinguish activities based purely on which sensors fired. Instead, the analysis of temporal characteristics, such as the duration of an activity, might provide important evidence for recognition. For example, shaving and brushing teeth both involve the use of the bathroom door and the faucet. If no other sensors are used, it is difficult to distinguish these activities based on the fact that the bathroom door and faucet are used. However, because shaving typically takes up more time than brushing teeth, the duration of the activity is likely to be very informative for recognition.

In the first-order Markov models, discussed in the previous chapter, state durations are modeled implicitly by means of self-transitions of states. The prior distribution of the duration in that case takes the form of a geometric distribution. Geometric distributions have a mode fixed at one timeslice and this might not always accurately represent the actual distribution of durations for an activity. For example, showering typically takes several minutes, to shower in one minute time is generally less likely than taking several minutes. Because duration modeling in Markov models follows directly from the self-transition of states, we cannot use a different distribution than the geometric distribution.

[1] The material in this chapter is largely drawn from [65, 67].

Using higher-order Markov models allows more flexible parameterization, but still restricts us to using the geometric distribution. Instead, we propose to use semi-Markov models in which we are free to choose the duration distribution.

In semi-Markov models, the duration of a state is modeled explicitly by a random variable. This explicit modeling allows us to use any distribution we wish for representing the duration of an activity. In this chapter, we compare the generative hidden semi-Markov model (HSMM) and the discriminative semi-Markov conditional random field (SMCRF) to their conventional Markov counterparts (see Table 4.1). We answer the following questions:

- **Which distribution should we use for modeling duration?** The freedom of using any distribution for modeling the duration of an activity raises the question which distribution is most suitable.

- **What is the effect of duration modeling in both generative and discriminative models?** We compare the performance of Markov models to semi-Markov models. Is there a difference in performance gain between the generative and discriminative models?

- **How does the feature representation affect the duration modeling?** Since duration modeling can help recognition when sensor data is ambiguous, it is interesting to see the effect of duration modeling for different feature representations.

The remainder of this chapter is organized as follows. In Section 4.2, we discuss related work. Section 4.3 describes the HSMM and its learning and inference algorithms. Section 4.4 describes SMCRFs and its learning and inference algorithms. In Section 4.5, we highlight the differences between these models and their conventional counterparts. In Section 4.6, we present the experiments and results using our real world data sets. Finally, in Section 4.8, we sum up our conclusions.

	Markov	**Semi-Markov**
Generative	HMM	HSMM
Discriminative	CRF	SMCRF

Tab. 4.1: Categorization of hidden Markov model (HMM), hidden semi-Markov model (HSMM), conditional random field (CRF) and semi-Markov conditional random field (SMCRF).

4.2 Related Work

The limited possibilities of modeling state durations in the HMM is considered as a major weakness of the model [130]. HSMMs were introduced to resolve that weakness and found their first application within speech recognition [85]. Various kinds of HSMMs exist, that mainly differ in the way the observation model is defined. In this chapter, we restrict ourselves to the most commonly used HSMM sometimes referred to as the explicit duration HMM. Murphy presented a unified review of the inference and learning algorithms for the different kinds of HSMMs by presenting them in the form of a dynamic Bayesian network [111]. The inference algorithm for HSMMs is formulated as an extension to the Viterbi algorithm for HMMs. Its computational complexity is a factor D more expensive than the original Viterbi algorithm, where D is the maximum duration considered. The limited modeling of state durations in HMMs also applies to CRFs and so SMCRFs were introduced and applied to information extraction tasks [139].

In work by Duong et al., HSMMs were applied to activity recognition [32]. They compared the performance of the model using various duration distributions and suggested the use of the Coxian distribution because of its computational efficiency. Truyen et al. applied a hierarchical version of SMCRFs to activity recognition [165]. The performance was compared to hierarchical CRF and a conventional CRF, the hierarchical SMCRF outperformed both. Both of these works used a small data set recorded in a laboratory setting to evaluate the model performance. No evaluation of semi-Markov models for activity recognition has been done on real world datasets. Furthermore, the performances of generative and discriminative semi-Markov models have never been compared on a single dataset. Such a comparison allows us to evaluate the advantage of discriminative modeling. By comparing the semi-Markov model performance with the performance of Markov models, we can determine the effect of accurate duration modeling in a real world setting.

4.3 Hidden Semi-Markov Model

The HSMM is a generative model that differs from the HMM because next to modeling the observations and transitions between hidden states, HSMMs model the duration of hidden states explicitly using a random variable.

In the case of the HMM duration is modeled implicitly by the self-transitions of states. Given an HMM in a known state, the prior probability that it stays in that state for l timeslices is

$$p_i(l) = (a_{ii})^{l-1}(1 - a_{ii}) \tag{4.1}$$

where $p_i(l)$ is the discrete probability density function (PDF) of duration l in state i and a_{ii} is the self-transition probability of state i [130]. We see that the function consists of $l-1$ self-transitions (i.e. the model remains in state i) and one transition to a different state than state i, as expressed by the term $1-a_{ii}$. This duration density function takes the form of a geometric distribution with a mode fixed at one timeslice. The self-transitions are part of the transition distribution of the HMM and therefore follow directly from the model definition. Duration modeling is therefore not explicitly defined for the HMM but follows as an inherent property of the model, therefore, we cannot use a different distribution to model the duration.

In the rest of this section, we give the model definition of HSMMs and briefly describe its inference and learning algorithms.

4.3.1 Model definition

To model the duration in an HSMM, we introduce a discrete random variable d_t, which represents the remaining duration of state y_t. When the Markov-chain enters the state, a value of d_t is sampled from the duration distribution. Then in the consecutive timeslices, the value of d_t is decreased by one at each timeslice, and as long as the value is larger than zero the model continues to stay in state y_t. When the value of d_t reaches zero a transition to a new state is made and the duration of the new state is sampled from the duration distribution.

Note, that d_t is defined as the *remaining* duration of a state, therefore the actual duration when generating a new duration is $d_t + 1$ timeslices. In order to make this distinction clear, we will use the length of a state l to refer to the actual duration and the remaining duration d_t, to refer to the count-down variable at time t.

The joint probability of the HSMM is factorized as

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

where we have used $p(y_1 \mid y_0, d_0) = p(y_1)$ and $p(d_1 \mid d_0, y_1) = p(d_1 \mid y_1)$ for the sake of notational simplicity. The observation model $p(\vec{x}_t \mid y_t)$ is the same as with the HMM. Transitions between states are modeled by the factor $p(y_t \mid y_{t-1}, d_{t-1})$, defined as:

$$p(y_t = i \mid y_{t-1} = j, d_{t-1}) = \begin{cases} \delta(i,j) & \text{if } d_{t-1} > 0 \text{ (remain in same state)} \\ a_{ij} & \text{if } d_{t-1} = 0 \text{ (transition)} \end{cases} \tag{4.2}$$

where $\delta(i,j)$ is the Kronecker delta function, giving 1 if $i = j$ and 0 otherwise. The parameter a_{ij} is part of a multinomial distribution. Some model definitions

of semi-Markov models force the value of a_{ii} to zero, effectively disabling self-transitions [111]. However, in this work, we do *not* force these values to zero and do allow self-transitions in the HSMM. The use of self-transitions in semi-Markov models allows convolutions of duration distributions and makes it possible that separate instances of activities immediately follow each other.

State durations are modeled by the term $p(d_t \mid d_{t-1}, y_t)$, defined as

$$p(d_t \mid d_{t-1}, y_t = k) = \begin{cases} p_k(d_t + 1) & \text{if } d_{t-1} = 0 \text{ (generate new duration)} \\ \delta(d_t, d_{t-1} - 1) & \text{if } d_{t-1} > 0 \text{ (count down } d_{t-1}) \end{cases} \tag{4.3}$$

where $p_k(d_t + 1)$ is the distribution used for modeling the state duration. In Section 4.6, we experiment with various distribution functions for modeling the duration.

4.3.2 Inference

In comparison with HMMs, the state space of HSMMs is larger and therefore inference requires more time. The Viterbi algorithm, used for inference in HMMs, iterates over each state at each timeslice and chooses the most likely state from which that state can be entered. Inference in HSMMs requires the same iterative procedure, but because the duration of a state is not known beforehand, the algorithm also needs to iterate over all possible durations at each timeslice. The complete procedure has a computational complexity of $O(TQ^2D)$, where D is the maximum duration an activity can have, T is the length of the sequence and Q is the number of states [111]. The value of D can be set differently for each activity. This can result in a great efficiency gain when different activities have largely different duration distributions. Sequences with durations larger than D automatically have a probability of 0. Further details on the Viterbi algorithm can be found in Appendix A.

4.3.3 Parameter Learning

We learn the model parameters by finding the maximum likelihood parameter values. Given some training data $\mathbf{x}_{1:T}, \mathbf{y}_{1:T}$, we want to find those parameters that maximize $p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T} \mid \theta)$. This is equivalent to finding the maximum likelihood parameter values of each of the factors that make up the joint probability.

Our training data is fully labeled with exact start and end time of each activity, so that we can optimize the parameters in closed form. The observation probability $p(x^n \mid y = i)$ is calculated similarly as with the HMM. Calculating the transition probability $p(y_t = i \mid y_{t-1} = j, d_{t-1} = l)$ requires counting the number of transitions in which the remaining duration d_{t-1} is 0. In all other cases, the duration

variable simply counts down. The parameters of the multinomial distribution are therefore calculated by

$$a_{ij} = \frac{\sum_{t=2}^{T} \delta(y_t, j)\delta(y_{t-1}, i)\delta(d_{t-1}, 0)}{\sum_{t=2}^{T} \delta(y_{t-1}, j)\delta(d_{t-1}, 0)} \tag{4.4}$$

where T is equal to the number of timeslices.

The parameter values of the duration distribution depend on which distribution is used. We compare a number of duration distributions in Section 4.6. The parameters of these distributions are learned by maximum likelihood estimation.

4.4 Semi-Markov Conditional Random Fields

Just like the HSMM is an HMM in which we model a duration variable, the SMCRF is a CRF in which duration is modeled explicitly. The conditional probability is $p(\mathbf{y}_{1:T}, \mathbf{d}_{1:T} \mid \mathbf{x}_{1:T})$, defined as

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

The feature functions $f_k(y_t, y_{t-1}, \vec{x}_t, d_t, d_{t-1})$ for the SMCRF can be grouped as observation feature functions, transition feature functions and duration feature functions. In defining the feature functions we use a multi-dimensional index to simplify notation, rather than the one-dimensional index used above. The observation feature function is defined similarly as with CRFs as $f_{vin}(x_t^n, y_t) = \delta(y_t, i) \cdot \delta(x_t^n, v)$. The transition feature function is defined as $f_{ij}(y_t, y_{t-1}, d_t) = \delta(y_t, i) \cdot \delta(y_{t-1}, j) \cdot \delta(d_t, 0)$. Defining the duration feature function depends on the type of distribution used for modeling the duration. We use feature functions of the form $g_i(y_t, d_t) = \delta(y_t, i) \cdot d_t^2$, $g_i'(y_t, d_t) = \delta(y_t, i) \cdot d_t$ and $g_i''(y_t, d_t) = \delta(y_t, i) \cdot 1$ which gives a contribution proportional to $(d_t - \lambda)^2$, for appropriate values of λ_i,λ_i' and λ_i''. Notice that each of the duration feature functions is assigned with its own parameter λ_i which is independent of the other parameters. The use of our duration feature functions therefore *allows*, but does not *enforce* a Gaussian distribution; the parameter learning will find the best distribution, which will only be Gaussian if that is what is present in the data.

4.4.1 Inference

Inference in SMCRFs is done using a Viterbi algorithm similar to that of HSMMs and has a computational complexity of $O(TQ^2D)$ [139]. See Appendix A for details about the algorithm.

4.4.2 Parameter Learning

Parameter estimation in SMCRFs is done similarly as with CRFs. The inclusion of duration simply adds another term in the equations. The parameters $\theta = \{\lambda_1, \ldots, \lambda_K\}$ of SMCRF are learned by maximizing the conditional log likelihood $l(\theta) = \log p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}, \theta)$ given by

$$l(\theta) = \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \vec{x}_t, d_t, d_{t-1}) - \log Z(\mathbf{x}_{1:T}) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2}$$

where the final term is a regularization term penalizing large values of λ to prevent overfitting. The constant σ is set beforehand and determines the strength of the penalization [153].

The partial derivative of $l(\theta)$ with respect to λ_i, is given by

$$\frac{\partial l}{\partial \lambda_i} = -\frac{\lambda_i}{\sigma^2} + \sum_{t=1}^{T} f_i(y_t, y_{t-1}, \vec{x}_t, d_t, d_{t-1}) - \sum_{t=1}^{T} \sum_{y_t, y_{t-1}, d_t, d_{t-1}} p(y_t, y_{t-1}, d_t, d_{t-1} \mid \vec{x}_t) f_i(y_t, y_{t-1}, \vec{x}_t, d_t, d_{t-1}).$$

4.5 Model Comparison

In the previous sections, we introduced a generative and a discriminative semi-Markov model. This section highlights the differences between these models and between their Markov counterparts. We discuss the effects of inaccurate duration modeling in both generative and discriminative models and explain the effects on computational complexity when using discriminative models.

4.5.1 Learning in generative models

In section 3.6.2, we illustrated the difference between generative and discriminative models by showing the effect of estimating the parameters of two Gaussian or normal distributions based on incorrect modeling assumptions. We explained how incorrect modeling assumptions are often used because either the family of the true underlying distribution is not known, or because the choice of distribution results in fewer parameter values, hence requiring less data to train the model. Here, we illustrate a similar effect in the case of duration modeling.

We provide an example involving activities A and B, which we wish to classify using only the duration of the activity. That is, the initial state distribution, the

observations and the transitions do not provide any information as to which activity is more likely. The durations of activities A and B are distributed as truncated normal distributions with unit variance and means $\mu_n^A = 3$ and $\mu_n^A = 7$, respectively (Fig. 4.1a).

When using semi-Markov models, we can simply use the normal distribution to accurately model the duration distributions. However, in the case of Markov models (e.g. HMM, CRF) the duration distribution is fixed to the geometric distribution, therefore resulting in an incorrect modeling assumption.

In the case of the HMM, the parameter values of the geometric distribution are set to the maximum likelihood estimates. We can calculate these parameter values by matching the first moment (the mean) of the normal distribution to the first moment of the geometric distribution. The geometric distribution takes the form $p_i(d) = (a_{ii})^{d-1}(1 - a_{ii})$, where d is the duration of a state and a_{ii} the self-transition probability of that state. The mean of the geometric distribution is calculated as follows:

$$\mu_g = \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) = \frac{1}{1 - a_{ii}}. \tag{4.5}$$

Using this formula, we can calculate the parameter values a_{ii} for the means of $\mu_g^A = 3$ and $\mu_g^B = 7$. This results in two geometric distributions which are shown in Figure 4.1b. Because the duration is the only source of information by which the activities can be classified, we can easily determine the outcome of the classification. The shaded area in the figure shows where the use of the geometric distribution results in misclassification. This illustrates how the modeling of duration with an incorrect PDF can lead to misclassification in the HMM.

In the case of CRFs, we are directly optimizing the conditional probability distribution. Since the duration distribution is our only informative source of information for classification, estimating its parameters values comes down to minimizing the classification error with respect to the real duration distribution. This can yield a solution as shown in Figure 4.1c and illustrates that also in the case of duration modeling, discriminative models are more robust towards violations of the modeling assumptions.

4.5.2 Computational complexity

An important consequence of using discriminative models is the increase in computational complexity during learning. In generative models, the parameters of the distributions used can usually be estimated using a closed form solution. Discriminative models, on the other hand, typically require numerical methods because no closed form solution is available. This is especially costly because

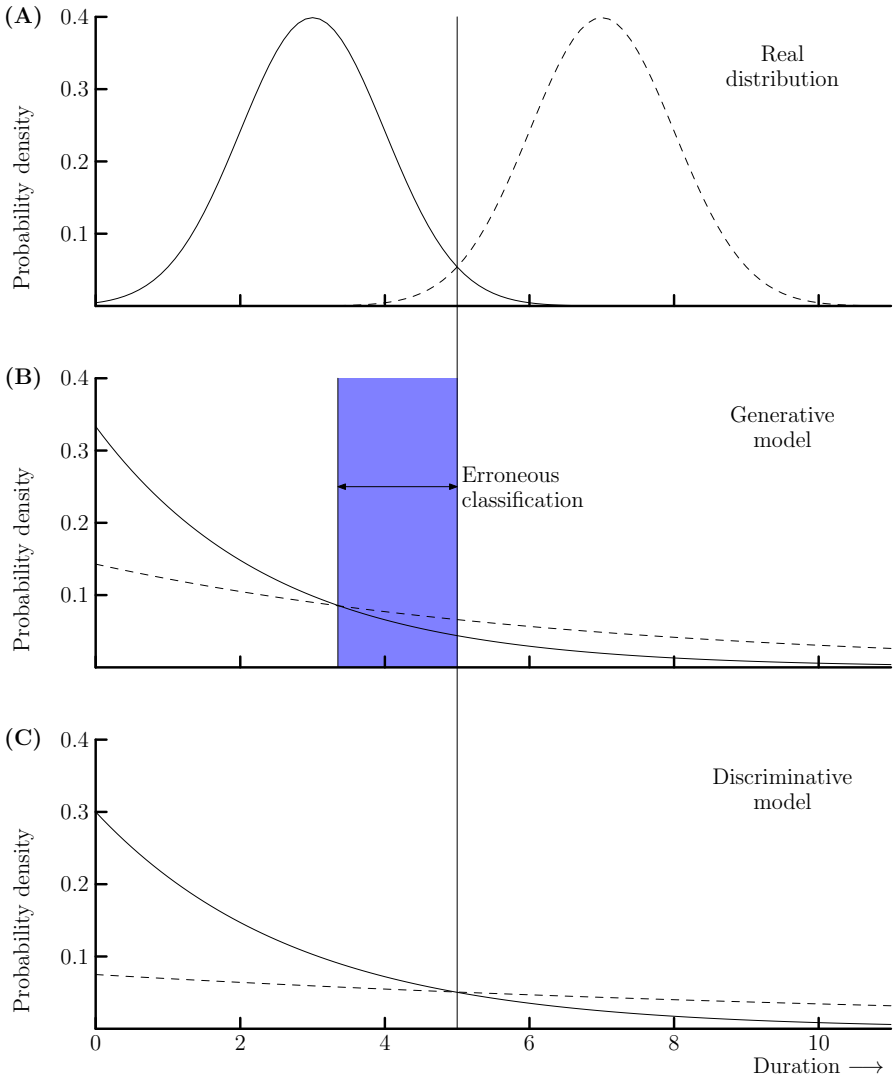


Fig. 4.1: Plots of (a) Gaussian distribution for means 3 (straight line) and 7 (dashed line). (b) Geometric distribution with a set of parameters learned using maximum likelihood estimation, typically used in generative models. The shaded area shows where the incorrect use of the geometric distribution leads to misclassification. (c) Geometric distribution with a set of parameters learned using discriminative models.

during each iteration of the learning phase, we need to perform inference to calculate the normalization term.

The use of semi-Markov models introduces an additional computational complexity. Because the durations of activities are not known for a novel sequence of observations, all possible durations need to be considered at each timestep. This makes the computational complexity of doing inference in semi-Markov models a factor D higher than in conventional models, where D is the maximum duration of an activity. This affects both HSMMs and SMCRFs, but when fully labeled data is available, the HSMM does not need to perform inference during training, because a closed form solution for parameter estimation is possible. Since the inference step is performed at each iteration during learning in the SMCRF, finding the model parameters for this model is very expensive.

In the experiments section, we report the amount of time needed for inference and learning in each of the models to illustrate these differences.

4.6 Experiments

In this section, we present the experiments and their results for classification based on duration modeling. We describe the datasets used, the experimental setup and the goal and results of the experiments. A total of three experiments are done. The first experiment is aimed at finding the ideal distribution for modeling the durations of activities. In the second experiment, we compare the recognition performance of the Markov and semi-Markov models. Finally, in the third experiment, we investigate the effect of using different feature representations on the recognition performance.

4.6.1 Datasets used

Due to the high computational complexity of the SMCRF model, we use subsets of the datasets collected in the three houses. In Chapter 3, we saw that most recognition confusion occurs between activities taking place in the same room using the same set of sensors. Therefore, we created subsets of the original datasets by selecting all the activities and sensors that are performed and present in one room. We did this for the kitchen and for the bathroom, resulting in six datasets (two for each house) which are challenging for recognition and a good testset for evaluating the effect of accurate duration modeling. Tables 4.2 and 4.3 show the list of sensors used for the bathroom and kitchen datasets, respectively. Tables 4.4 and 4.5 show the activities of these datasets.

Sensors		
BathroomA	BathroomB	BathroomC
Bathroom door	Toilet flush	Bathroom door
Toilet door	Bathroom door	Toilet flush
Bedroom door	Bathroom PIR	Bathtub PIR
	Kitchen PIR	Dresser PIR
	Sink	
	Chair	

Tab. 4.2: List of sensors used in each of the bathroom datasets. PIR is short for 'passive infrared', the sink and toilet flush use a float sensor, the chair uses a pressure mat and the remaining sensors use reed switches to measure the open-close state.

Sensors		
KitchenA	KitchenB	KitchenC
Microwave	Microwave	Microwave
Refrigerator	Refrigerator	Refrigerator
Freezer	Frontdoor	Freezer
Cupboard with plates	Cupboard with plates	Cupboard with plates
Cupboard with cups	Cupboard with groceries	Cupboard with cups
Cupboard with pans	Stove lid	Cupboard with pans
Cupboard with groceries	Toaster	Cupboard with boxes
Dishwasher	Cutlary drawer	Cutlary drawer
	Sink	
	Kitchen PIR	

Tab. 4.3: List of sensors used in each of the kitchen datasets. PIR is short for 'passive infrared', the sink uses a float sensor, the stove lid and drawers use a mercury sensor and the remaining sensors use reed switches to measure the open-close state.

BathroomA			BathroomB			BathroomC		
Activity	Num.	Perc.	Activity	Num.	Perc.	Activity	Num.	Perc.
Brush teeth	16	0.1	Brush teeth	13	0.3	Brush teeth	26	0.4
Showering	23	1.1	Showering	11	1.1	Showering	10	0.2
Toileting	114	1.3	Toileting	27	0.6	Bathing	4	0.8
Other	-	97.5	Other	-	98.0	Shaving	7	0.3
						Other	-	98.3

Tab. 4.4: The activities that were annotated in the bathroom datasets. The 'Num.' column shows the number of times the activity occurs in the dataset. 'Perc.' indicates the percentage of timeslices that the activity occurs. All unannotated timeslices were collected in a single 'Other' activity. The 'Num.' column is not filled out for the 'Other' activity because there is no annotation available for that activity and it is therefore not clear whether a single sequence of 'Other' activity should be counted as one or as multiple occurrences of that activity.

KitchenA			KitchenB			KitchenC		
Activity	Num.	Perc.	Activity	Num.	Perc.	Activity	Num.	Perc.
Breakfast	20	0.4	prep. Breakfast	9	0.8	Breakfast	18	0.7
Dinner	9	1.5	prep. Dinner	6	0.8	Dinner	11	1.9
Snack	12	0.2	Wash Dishes	6	0.3	Snack	9	0.1
Drink	20	0.2	Drink	8	0.1	Drink	10	0.2
Other	-	97.7	Eat Breakfast	10	1.3	Other	-	97.1
			Eat Dinner	5	0.5			
			Other	-	96.2			

Tab. 4.5: The activities that were annotated in the kitchen datasets. The 'Num.' column shows the number of times the activity occurs in the dataset. 'Perc.' indicates the percentage of timeslices that the activity occurs. All unannotated timeslices were collected in a single 'Other' activity. The 'Num.' column is not filled out for the 'Other' activity because there is no annotation available for that activity and it is therefore not clear whether a single sequence of 'Other' activity should be counted as one or as multiple occurrences of that activity.

4.6.2 Experimental Setup

Data obtained from the sensors is discretized in timeslices of length $\Delta t = 60$ seconds. We split our data into a test and training set using a 'leave one day out' approach. In this approach, one full day of sensor readings is used for testing and the remaining days are used for training. A day of sensor data starts at 8 am and ends at 12 pm. This is done to reduce the computation time, which is very high in the case of the SMCRF. The hours were chosen as such, because few activities, besides the 'other' activity, take place outside those hours. We cycle over all the days and report the average performance measure.

In Section 3.7.2, we introduced the precision, recall and F-measure and showed that they are reliable measures for evaluating the performance of our model. We do not include the accuracy measure in reporting the results. In the previous chapter, we showed that the accuracy is not a useful measure for us, because it weights performances relative to the frequency of an activity.

The maximum duration D used by the semi-Markov models is determined by taking the maximum duration in the training set and adding 25% to account for outliers. For the 'other' activity the maximum duration was set to 1, therefore duration modeling for that activity is equivalent to using the geometric distribution as is done with the HMM. The 'other' activity can consist of very long sequences, which would result in a very high value for D. In either case, it is not likely that the distribution of durations for the 'other' activity can be easily parameterized because it is a collection of various activities.

Significance testing between two cases A and B is done at a confidence interval of 95% using a one-tail student t-test and using matching paired days. This means that the performance in case A for a particular day of the cross validation is com-

pared to the performance in case *B* for that exact same day of the cross validation. We do this because there are large differences in the activities performed throughout the various days. It can be said that some days are more challenging in terms of recognition than others. However, a single day will be equally challenging for both models. Therefore, to allow a fair comparison, it is important to match the performance of individual days while calculating the significance.

4.6.3 Experiment 1: Duration Distribution

In this experiment, we wish to determine which family of distributions is best suited for modeling the duration of activities. We experiment with three unimodal distributions, the gamma, Poisson and Gaussian distributions and we experiment with the multivariate and histogram distributions. In the case of the histogram distribution, we are free to choose the number of bins for representing the distribution. We experiment with 3, 5 and 10 bins, the multivariate distribution is equivalent to a histogram distribution with *D* bins, where *D* is the maximum duration of an activity. The F-measure values for the different datasets and distributions can be found in Table 4.6.

We see that the Poisson distribution performs worst in three of the six datasets, while the Gaussian distribution performs best in four of the six datasets. Overall the differences in performance among the various distributions are small. Using a histogram approach works well on all datasets except for the house C kitchen dataset. Depending on the number of bins used in the histogram the results vary, but no single number of bins consistently gives the best performance.

4.6.4 Experiment 2: Model

In this experiment, we used the 'changepoint & last' sensor representation. The Gaussian distribution is used for modeling duration in the HSMM and the feature functions described in Section 4.4 are used for the SMCRF. These feature functions allow the duration to be modeled as a Gaussian distribution. The results for these experiments using both Markov and semi-Markov models on the kitchen and bathroom datasets can be found in Table 4.7.

These results show that the HSMM significantly outperforms the HMM in terms of F-measure on all the datasets, except the bathroom C dataset. In the case of the bathroom C dataset, the HSMM on average performs better than the HMM, but this increase is not significant. This is an important result, because it shows that modeling duration accurately when using generative models in this experimental setup can result in a significant gain in performance.

The results further show that the SMCRF significantly outperforms the CRF in

	House A Bathroom	House A Kitchen
Gamma	78 ± 16	67 ± 24
Gauss	78 ± 16	69 ± 24
Poisson	60 ± 17	62 ± 30
Multivariate	77 ± 17	65 ± 24
Histogram 3 bins	77 ± 17	68 ± 24
Histogram 5 bins	77 ± 16	67 ± 25
Histogram 10 bins	77 ± 16	65 ± 24
	House B Bathroom	**House B Kitchen**
Gamma	76 ± 15	51 ± 19
Gauss	76 ± 15	54 ± 19
Poisson	73 ± 13	53 ± 21
Multivariate	79 ± 16	52 ± 20
Histogram 3 bins	75 ± 13	53 ± 20
Histogram 5 bins	76 ± 15	53 ± 19
Histogram 10 bins	77 ± 15	54 ± 19
	House C Bathroom	**House C Kitchen**
Gamma	65 ± 24	54 ± 21
Gauss	62 ± 26	56 ± 22
Poisson	59 ± 28	49 ± 16
Multivariate	62 ± 25	46 ± 20
Histogram 3 bins	58 ± 26	48 ± 21
Histogram 5 bins	65 ± 23	46 ± 19
Histogram 10 bins	64 ± 25	46 ± 20

Tab. 4.6: F-measure values for various duration distributions for all the datasets.

terms of F-measure on the bathroom A and kitchen B datasets. In the case of the kitchen A dataset, the CRF performs on average slightly better than the SMCRF. While in the case of the bathroom B and bathroom C datasets the SMCRF performs on average better, but this increase is not significant. These results show that an accurate duration modeling in this experimental setup can still result in a significant gain in performance, but does so less consistently as in the generative case.

When comparing the F-measure performance of the HSMM to that of the SMCRF, we see that on the kitchen A dataset, the SMCRF on average performs better than the HSMM, but this increase is not significant. On all five other datasets, the F-measure performance of the two models is either equal or nearly equal. This is an interesting result because our previous comparison revealed that accurate duration modeling in discriminative models does not result in a significant increase in performance as often as it does in generative models. Nonetheless discriminative semi-Markov models manage to perform at least equally well as

	Bathroom A			Kitchen A		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
HMM	50 ± 13	67 ± 15	57 ± 12	56 ± 30	64 ± 33	59 ± 31
HSMM	70 ± 17	85 ± 14	76 ± 15	65 ± 27	75 ± 21	69 ± 24
CRF	73 ± 17	74 ± 14	73 ± 15	80 ± 21	79 ± 20	79 ± 20
SMCRF	75 ± 17	75 ± 14	75 ± 15	77 ± 23	77 ± 19	76 ± 21
	Bathroom B			**Kitchen B**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
HMM	64 ± 18	85 ± 14	72 ± 15	47 ± 23	56 ± 23	50 ± 22
HSMM	67 ± 20	91 ± 13	76 ± 15	47 ± 22	67 ± 20	54 ± 19
CRF	72 ± 16	75 ± 16	73 ± 15	42 ± 24	46 ± 22	44 ± 23
SMCRF	75 ± 17	77 ± 18	76 ± 17	52 ± 33	56 ± 29	54 ± 31
	Bathroom C			**Kitchen C**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
HMM	48 ± 32	57 ± 32	52 ± 32	46 ± 21	49 ± 22	46 ± 19
HSMM	60 ± 27	69 ± 27	64 ± 26	54 ± 23	61 ± 24	56 ± 22
CRF	53 ± 27	62 ± 22	57 ± 25	55 ± 28	57 ± 24	55 ± 25
SMCRF	60 ± 27	65 ± 24	62 ± 26	53 ± 26	57 ± 24	55 ± 25

Tab. 4.7: Experiment 2: Precision, recall and F-measure for hidden Markov model (HMM), hidden semi-Markov model (HSMM), conditional random field (CRF) and semi-Markov conditional random field (SMCRF). Experiments were performed on the kitchen and bathroom datasets. The changepoint and last sensor representation was used.

generative semi-Markov models.

Finally, we compare the F-measure performance of the CRF to that of the HMM and find that the CRF significantly outperforms the HMM in terms of F-measure on the bathroom A and kitchen C datasets. On the kitchen A, bathroom B and bathroom C datasets the CRF performs on average better than the HMM, but this increase in not significant. The HMM performs on average better than the CRF on the kitchen B dataset, but not significantly. One possible explanation for these results is that the CRF is able to model durations better, due to its ability to deal with violations of the modeling assumptions.

Table 4.8 shows the computation times for learning and inference in all models on the bathroom A dataset. We see that learning in discriminative models takes significantly longer than in generative models. Learning the parameters of the SMCRF also takes much longer than learning the parameters of the CRF. Finally, the table shows that inference for semi-Markov models takes longer than for Markov models.

Model	HMM	HSMM	CRF	SMCRF
Learning Time	0.2 s	0.3 s	190.8 s	9882.2 s
Inference Time	1.1 s	1.9 s	1.3 s	1.8 s

Tab. 4.8: Computation time in seconds for learning and inference on the bathroomA dataset.

4.6.5 Experiment 3: Feature Representation

This experiment is similar as experiment 2, however, this time the 'changepoint' sensor representation is used. The results can be found in Table 4.9.

In terms of F-measure, the HSMM performs on average better than the HMM on all datasets except for the bathroom A and kitchen A datasets, where they perform equal. Only in the case of the kitchen C dataset, the increase in performance is significant. We see that with this sensor representation accurate duration modeling still helps in generative models, but does not result in such a significant gain in performance as with the previous experiment. When comparing the F-measure performance of the HMM and the HSMM of this experiment to the F-measure performances of the previous experiment, we see that the change in feature representation has resulted in an increase in performance on all datasets.

The SMCRF does not significantly outperform the CRF on any of the datasets, but on average performs better than the CRF for all datasets except for the bathroom A dataset (where it performs equal) and the kitchen C dataset (where it performs slightly worse). When comparing the results for the CRF and the SMCRF of this experiment to the F-measure performances of the previous experiment, we see that most results are either equal or close to equal. The change in feature representation seems to have less effect on the discriminative models.

4.7 Discussion

Duration distribution

Our first experiment was aimed at answering which distribution is best suited for modeling the duration of activities. We experimented with three unimodal distributions, a multivariate distribution and a histogram approach for various numbers of bins. In case the duration distribution of activities is consistently not unimodal, the histogram and multivariate distributions would probably have performed better than the unimodal distributions. However, the results showed little difference in the performance of the various distributions. The Gaussian distribution performed best in four of the six datasets and is therefore the distri-

	Bathroom A			Kitchen A		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
HMM	80 ± 17	84 ± 14	81 ± 15	66 ± 27	75 ± 20	70 ± 24
HSMM	79 ± 18	83 ± 16	81 ± 16	66 ± 26	76 ± 20	70 ± 23
CRF	72 ± 21	72 ± 18	72 ± 19	78 ± 23	76 ± 21	76 ± 22
SMCRF	73 ± 21	73 ± 18	72 ± 19	79 ± 24	80 ± 19	79 ± 22
	Bathroom B			**Kitchen B**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
HMM	68 ± 11	88 ± 14	76 ± 10	46 ± 22	64 ± 25	52 ± 22
HSMM	70 ± 15	91 ± 14	78 ± 13	50 ± 20	69 ± 22	57 ± 20
CRF	72 ± 16	74 ± 16	73 ± 15	41 ± 24	44 ± 22	43 ± 23
SMCRF	77 ± 15	78 ± 17	77 ± 15	46 ± 33	49 ± 27	47 ± 30
	Bathroom C			**Kitchen C**		
	Precision	**Recall**	**F-Measure**	**Precision**	**Recall**	**F-Measure**
HMM	57 ± 27	73 ± 23	63 ± 25	50 ± 23	59 ± 20	53 ± 20
HSMM	58 ± 27	75 ± 22	64 ± 24	61 ± 27	67 ± 22	62 ± 23
CRF	55 ± 28	60 ± 26	57 ± 27	61 ± 28	58 ± 23	59 ± 24
SMCRF	61 ± 28	64 ± 26	62 ± 27	61 ± 30	58 ± 23	58 ± 25

Tab. 4.9: Experiment 3: Precision, recall and F-measure for hidden Markov model (HMM), hidden semi-Markov model (HSMM), conditional random field (CRF) and semi-Markov conditional random field (SMCRF). Experiments were performed on the kitchen and bathroom datasets. The changepoint sensor representation was used.

bution that was used in the other experiments. An advantage of the Gaussian distribution is that it has convenient mathematical properties. In the case of generative models, the parameters of a Gaussian can be calculated in closed form and when using discriminative models, we can define a set of feature functions that allows the duration to be modeled as a Gaussian distribution.

In our current experimental setup, the same distribution was used for all activities. It is possible that the duration of different activities is distributed differently. For example, one activity might have a unimodal distribution, while another might have a bimodal distribution. In future work, we could experiment with using different distributions for different activities. Another interesting extension on this work is the inclusion of the Coxian distribution for modeling the duration of activities. This distribution was used in related work to model the duration of activities because of its computational efficiency and has been shown to give a good performance in activity recognition [32].

Feature Representation

The results of experiment 3 showed that the HMM performs much better when using the changepoint representation than when using the changepoint & last representation that was used in experiment 2. This is somewhat surprising because the same comparison was made in the previous chapter in which the HMM performs equally well for both feature representations and even performs better in one case when using the changepoint & last representation. The explanation lies in the use of subset datasets. In this chapter, we have created several subsets of the datasets used in the previous chapter. These subsets contain considerably less sensors than the original dataset. As explained in the previous chapter, the last representation works best when a large number of sensors is used, so that there is a high chance that another sensor is triggered once a person starts moving inside the house. Because all the sensors in our subsets are positioned together in a single room, the last representation completely loses its effectiveness as soon as the subject leaves that room. In fact, the last representation pollutes the observation space because at all times one of the sensors will continue to fire as 'other' activities are performed outside the room. Although this is clearly not what we would like to use for activity recognition, it does make some properties of duration modeling and of discriminative models clear.

Duration Modeling

From the results of experiment 2, we see that the performance of the HMM is clearly affected by the inclusion of the last representation. The HSMM on the other hand is hardly affected by the use of the last representation. We see that the use of accurate duration modeling in generative models manages to overcome this bad working feature representation. The inclusion of an accurate duration model prevents the model from inferring unrealistically long sequences of activities, even though the sensor data still indicates the relevant sensors are firing.

Looking at the results of experiment 3, we have seen that duration modeling still helps the performance in the cases of the B and C datasets. When a feature representation provides a good indication of which activity takes place, accurate duration modeling will not result in a significant gain in performance. Only the cases that we saw in the previous chapter, where activities performed in the same room are confused with each other, will benefit from the duration modeling. This explains why we still see an increase in performance for the B and C datasets in experiment 3.

Discriminative models

The problem with the last feature representation, described in the previous subsection, seems to affect the performance of the HMM, but not the performance of the CRF. This is because a discriminative model takes into account how well a particular feature is able to discriminate between classes, while a generative model does not take this into account. Discriminative models can assign a weight of 0 to a particular feature, effectively canceling out the contribution of that feature to the recognition process. Furthermore, discriminative models are able to deal with violations of the modeling assumptions, which means it can find a set of parameters that accurately discriminates the data, but are not the maximum likelihood parameters of the underlying distribution.

In the case of discriminative models, we see that accurate duration modeling can still help the performance. The inclusion of duration models basically gives the SMCRF a set of new tools for discriminating the data. As a result, the time to estimate the model parameters of the SMCRF takes up to fifty times longer than estimating the parameters of the CRF. Compared to the HSMM, learning the parameters takes as much as thirty-thousand times longer for the SMCRF. This is because the parameters of the HSMM can be estimated in closed form, while we use numerical optimization methods for the SMCRF.

The differences in performance between the SMCRF and the HSMM in experiment 3 does not give a conclusive picture. On the B and C datasets both models obtain nearly the same performance, while on the A datasets the HSMM performs better on the kitchen dataset and the SMCRF performs better on the bathroom dataset.

Future work

Besides the increase in performance resulting from accurate duration modeling, the use of semi-Markov models also has a number of important consequences which allow even more accurate modeling options. The explicit modeling of the duration of states allows more complex observation models. In conventional Markov models, the observation model is typically defined to depend only on the state at the timeslice the observation is made. In semi-Markov models, it is easier to think in terms of segments. An activity that lasts for one or more timeslices can be considered as a single segment. The observation model can be defined to depend on such a segment. This means the observation is dependent on all the states for as long as the activity lasts. Since activities vary in duration this means the observation model is of variable length and allows the inclusion of features that cover the entire segment of the activity. For example, it is possible to define a feature which states that the microwave was used at least once within the

execution of an activity. Semi-Markov models are therefore sometimes referred to as variable-length Markov models or segment models [111].

Another possible extension lies in the order of the Markov model. It can be beneficial to use a higher order Markov model to model transitions over longer periods of time. For example, in between two activities, it is likely that people generally spend some time doing something that we would label as 'other' activity. A typical sequence of an evening routine can therefore be, 'prepare dinner', 'other', 'eat dinner', 'other', 'brush teeth', 'other', 'sleeping'. A first order Markov model would only be able to capture the transition from and to 'other' to the various activities. While there is a very clear structure in this evening routine. The problem with conventional Markov models is that a higher order Markov model means we condition on a higher number of timeslices in the sequence. Since activities are not performed at a constant duration, it is not clear how many timeslices we should jump back to capture the sequence in the evening routine. By using semi-Markov models, we can use a single state variable to model an activity that lasts several timeslices. The state representation therefore becomes invariant of the duration of an activity. This makes it possible for higher order semi-Markov models to capture the relevant dependencies in an evening routine such as described above. The resulting transition model could therefore consider the probability of $p(sleeping \mid other, brushteeth)$.

4.8 Conclusions

Semi-Markov models differ from conventional Markov models in that they include a variable for modeling the duration of a state. This makes it possible to use any distribution for modeling state duration while in conventional Markov models we are forced to use the geometric distribution. In this chapter, we presented a generative semi-Markov model, the HSMM, which is an extension of the HMM discussed in the previous chapter. And we presented a discriminative semi-Markov model, the SMCRF, which is an extension of the CRF. We provided a comparison between generative and discriminative models with respect to duration modeling and illustrated how inaccurate duration modeling can result in erroneous classification in generative models, but results in error free classification in discriminative models.

For our experiments, we created six datasets which are subsets of the three real world datasets used in the previous chapter. This was done to overcome the computational issues with respect to the parameter estimation in the SMCRF. Our first experiment was aimed at determining the most suitable probability distribution for modeling the duration of an activity. We experimented with three unimodal distributions, a multivariate distribution and a histogram approach with various numbers of bins. The Gaussian distribution performed best in four

of the six datasets and is therefore the distribution that was used in the other experiments.

The other two experiments compared the performance of our generative and discriminative semi-Markov models to their conventional Markov model counterparts. In one experiment, the 'changepoint&last' feature representation was used and in the other experiment the 'changepoint' feature presentation was used. The last feature representation turned out to be less effective on the six subset datasets we created, because the subsets contain only a small number of sensors. This revealed that the HMM is very sensitive to such an ineffective feature representation, but that accurate duration modeling of the HSMM results in a significant gain in performance on five of the six datasets. When using the changepoint representation, a significant gain in performance is only observed in one of the six datasets. This shows that accurate duration modeling is important in generative models when the feature representation is unable to provide sufficient support for classification.

Discriminative models are less sensitive to the inclusion of ineffective feature representations, because they are more robust in dealing with violations of the modeling assumptions. The CRF therefore managed to significantly outperform the HMM in two of the six datasets when the ineffective feature representation was included. Accurate duration modeling in discriminative models results in an average increase in performance, but this increase is not significant. The HSMM and the SMCRF give similar performance on most datasets, neither model manages to consistently significantly outperform the other. Learning the model parameters of a SMCRF takes up to thirty-thousand times longer than estimating the parameters of the HSMM. This is because the parameters of the HSMM can be estimated in closed form, while we use numerical optimization methods for the SMCRF.

5

Hierarchical models

5.1 Introduction

In the introduction of this thesis we provided a distinction between action primitives, actions and activities. The models presented in the previous chapters recognize activities by modeling the relation between a sensor pattern and an activity. A pattern of sensor readings generally corresponds to an action primitive and so those models take into account the activities and the action primitives, but do not explicitly model the actions which make up an activity. Using hierarchical models allows us to incorporate the actions explicitly in our model and in this chapter, we test the hypothesis that this results in a more accurate representation of the internal structure of an activity.

In this chapter we present a two-layer hierarchical model for activity recognition and apply it to real world data. The top layer state variables of the model represent the activities that are performed, while the bottom layer state variables represent the actions. Although it is possible to train such a model using data which is annotated with labels of both activities and actions, in this chapter, we train the model using only labels for the activities. There are two advantages to this approach: 1) Annotating the data becomes significantly less involved when only the activities have to be annotated, 2) We do not force any structure upon the model with respect to the actions, but rather let the model find this structure in the data automatically. The automatic allocation of structure can be considered as a clustering task. The clusters found in the data do not necessarily have to be meaningful clusters that correspond to actual actions that are intuitive to humans. We therefore distinguish between the term 'action clusters' to refer to the actions

found through clustering and 'actions' to refer to the actions intuitive to humans.

In this chapter we answer the following questions:

- **How many action clusters are needed to model a given set of activities?** We do not use any labeled data to guide the model into determining which observation corresponds to which action. We can therefore experiment with various number of action clusters. More action clusters means the model is more flexible, but also means there are more model parameters that need to be estimated. Therefore, the ideal model will be a trade-off between this flexibility and the number of parameters needed.
- **Should each activity have its own set of action clusters, or should action clusters be shared among activities?** In designing a hierarchical model for activity recognition, we can choose to have a separate set of action clusters for each activity. Alternatively, we can use the same set of action clusters for representing all the activities, that is, share a single set of action clusters among all activities. For example, when distinguishing between the activities 'preparing spaghetti' and 'preparing a salad' both activities might involve an action that we can describe as 'cutting vegetables'. It is very well possible that cutting vegetables when preparing spaghetti is slightly different than cutting vegetables when preparing a salad. Using a separate set of action clusters for each activity would allow us to capture such differences. On the other hand, representing the action using a shared set means we have more training data available (i.e. part of the training data of both 'preparing spaghetti' and 'preparing a salad') to estimate the parameters of that single action cluster.
- **How does the performance of hierarchical modeling compare to the performance of Markov and semi-Markov models?** The use of a hierarchical model allows us to more accurately model the internal structure of an activity. But does increase in accuracy also result in an increase in model performance?

The remainder of this chapter is organized as follows. In Section 5.2, related work of hierarchical models is discussed. Section 5.3 provides the details of our hierarchical model and its learning and inference algorithms. Section 5.4 presents the experiments and results and in Section 5.5, we discuss these results. Finally, in Section 5.6, we will sum up the conclusions.

5.2 Related Work

One of the most popular probabilistic models for real world applications is the hidden Markov model (HMM). The HMM is closely related to stochastic regular

grammars, as they are both part of the family of stochastic finite state automata [16]. Stochastic regular grammars are the least expressive grammar out of all the formal grammars, according to the Chomsky hierarchy [22]. It is therefore interesting to consider the application of the more expressive stochastic context free grammar (SCFG) to such real world applications. However, a downside of using SCFGs is that the computation complexity of the Inside-Outside algorithm, used for inference in the SCFG, is cubic with respect to the length of the sequence that is being inferred. Furthermore, SCFGs allow hierarchies of infinite number of layers, while in many problem domains a hierarchy of a predefined fixed length is sufficient [82]. Therefore, instead of using SCFGs the use of hierarchical hidden Markov model (HHMM) is often more practical since it allows us to model hierarchies of a fixed number of layers and there exist efficient inference algorithms for it. Fine et al. presented an inference algorithm for the HHMM based on the Inside-Outside algorithm of the SCFG [36], which takes $O(T^3Q^L)$ time, where T is the length of the sequence, L is the number of layers in the hierarchy and Q is the number of states at each level of the hierarchy. Murphy showed that the HHMM can be represented as a dynamic Bayesian network (DBN) [112], thereby deriving a much simpler and more efficient inference algorithm, which takes at most $O(TQ^{2L})$. This has allowed the application of hierarchical models in many different domains, such as natural language processing [48], handwriting recognition [36] and information extraction [140, 146].

With respect to activity recognition, hierarchical models have mainly been applied to video data. There is work involving simple activities or actions such as distinguishing between entering and leaving a store [110]. But also work in which activities of daily living are recognized from video data [33, 91, 118, 123]. Nguyen et al. compare the performance of a learned HHMM, a hand-coded HHMM and a conventional HMM, the learned HHMM gives the best performance [118]. Duong et al. compare the performance of the HHMM and the hidden semi-Markov model (HSMM). In their work, the HHMM gives very poor performance in the recognition task which, according to the authors, is caused by a poorly estimated transition matrix. They do not explain why the hierarchical model is unable to learn the transition matrix, while the semi-Markov model is able to learn this matrix accurately [33]. In work by Luhr et al., hierarchical models consisting of several layers are hand crafted by closely inspecting the sequence of actions performed by the subject. Their preliminary results show that these models perform well in recognizing several cooking related activities [91]. Oliver et al. combine input from microphones, cameras and keyboard activity, to recognize several office activities such as having a phone conversation or giving a presentation. They use a layered hierarchical model in which the output of one layer serves as input for the next layer. The parameters for the HMM in each layer are learned independently of the other layers, therefore allowing very efficient parameter estimation. They compare the performance of their layered HMM to a conventional HMM and show a significant increase in performance

[123].

Activity recognition has also been performed using GPS data. Subramanya et al. jointly model the activity someone is performing and the location the subject is in. Examples of activities include walking, running and driving a vehicle, while examples of the locations are indoors and outdoors [151]. In work by Liao et al., more specific locations are modeled, such as being at a home, a friend's house or being at work. This allows them to accurately recognize activities such as working, sleeping and visiting [86].

Overall these works show that the potential of hierarchical models for activity recognition has been noted by other researchers. However, none of these works involve the recognition of activities from wireless sensor network data. This chapter contributes in this area by providing experimental results on several real world datasets, consisting of several weeks of data involving a large number of activities of daily living.

5.3 Hierarchical Hidden Markov Model

In a conventional HMM, the hidden state consists of a single state variable which generates an observation at each timeslice. When using a HHMM the hidden state consists of two (or possibly more) layers, with a state variable at each layer. Generally, the state variable at one layer generates the state variable at the layer below. The observations are generated by the bottom layer state variables, possibly combined with any of the state variables from the other layers.

In this section, we discuss the details of a two-layer hierarchical model for activity recognition and explain the inference and learning algorithms.

5.3.1 Model definition

We consider a two-layer hierarchical model for activity recognition. The top layer state variables y_t represent the activities and the bottom layer variables z_t represent the action clusters (Fig. 5.1). Each activity consists of a sequence of action clusters and the temporal ordering of the action clusters in such a sequence can vary between different executions of an activity. Of particular interest to us is the last action cluster that is performed at the end of an activity, because this action cluster signifies the end of the action cluster sequence and announces the start of a new sequence of action clusters. We therefore introduce a third variable, the finished state variable f_t, which is used as a binary indicator to indicate the bottom layer has finished its sequence.

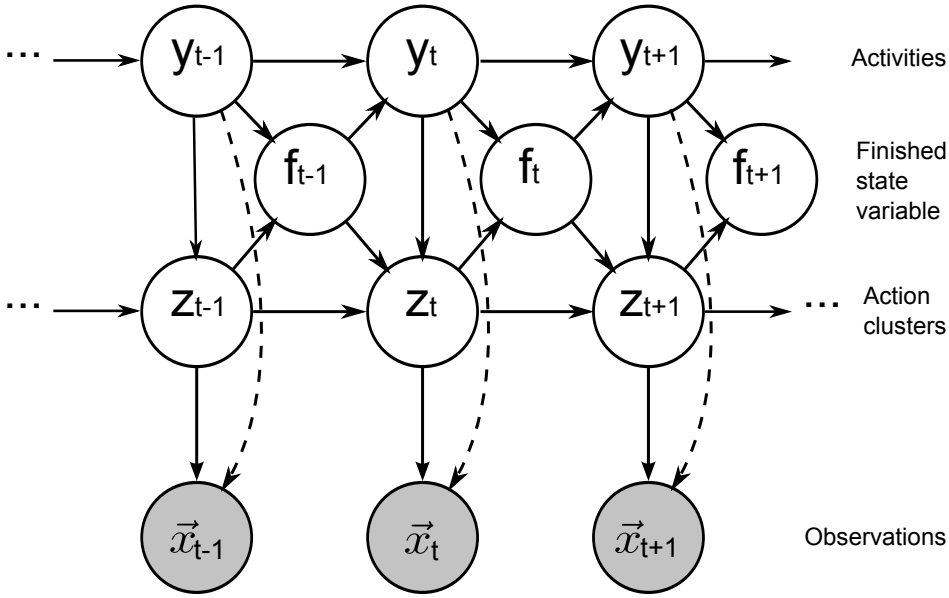


Fig. 5.1: The graphical representation of a two-layer HHMM. Shaded nodes represent observable variables, while white nodes represent hidden ones. The dashed line is an optional dependency relation; we can choose to model the observation probability as $p(\vec{x}_t \mid y_t, z_t)$ or as $p(\vec{x}_t \mid z_t)$.

We further explain the details of this model by going over all the factors of the joint probability distribution of hidden states and observations given by:

$$p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{f}_{1:T}, \mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\vec{x}_t \mid y_t, z_t) p(y_t \mid y_{t-1}, f_{t-1}) p(z_t \mid z_{t-1}, y_t, f_{t-1}) p(f_t \mid z_t, y_t)$$

where we have defined $p(y_1 \mid y_0, f_0) = p(y_1)$ and $p(z_1 \mid z_0, y_1, f_0) = p(z_1 \mid y_1)$ for the sake of notational simplicity. The entire model consists of a set of parameters $\theta = \{\pi_0, \pi_{1:Q}, A_0, A_{1:Q}, B, \phi\}$. The initial state parameters π and transition parameters A exist for both the top layer and bottom layer states. To distinguish between these two types of parameters, we include a 0 in the subscript to indicate that a parameter is of the top layer and an index of 1 to Q for each of the bottom layer parameters. The distributions of the bottom layer states depend on which top layer state the model is in and so there is a separate set of bottom layer state parameters for each possible top layer state, with Q being the number of top layer states. For example, if the model at one point is in the top state $y_t = k$, then the transition parameter A_k is used for the bottom layer state transitions. We now provide a detailed explanation of each of the factors that make up the joint probability and how they are parameterized.

At the first timeslice, the initial state distribution of the top layer states is represented by a multinomial distribution which is parameterized as $p(y_1 = j) = \pi_0(j)$. This top layer state generates a bottom layer state, also represented by a multinomial distribution and parameterized as $p(z_1 = j \mid y_1 = k) = \pi_k(j)$.

The factor $p(z_t = j \mid z_{t-1} = i, y_t = k, f_{t-1} = f)$ represents the transition probabilities of the bottom layer state variable. These transitions allow us to incorporate the probability of a particular temporal order of action clusters with respect to a given activity. A transition into a new state z_t, depends on the previous bottom layer variable z_{t-1}, the current top layer state variable y_t and the finished state variable f_{t-1}. Two distributions make up this factor, depending on the value of the finished state variable f_{t-1}. If in the previous timeslice the bottom layer state sequence ended ($f_{t-1} = 1$), a new sequence of bottom layer states starts at this timeslice and therefore the top layer state generates a bottom layer state using the same distribution as we saw at the first timeslice, parameterized by the set of parameters $p(z_t = j \mid z_{t-1} = i, y_t = k, f_{t-1} = f) = \pi_k(j)$. In case the bottom layer state sequence did not end ($f_{t-1} = 0$), a transition to a new bottom layer state is made using the transition matrix parameterized as $p(z_t = j \mid z_{t-1} = i, y_t = k, f_{t-1} = f) = A_k(i, j)$. These two cases can be compactly formulated as:

$$p(z_t = j \mid z_{t-1} = i, y_t = k, f_{t-1} = f) = \begin{cases} A_k(i, j) & \text{if } f = 0 \\ \pi_k(j) & \text{if } f = 1 \end{cases} \tag{5.1}$$

Transitions of the top layer state variables are represented by the factor $p(y_t = j \mid y_{t-1} = i, f_{t-1} = f)$. This factor is similar to the transition distribution of an HMM, except that it also depends on the finished state variable f_{t-1}. This dependency is important because we want to restrict the model in transitioning to a different top layer state as long as the bottom layer state sequence has not finished. When a bottom layer state sequence did not finish, the top layer state variable continues into the next timeslice with the same state value ($y_t = y_{t-1}$). Once the bottom layer state sequence has ended, a transition of the top layer state is made according to a transition matrix parameterized as $p(y_t = j \mid y_{t-1} = i, f_{t-1} = f) = A_0(i, j)$. These two cases can be compactly formulated as:

$$p(y_t = j \mid y_{t-1} = i, f_{t-1} = f) = \begin{cases} \delta(i, j) & \text{if } f = 0 \\ A_0(i, j) & \text{if } f = 1 \end{cases} \tag{5.2}$$

where $\delta(i, j)$ is the Kronecker delta function, giving 1 if $i = j$ and 0 otherwise.

The probability of a bottom layer state sequence finishing is represented by the factor $p(f_t = f \mid y_t = j, z_t = l)$. This factor depends on both the bottom layer state z_t and the top layer state y_t. Even though the variable f_t indicates whether z_t is a finishing state, it is important that the distribution is also conditioned on the top layer state y_t. This is because the probability of a particular action cluster being

the last action cluster for that activity can differ among activities. The factor is represented using a binomial distribution, parameterized as $p(f_t = f \mid y_t = j, z_t = l) = \phi_f(j, l)$.

Two possible observation models

In the graphical representation of our hierarchical model, shown in Figure 5.1, there is a dashed line between the top layer state variables y_t and the observation variables $\vec{x}_t$. This line represents an optional dependency relationship, because we wish to experiment with two types of observation models. If we do take the dependence relation into account, our observation model is represented by the factor $p(\vec{x}_t \mid y_t, z_t)$. In this model, each combination of top and bottom state values gets its own set of parameters. Alternatively, if we do not include the dependence relation, our observation model is represented by the factor $p(\vec{x}_t \mid z_t)$. In this case, the observation model is independent of the top layer state variable. Note that in the transition probabilities of the bottom layer state variable described above, there still exists a dependency on the top layer state, regardless of which observation model is used. The same holds for the finished state probability distribution.

Observations are modeled as independent Bernoulli distributions, as was done in the previous chapters. The two observation models are therefore defined as:

Model 1:

$$p(\vec{x}_t \mid y_t, z_t) = \prod_{n=1}^{N} p(x_n \mid y_t, z_t) \tag{5.3}$$

$$p(x_n \mid y_t = j, z_t = k) = \mu_{jkn}^{x_n}(1 - \mu_{jkn})^{(1-x_n)} \tag{5.4}$$

Model 2:

$$p(\vec{x}_t \mid z_t) = \prod_{n=1}^{N} p(x_n \mid z_t) \tag{5.5}$$

$$p(x_n \mid z_t = k) = \mu_{kn}^{x_n}(1 - \mu_{kn})^{(1-x_n)} \tag{5.6}$$

where N is the number of sensors used for observation. Model 1 requires Q times more parameters than Model 2, because of the additional dependency on the top layer states, with Q being the number of top layer state values. The observation parameters are collectively represented by a variable $B = \{\mu_{jkn}\}$ for Model 1 and $B = \{\mu_{kn}\}$ for Model 2.

5.3.2 Inference using a flattened implementation

Inference in a HHMM can be done by applying the junction tree algorithm [75]. However, by representing our HHMM as a HMM, we can simply apply the Viterbi algorithm and forward-backward algorithm for HMMs [112]. We can flatten our HHMM to a HMM by creating a HMM state for every possible combination of states in the HHMM. This means that we create a HMM with a state space containing all the combinations of y_t, z_t and f_t variables. Since we can always determine which HMM state value corresponds to which combination of HHMM state values, we will continue to use the HHMM state values to index the variables within the flattened implementation.

The parameters of the resulting HMM can be constructed from the parameters of the HHMM using the following equations. If for a particular variable no value is specified, then the calculation is independent of the value of that variable. The initial state distribution π is formed by combining the initial state of both the top and bottom layer $\pi(y_1 = i, z_1 = j, f_1) = \pi_0(i)\pi_i(j)$. Observation parameters remain unchanged. And the transition parameters A depend on whether the bottom layer state sequence ended in the previous timeslice, giving:

$$A(y_{t-1} = i, z_{t-1} = k, f_{t-1} = 0, y_t = j, z_t = l, f_t = f) = \delta(i, j)A_j(k, l)\phi_f(j, l) \quad (5.7)$$

$$A(y_{t-1} = i, z_{t-1} = k, f_{t-1} = 1, y_t = j, z_t = l, f_t = f) = A_0(i, j)\pi_j(l)\phi_f(j, l) \quad (5.8)$$

A downside of this flattened implementation is that the combined state space requires a lot of memory. For a large number of state values and a large number of layers, this easily consumes too much space to be tractable. Fortunately, since our HHMM consists of only two layers this approach is still usable.

5.3.3 Parameter learning

Parameters are learned iteratively using the Expectation Maximization (EM) algorithm [8] which can be applied to our flattened model. The E-step consists of using the forward-backward algorithm to calculate the probability distribution $p(\mathbf{y}_{1:T}, \mathbf{z}_{1:T}, \mathbf{f}_{1:T} \mid \mathbf{x}_{1:T}, \theta)$. From this distribution, we can calculate the expectation and reestimate the parameters in the M-step. The equations for reestimating the

parameters are given by:

$$\pi_0(j) = p(y_1) \tag{5.9}$$

$$\pi_k(j) = \frac{\sum_{t=1}^{T} p(y_t = k, z_t = j, f_{t-1} = 1)}{\sum_{t=1}^{T} \sum_{y_t} p(y_t, z_t = j, f_{t-1} = 1)} \tag{5.10}$$

$$A_0(i,j) = \frac{\sum_{t=2}^{T} p(y_t = j, y_{t-1} = i, f_{t-1} = 1)}{\sum_{t=2}^{T} \sum_{y_t} p(y_t, y_{t-1} = i, f_{t-1} = 1)} \tag{5.11}$$

$$A_k(i,j) = \frac{\sum_{t=2}^{T} p(z_t = j, z_{t-1} = i, y_t = k, f_{t-1} = 0)}{\sum_{t=2}^{T} \sum_{z_t} p(z_t, z_{t-1} = i, y_t = k, f_{t-1} = 0)} \tag{5.12}$$

$$\phi_f(j,l) = \frac{\sum_{t=1}^{T} p(f_t = f, y_t = j, z_t = l)}{\sum_{t=2}^{T} \sum_{f_t} p(f_t, y_t = j, z_t = l)} \tag{5.13}$$

$$\tag{5.14}$$

Observation Model 1: $\mu_{jkn} = \frac{\sum_{t=1}^{T} \delta(x_t^n,1) p(y_t=j,z_t=k)}{\sum_{t=1}^{T} p(y_t=j,z_t=k)}$

Observation Model 2: $\mu_{kn} = \frac{\sum_{t=1}^{T} \delta(x_t^n,1) p(z_t=k)}{\sum_{t=1}^{T} p(z_t=k)}$

The procedure of calculating the forward-backward probabilities and reestimating the parameters is repeated until the parameter values converge.

5.4 Experiments

Our experiments are aimed at determining which number of action clusters are needed for modeling activities, which observation model gives the best performance and how the performance of hierarchical models compares to the performance of the HMM and the HSMM. Our first experiment compares the performance of the hierarchical model using Observation Model 1 to the performance of the HMM and the HSMM. The second experiment makes the same comparison, but uses Observation Model 2. In both experiments, results are given for various number of action clusters. The remainder of this section first presents the details of our experimental setup, then describes the experiments and the results and finally discusses the outcomes.

5.4.1 Experimental Setup

We use the same datasets and activities that were used in Chapter 3. A summary of relevant details for each dataset can be found in Table 5.1. The datasets include annotation of activities, but do not include annotation of actions. Since we do

	House A	House B	House C
Activities	10	14	16
Sensors	14	23	21
Days of data	25 days	13 days	18 days

Tab. 5.1: Information about the datasets recorded in three different homes using a wireless sensor network.

not have any ground truth for the actions and because we are only interested in using action clusters for modeling purposes, our evaluation is based solely on the inferred activities. In Section 3.7.2, we introduced the precision, recall and F-measure and showed they are reliable measures for evaluating the performance of our model. The experimental results of Chapters 3 and 4 showed that the changepoint feature representation consistently gives a good performance, therefore we use that feature representation for the experiments in this chapter.

Data obtained from the sensors is transformed to the changepoint representation and discretized in timeslices of length $\Delta t = 60$ seconds. We split our data into a test and training set using a 'leave one day out' approach. In this approach, one full day of sensor readings is used for testing and the remaining days are used for training. We cycle over all the days in the dataset, so that each day is used once for testing. For the HMM and the HSMM, we present the results by taking the average over all the test days. In the case of the HHMM, we do five separate runs of the entire experiment and average the results over these five runs and then average over all the days. This is done because the EM algorithm requires a random initialization of the parameters. The EM algorithm is run until convergence, which guarantees a local maximum solution. Because we only use labeled data for the activities, the EM algorithm performs a clustering operation on the data with respect to the action clusters and it is likely that there are many local maxima in the parameter space. The initialization of the initial parameters for the first iteration of the EM algorithm plays an important role here. If a parameter value is initialized to a value close to 0, it will take long before the value converges. We therefore initialize our parameters uniformly and add normally distributed jitter with a mean of 0 and a variance of 0.01. Because we calculate our performance measures by taking the average over multiple runs, we reduce the impact of initialization and therefore obtain a more reliable result. To get an impression of how the model performs when a good initialization of the parameters is used, we also report the maximum performance possible. This result is obtained by selecting the highest performing day from the five runs for each of the cross validation days.

5.4.2 Experiment 1: Observation Model 1

In this experiment, we use Observation Model 1 ($p(\vec{x}_t \mid y_t, z_t)$). We compare the performance of our HHMM to the performance of the HMM and HSMM. Furthermore, we experiment with various number of action clusters. The average F-measure performance over five runs for various number of action clusters is given in Figure 5.2 for all three houses.

We see that the performance of the HHMM is equal to the HMM when a single action cluster per activity is used. Using a single action cluster for each activity is equivalent to using an HMM and therefore results in the same performance. Generally the best performance for the HHMM is obtained when using two or three action clusters, the performance decreases as more action clusters are considered.

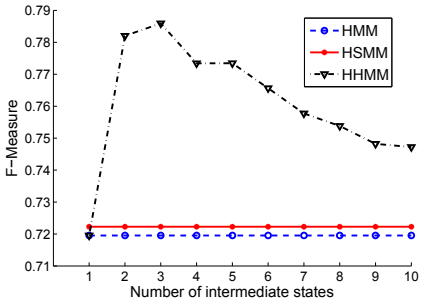
Table 5.2 shows the same results, but now the precision, recall and standard deviations are included. The maximum performance over five runs is also included in the table. These results show us that the increase in performance is the result of an increase in both precision and recall. To determine the significance of our results we used a one-tail student t-test with matching paired days. The increase in F-measure performance of the HHMM, taken over an average of five runs, compared to the F-measure performance of the HMM and the HSMM is significant for houses A and C, at a confidence interval of 95%. When making the same comparison using the maximum performance of the HHMM over five runs, the increase in F-measure performance is significant for all three houses, at a confidence interval of 95%. Finally, when comparing the performance of the average performance over five runs to the maximum performance, we see that the maximum performance generally results in an increase of at least three percent points.

5.4.3 Experiment 2: Observation model 2

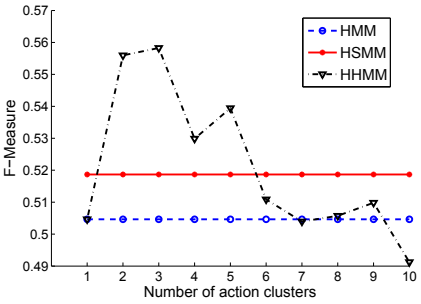
In experiment 2 Observation Model 2 ($p(\vec{x}_t \mid z_t)$) is used. The experimental setup is similar to experiment 1 and the average F-measure performance over five runs for various number of action clusters is given in Figure 5.3 for all three houses.

We see that in Houses A and B, the HHMM does not manage to perform better than the HMM or the HSMM. In House C, we see a slight improvement in performance over the HMM and the HSMM, when 15 action clusters are used, but this increase in not significant. Overall, the best performance is obtained when using 10 or 15 action clusters. Using more or less action clusters than that quickly results in a significant decrease in performance.

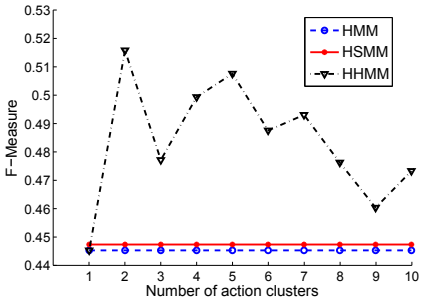
Table 5.3 shows the same results, with the precision, recall, standard deviations



(a) House A



(b) House B



(c) House C

Fig. 5.2: Experiment 1: Plot of the F-measure performance of the hidden Markov model (HMM), hidden semi-Markov model (HSMM) and hierarchical hidden Markov model (HHMM) using Observation Model 1. The number of action clusters signifies the number of state values that are used for the bottom layer state variable of the HHMM.

	Model	Type	Num. Actions	Precision	Recall	F-Measure
House A	HMM	-	-	70 ± 16	74 ± 13	72 ± 14
	HSMM	-	-	70 ± 17	75 ± 13	72 ± 15
	HHMM	Avg.	3	76 ± 13	81 ± 10	79 ± 11
		Max.	3	81 ± 13	84 ± 10	82 ± 11
	Model	**Type**	**Num. Actions**	**Precision**	**Recall**	**F-Measure**
House B	HMM	-	-	45 ± 17	60 ± 14	50 ± 16
	HSMM	-	-	46 ± 16	61 ± 14	52 ± 15
	HHMM	Avg.	3	51 ± 9	62 ± 7	56 ± 8
		Max.	3	56 ± 12	67 ± 8	60 ± 10
	Model	**Type**	**Num. Actions**	**Precision**	**Recall**	**F-Measure**
House C	HMM	-	-	41 ± 8	50 ± 12	45 ± 8
	HSMM	-	-	41 ± 8	50 ± 11	45 ± 8
	HHMM	Avg.	2	49 ± 16	55 ± 15	52 ± 15
		Max.	2	54 ± 16	59 ± 15	55 ± 15

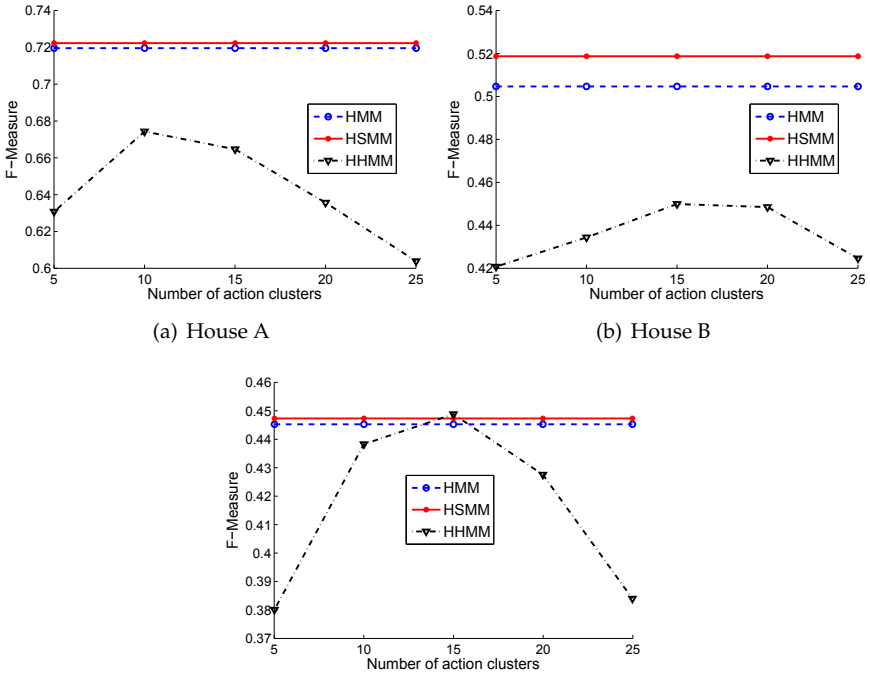
Tab. 5.2: Experiment 1: Precision, recall and F-measure for hidden Markov model (HMM), hidden semi-Markov model (HSMM) and hierarchical hidden Markov model (HHMM). Type indicates the type of performance measure that was used, Avg. stands for the average over five runs and Max. stands for the maximum over five runs. Num. Actions represents the number of state values that were used for the bottom state layer variable.

and maximum performance over five runs included. When comparing the performance of the HHMM using the maximum performance over five runs to the performance of the HMM and the HSMM, we see that the HHMM on average performs better in the case of houses A and C. However, this increase in performance is not significant at a confidence interval of 95%. Finally, when comparing the performance of the average performance over five runs to the maximum performance, we see that the maximum performance can result in an increase ranging from three to six percent points.

5.5 Discussion

Model of Observation

The results from our experiments show that observation model 1 gives significantly better performance than Observation Model 2. When using Observation Model 1 a separate set of action clusters is used for each activity, while with observation model 2 a single set of action clusters is used for all activities. Our result is therefore rather counterintuitive because when using a single set of action clusters for all activities, the model is able to reuse certain action clusters for modeling multiple activities.



(a) House A

(b) House B

(c) House C

Fig. 5.3: Experiment 2: Plot of the F-measure performance of the hidden Markov model (HMM), hidden semi-Markov model (HSMM) and hierarchical hidden Markov model (HHMM) using Observation Model 2. The number of action clusters signifies the number of state values that are used for the bottom layer state variable of the HHMM.

	Model	Type	Num. Actions	Precision	Recall	F-Measure
House A	HMM	-	-	70 ± 16	74 ± 13	72 ± 14
	HSMM	-	-	70 ± 17	75 ± 13	72 ± 15
	HHMM	Avg.	10	67 ± 15	69 ± 12	67 ± 13
		Max.	10	73 ± 14	73 ± 12	73 ± 13
	Model	**Type**	**Num. Actions**	**Precision**	**Recall**	**F-Measure**
House B	HMM	-	-	45 ± 17	60 ± 14	50 ± 16
	HSMM	-	-	46 ± 16	61 ± 14	52 ± 15
	HHMM	Avg.	15	42 ± 11	49 ± 8	45 ± 9
		Max.	15	46 ± 12	52 ± 8	48 ± 10
	Model	**Type**	**Num. Actions**	**Precision**	**Recall**	**F-Measure**
House C	HMM	-	-	41 ± 8	50 ± 12	45 ± 8
	HSMM	-	-	41 ± 8	50 ± 11	45 ± 8
	HHMM	Avg.	15	43 ± 9	47 ± 9	45 ± 9
		Max.	15	48 ± 9	51 ± 10	49 ± 9

Tab. 5.3: Experiment 2: Precision, recall and F-measure for hidden Markov model (HMM) and hierarchical hidden Markov model (HHMM). Type indicates the type of performance measure that was used, Avg. stands for the average over five runs and Max. stands for the maximum over five runs. Num. Actions represents the number of state values that were used for the bottom state layer variable.

The explanation for this lack of increase in performance lies in the difficulty to find action clusters and the lack of labels for the actions. Observation Model 1 is defined to be dependent on both the action clusters and the activities. Because we have labeled data for the activities, clustering the data into action clusters is done for each activity separately. On the other hand, Observation Model 2 only depends on the action clusters, for which there is no labeled data available. Action clusters therefore need to be found in the data of all activities put together. Even if consistent action clusters can be allocated in the data, there is no guarantee that this fit is ideal for the classification of activities.

In future work, it would be interesting to see if the inclusion of labels for the actions would result in a better performance when using Observation Model 2. Manually labeling the actions and explicitly appointing actions that are shared among activities could strongly improve the performance. It would not be necessary to label all the timeslices with actions, a partial labeling provided by a human expert would help to cluster the data in a more sensible manner.

Number of action clusters used

The number of action clusters used for modeling the data has a strong impact on the performance of the model. When Observation Model 1 is used, the use of two or three action clusters results in the best performance. While when using

Observation Model 2, ten or fifteen action clusters gives the best performance. Using more or less action clusters quickly results in a significant decrease in performance in both models. In the case of using more action clusters, this decrease in performance can be explained by the increase in the number of parameters. Because we have a limited amount of training data available, too many parameters will result in an inaccurate estimation of these parameters [30]. Too few action clusters will limit the model in its expressive power, so that even with accurately estimated parameters it will not be able to model the data accurately.

The optimal number of action clusters for Observation Model 2 is a lot higher than the optimal number of action clusters for Observation Model 1. The explanation for this difference lies in the number of free parameters available for modeling the action clusters. In Observation Model 2, the total number of observation parameters that the model uses for each sensor is equal to the number of action clusters used. On the other hand, when using Observation Model 1, the total number of observation parameters used for each sensor is equal to the number of action clusters times the number of activities. This is because the observation probability in Observation Model 1 is conditioned on both the action clusters and the activities. A comparison between the two observation models with respect to the number of action clusters used should therefore consider cases in which the same number of parameters for each sensor is used. In the case of house A, there were ten activities available and so when considering the use of Observation Model 1 with two action clusters, twenty parameters are needed for each sensor. In the case of Observation Model 2, using ten to fifteen action clusters gives the best performance and ten to fifteen parameters are needed per sensor. Observation Model 2 therefore requires less observation parameters than Observation Model 1 for its best performance. Still, Observation Model 1 manages to perform better than Observation Model 2, even though it requires more parameters.

Maximum performance

The difference in performance between the average performance over five runs and the maximum performance tell us that the parameter estimation method of the HHMM is quite sensitive to initialization. The use of the maximum performance measure gives us insight into the maximum capabilities of the HHMM, but should not be used to present the performance of the HHMM in a real world setting. The problem is that in a real world setting, we are dealing with a sequence of observations for which there is no ground truth available. It is therefore not possible to determine which set of parameters give the best results for the given observation sequence. We could do multiple runs of the EM algorithm, using different random initializations and select the set of parameters that give the highest

likelihood on the training data. However, validating the quality of different runs on training data can result in overfitting and therefore the proper way of doing this is to use a validation set. A validation set is constructed by excluding part of the training data from the training procedure and using that data for evaluating which set of parameters is best suited. Of course the amount of training data used during optimization influences the quality of the estimated parameters. And using too little data for the validation set still risks the possibility of overfitting. Therefore, the balance between the size of the data used for training and the size of the validation set should be carefully considered. Alternatively, a different method of initialization might result in an average performance that is closer to the maximum performance reported in this chapter.

Modeling the internal structure of activities

When using Observation Model 1, the HHMM managed to significantly outperform the HMM and the HSMM in two of the three houses. This increase in performance is mainly due to differences in the observation model and the modeling of transition probabilities of the bottom layer state variable.

The inclusion of action clusters allows the model to divide activities into separate stages, based on the actions that are performed. By modeling the transition probabilities between consecutive action clusters, we are able to calculate the probability of a particular temporal ordering of action clusters within an activity. To be able to model the order in which an activity is performed when using a HMM, we would have to include features that indicate that sensor A fire before sensor B. With a large number of sensors, the number of transitions grows quickly and so do the number of parameters that need to be estimated. By using action clusters as an abstraction of sensor patterns, we can limit the number of possible transitions and model temporal ordering using less parameters.

Another advantage of modeling the transition probabilities of the bottom layer state variable is it allows more accurate duration modeling. We have seen in Chapter 4 how the inclusion of a duration variable allows us to use any distribution we want for modeling the duration of a state. The HHMM does not include a duration variable and therefore we are restricted to the geometric distribution, which follows from the self transition of states. However, our HHMM contains state transitions on two layers. This means that the probability distribution of the duration of an activity is not only based on the self transitions of the top layer state variable, but also on all possible transitions (not only self-transitions) of the bottom layer state variable. This probability distribution corresponds to a convolution of geometric distributions and therefore allows more accurate modeling than a single geometric distribution.

5.6 Conclusion

Hierarchical models allow us to model the relationships between action primitives, actions and activities. We presented a two layer hierarchical hidden Markov model for activity recognition in which one of the layers corresponds to action clusters and one layer corresponds to the activities. We speak of action clusters because clusters of actions are automatically found in the data and do not necessarily have to be actions that are intuitive to humans. The use of multiple layers allowed the choice between two observation models. One model in which a separate set of action clusters is used for each activity, and another model in which action clusters are shared among activities. To estimate the model parameters, we assumed to have annotated data for the activities, but no annotation for the actions. We are therefore free to choose the number of action clusters to use.

Experiments on three real world datasets revealed that using a separate set of action clusters for each activity works best. This is because allocating action clusters for each activity separately results in clusters that are meaningful with respect to the classification of activities. Using a shared set of action clusters does not necessarily result in meaningful clusters and therefore using that observation model gives a significantly lower performance. The performance of that model might be increased by using labels for the actions provided by a human annotator.

The use of two or three action clusters gives the best performance, when using a separate set of action clusters for each activity. Too few action clusters does not provide the model with enough expressive power, while too many action clusters results in too many parameters for which there is too little data to estimate them accurately.

Our proposed hierarchical model significantly outperforms the HMM and the HSMM in two of the three real world datasets, with an increase of 7 percentage points in F-measure performance for both datasets. This gain in performance is caused by the ability of the hierarchical model to model the temporal order of action clusters, and because the use of multiple layers results in a duration distribution which corresponds to a convolution of geometric distributions.

6

Transfer Learning

6.1 Introduction

In the previous chapters, we have discussed various models for activity recognition. These models, and most other state of the art activity recognition models, are supervised models that require annotated datasets to learn the model parameters [32, 71, 86, 165]. This annotation has to be provided by a human observer who either actively annotates the activities as the dataset is recorded, or passively annotates them by inspecting a video recording of the subject performing the activities. Both active and passive annotation are time consuming operations and are therefore considered very expensive.

The application we have in mind for applying activity recognition systems is the health monitoring of elderly in their own homes. Therefore, it is expected that in the near future such systems will be installed in many different homes. However, due to differences in both the layout of houses and the behavior of their inhabitants, an activity recognition model trained for one home cannot be directly used in another home. Instead, we will have to record and annotate a separate dataset for each home we wish to install an activity recognition system in. Since annotating a dataset is such an expensive operation, this is far from ideal.

We hypothesize that despite the differences among houses, there are also similarities among houses with respect to the execution of activities. If this hypothesis is valid, annotated datasets can provide us with generic information about how

[1] The material in this chapter is largely drawn from [64, 66].

people perform activities. In this chapter, we present learning methods that use existing annotated datasets of different houses to estimate the model parameters for a new house. We consider each house that we perform activity recognition in as a separate classification task. Using datasets from various classification tasks to learn the model parameters for a different but related classification task is known as transfer learning [5, 15, 162]. We will answer the following questions with respect to transfer learning for models of activity recognition:

- **Is transfer learning for models of activity recognition possible?** We hypothesize that there are similarities among houses with respect to the execution of activities. Are these similarities really present, and are they numerous enough to provide a useful contribution to the estimation of model parameters?

- **How can we deal with differences in the layout of houses?** There is a sensor network in each house that we perform activity recognition in. Due to differences in the layout of houses, there might be differences between sensor networks. For example, a different number of sensors might be used, or sensors might sense different properties. The sensor data is represented in a feature space, where in our approach one sensor typically corresponds to one dimension in the feature space. Differences in sensor networks therefore result in feature spaces with different dimensionality and features with different semantic interpretations. To be able to perform transfer learning, we need to somehow overcome these differences.

- **How do we deal with differences in behavior of inhabitants?** Differences in the behavior of inhabitants can be subtle (e.g. one person prefers sugar in their coffee, while another prefers their coffee black) or can mean that an activity is performed in a very different way (e.g. one person uses instant coffee, while another uses a coffee machine). We could ignore such differences and create a generic activity recognition model that we use for all houses. Or we could take the differences into account and create a specific model for each house we perform activity recognition in. How can we perform transfer learning such that we end up with a specific model for each house? How does the recognition performance of such a specific model compare to a generic model?

- **Does the inclusion of unlabeled data help the recognition performance?** Obtaining sensor data that is annotated with labels of activities is expensive. However, collecting large amounts of sensor data that we simply do not annotate is inexpensive. Such unlabeled data contains specific information about sensor patterns that typically occur in the house we wish to perform activity recognition in. Even though the data does not include labels of activities, including such data during training might result in better estimated model parameters.

The rest of this chapter is organized as follows. Section 6.2 describes related work of both activity recognition and transfer learning. Section 6.3 provides a brief summary of the hidden Markov model (HMM), the model used for recognizing the activities. In Section 6.4, we describe transfer learning in detail. Section 6.5 presents the experiments and results and in section 6.6, we discuss these results. Finally, in Section 6.7 we sum up our conclusions.

6.2 Related Work

When using transfer learning, we typically have a classification task for which we wish to estimate the model parameters, and we have several classification tasks for which we have datasets that can be used to obtain information from. The tasks that provide us with information are called *source* tasks, and the task for which we are estimating the model parameters is called the *target* task. In a typical transfer learning setup, we generally have annotated datasets available for the source tasks, while for the target task, we either have no datasets, a dataset consisting of unlabeled data or a dataset containing very little annotation.

Early work on transfer learning primarily focused on multi-task learning in which several classification tasks were learned jointly, yielding a better performance than learning each classification tasks separately [5, 15, 162]. For example, the goal in newsgroup classification tasks, is to classify which newsgroup a particular document belongs [26, 132]. Consider that our target task is classifying whether a document comes from a newsgroup about space or about hardware. When including training data of several source tasks of other newsgroups such as religion, baseball and motorcycles the performance of the target task improves significantly [132]. This is because the other newsgroups provide information about the co-occurrence of words. A word such as 'moon' might often occur together with the word 'rocket'. If the word 'rocket' did not occur in the space newsgroup dataset of the target task, but the word 'moon' did, the classifier can still learn that 'rocket' is descriptive for the space newsgroup, because it occurs often together with 'moon' in the datasets of the source tasks.

The optimal way to perform transfer learning is still an active topic of research. One approach that seems to work well with probabilistic models is to learn a prior distribution over the model parameters from the source task datasets and use that prior to learn the model parameters of the target task [83, 132]. The prior provides an initial estimate of the model parameters of the target task and the influence of the prior decreases in case more training data of the target task is obtained. Therefore, the use of a prior provides a natural mechanism to balance the effect of the prior distribution and the available amount of training data for the target task while learning the model parameters. This approach has been successfully applied to independent and identically distributed (i.i.d.) data,

where a discriminative model was used for collaborative filtering [83] and binary text classification [132]. Datapoints in an i.i.d. datasets are all independent of each other and so there are no temporal dependencies among the datapoints. Activity recognition, however, presents us with two important challenges: First, our measurements are part of a time series, and are therefore not i.i.d. Second, we can easily obtain large amounts of unlabeled data from a house and would like to use that data for our parameter estimation. This rules out the use of discriminative models, because it is not possible to estimate the parameters of a discriminative model using unlabeled data. The methods we propose in this chapter apply transfer learning to time series, using a generative model to allow the use of both labeled and unlabeled data during learning.

6.3 Model for Activity Recognition

In this section, we refine our notation to account for the use of unlabeled data and we go over the details on our model for activity recognition.

The data obtained from the sensors is discretized into T timeslices of length Δt. A single feature value is denoted as x_t^i, indicating the value of feature i at timeslice t, with $x_t^i \in \{0, 1\}$. Feature values can either represent the raw values obtained directly from the sensor, or can be transformed according to a fixed function. In a house with N installed sensors, we define a binary observation vector $\vec{x}_t = (x_t^1, x_t^2, \ldots, x_t^N)^T$. A variable representing the activity at timeslice t is denoted with $y_t \in \{1, \ldots, Q\}$ for Q possible activities. This variable is used for inferring a sequence of activities by a model of activity recognition. The annotation provided by a dataset is defined as a variable $l_t \in \{0, \ldots, Q\}$ denoting the value of the label for an activity. A label value of 0 indicates that no annotation is available for that particular timeslice. The combination of a sequence of features and a sequence of labels forms a dataset for activity recognition and is denoted as $\mathcal{D} = \{\mathbf{x}_{1:T}, l_{1:T}\}$.

To infer a sequence of activities $\mathbf{y}_{1:T}$ from a sequence of features $\mathbf{x}_{1:T}$, we use the HMM that was introduced in Chapter 3. We provide a brief summary of the details of the HMM here. The joint probability distribution factorizes as:

$$p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) = p(y_1) \prod_{t=1}^{T} p(\vec{x}_t \mid y_t) \prod_{t=2}^{T} p(y_t \mid y_{t-1}). \tag{6.1}$$

The different factors represent: the initial state distribution $p(y_1)$ is represented as a multinomial distribution parameterized by π; the observation distribution $p(\vec{x}_t \mid y_t)$ is a combination of independent Bernoulli distributions, parameterized by $B = \{\mu_{in}\}$; the transition distribution $p(y_t \mid y_{t-1})$ is represented as a collection

of multinomial distributions, parameterized by $A = \{a_i\}$. The entire model is therefore parameterized by a set of three parameters $\theta = \{\pi, A, B\}$.

6.4 Transfer Learning for Activity Recognition

In transfer learning for activity recognition, we assume we have R source houses and a single target house. We denote variables related to the target house with a superscript (0) and variables related to a source house with a superscript of the index of the source house. We assume there is a fully annotated dataset $\mathcal{D}^{(i)}$ with $i \in 1 : R$ for each of the source houses. The dataset $\mathcal{D}^{(0)}$ of the target house can contain a mix of unlabeled and labeled timeslices. We assume each house can have a different number of sensors $N^{(i)}$ with $i \in 0 : R$, but for all houses the same number of activities Q are considered. This means the features of each dataset can be of different dimensionality, but the same labeling is used in each dataset.

Transfer learning comes down to estimating the model parameters $\theta^{(0)}$ of the HMM for the target house. We want to include the source house datasets in our parameter estimation method and therefore have to deal with two issues. First, how do we deal with the different dimensionality of the features of all the datasets. Second, how can we estimate the model parameters of the target house using both the source and target datasets.

6.4.1 Mapping using Meta Features

The differences in dimensionality of the features prevents us from combining the datasets from the houses. To solve this, we need to introduce some kind of mapping such that each sensor feature space with dimensionality $N^{(i)}$ is mapped to a common feature space of dimensionality M. We use meta features [83] for this mapping, which are features that describe the properties of the actual features. Each sensor feature is described by one or more meta features, for example, a sensor on the microwave might have one meta feature describing the sensor is located in the kitchen, and another that the sensor is attached to a heating device.

In our approach, we define a set of M meta features and manually create a mapping vector $g_n^{(i)}$ to be a row vector of binary indicators, indicating which meta features correspond to sensor n of house i (Table 6.1). Together these vectors form a $N \times M$ mapping matrix $G^{(i)} = \left\{g_n^{(i)}\right\}$ which consists of the vertical concatenation of the meta feature row vectors. Defining a set of meta features requires insight on the effect the feature space will have on classification and therefore needs to be done by a pattern recognition expert. Once the meta features have been defined, assigning a set of meta features to sensors requires only common sense and is a trivial task that can be performed by an end user.

House	Sensors	Bathroom Entrance	Kitchen Heating	Kitchen Storage	Toilet Entrance
House A	Stove	0	1	0	0
	Bathroom door	1	0	0	0
	Toilet door	0	0	0	1
House B	Microwave	0	1	0	0
	Bathroom door	1	0	0	1

Tab. 6.1: Example of sensors (rows) being represented by meta features (columns) for two houses. The row of binary values corresponds to the mapping vector for that particular sensor. In this example the house A bathroom and toilet are located in separate rooms and therefore both have a separate entrance, in house B they are located in the same room and therefore have a single entrance.

6.4.2 Estimating the model parameters

With the mapping to the meta feature space, we can solve the issues with the differences in dimensionality among the feature spaces of the datasets. This allows us to combine the datasets to estimate the model parameters $\theta^{(0)}$ for the target house. In this section, we present two approaches for doing this, the first approach results in a *generic* model applicable to all houses and the second approach results in a model *specific* to the target house.

Generic model approach: A single model for all houses

In this approach, no distinction is made between source and target datasets. The approach consists of two steps, first the meta feature mapping is applied to the sensor feature matrices of all houses, so that the sensor data of all houses is represented in feature matrices of equal dimensionality. We then estimate the maximum likelihood parameters of the target house by applying the Expectation Maximization (EM) algorithm [8] to the datasets of all the houses (Fig. 6.1).

Step 1: Mapping to meta-feature space To perform the mapping we define a function $h(G, \vec{x}_t)$ which takes as input the mapping matrix G and an observation vector $\vec{x}_t$ and outputs a vector of binary indicators indicating whether for a particular meta feature at least one sensor that corresponds to that meta feature fired at timeslice t.

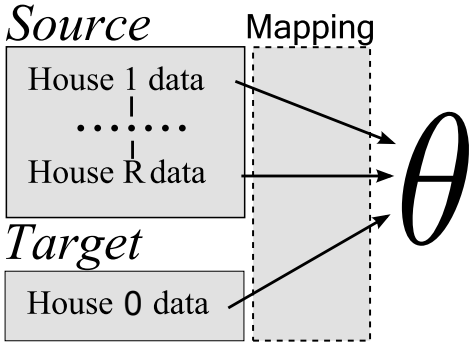


Fig. 6.1: Graphical representation of the transfer learning framework resulting in a generic model. Data from the source and target houses is mapped to the meta feature space and one set of parameters θ is learned using maximum likelihood.

Step 2: Estimating the Maximum Likelihood parameters The maximum likelihood parameters are learned by using the EM algorithm. Applying the EM algorithm consists of two steps, in the E-step the expectations are calculated using an initial set of parameters and in the M-step a new set of parameters is calculated from the expectations. These two steps are repeated and after each iteration the new set of parameters calculated in the M-step are used in the E-step to recalculate the expectations. The procedure is terminated when the parameter values converge.

The expectations over the state values given a set of training data can be efficiently calculated using the forward-backward algorithm [130]. We define $\xi_t(i,j) = p(y_t = i, y_{t+1} = j \mid \vec{c}_{1:T}, l_{1:T}, \theta)$, which is the probability outputted by the forward-backward algorithm given a set of parameters θ and a set of training data. The training data consists of the mapped sensor data $\vec{c}_{1:T}$ and the labels $l_{1:T}$, in which it is possible to include training data for which no annotation is available (i.e. having $l_t = 0$). The mapped sensor data is obtained by applying the mapping function $\vec{c}_{1:T} = h(G, \vec{x}_{1:t})$. Summing over $\xi_t(i,j)$ for all time indices t gives us a quantity which can be interpreted as the expected number of times that a transition from y_t to y_{t+1} is made. If we then take the resulting quantity and sum over the possible state values for y_{t+1}, we get the expected number of times that state y_t is visited. In the generic model approach to estimate the model parameters, we add the expectations of all houses to obtain a set of parameters that incorporates the data of all houses. The equations for maximizing the parameters are similar to the standard maximization equations for the HMM [130],

with the exception of an added sum term that sums over all the datasets $0 : R$.

$$\pi_i = \frac{\sum_{r=0}^{R} \sum_{j=1}^{Q} \xi_1^{(r)}(i,j)}{\sum_{r=0}^{R} \sum_{i=1}^{Q} \sum_{j=1}^{Q} \xi_1^{(r)}(i,j)} \tag{6.2}$$

$$a_{ij} = \frac{\sum_{r=0}^{R} \sum_{t=1}^{T-1} \xi_t^{(r)}(i,j)}{\sum_{r=0}^{R} \sum_{t=1}^{T-1} \sum_{j=1}^{Q} \xi_t^{(r)}(i,j)} \tag{6.3}$$

$$\mu_{in} = \frac{\sum_{r=0}^{R} \sum_{t=1}^{T-1} \sum_{j=1}^{Q} \delta((c_t^n)^{(r)}, 1) \xi_t^{(r)}(i,j)}{\sum_{r=0}^{R} \sum_{t=1}^{T-1} \sum_{j=1}^{Q} \xi_t^{(r)}(i,j)} \tag{6.4}$$

Since no distinction is made between the source and target datasets there is no difference between the model parameters of the target house and those of the source houses, that is $\theta^{(0)} = \theta^{(i)}$ for all $i \in 1 : R$.

Specific model approach: One model specifically for the target house

The distinction between the target and source datasets is that the target dataset contains specific information about the house we wish to perform activity recognition in, while the source datasets contain generic information about activity recognition. In this approach, we take that distinction into account and learn a set of parameters specifically for the target house.

Combining the generic information together with the specific information is done by using maximum a posteriori (MAP) parameter estimation. In MAP parameters estimation the likelihood function of the training data is combined with a prior distribution over the parameters. This can be formulated as:

$$\overbrace{p(\theta \mid \mathcal{D}, \Psi)}^{Posterior} \propto \overbrace{p(\mathcal{D} \mid \theta)}^{Likelihood} \overbrace{p(\theta \mid \Psi)}^{Prior} \tag{6.5}$$

where θ are the model parameters, $\mathcal{D}$ is the training data and Ψ are the parameters of the prior distribution, known as the hyperparameters. The prior distribution provides an initial estimate of the parameter values before incorporating any training data. For example, in the case of activity recognition, we could use our common sense and say that the prior probability of the toilet flush sensor firing when the activity toileting is performed is concentrated around 99%. Rather than using our common sense, we use the source datasets to estimate the prior distribution. The target dataset is used in the likelihood function and provides house specific information, while the learned prior distribution provides generic information. By jointly maximimizing these terms, we can estimate the MAP parameters of the target house. This entire procedure is broken down into three steps. Step 1 consists of learning the maximum likelihood parameters from

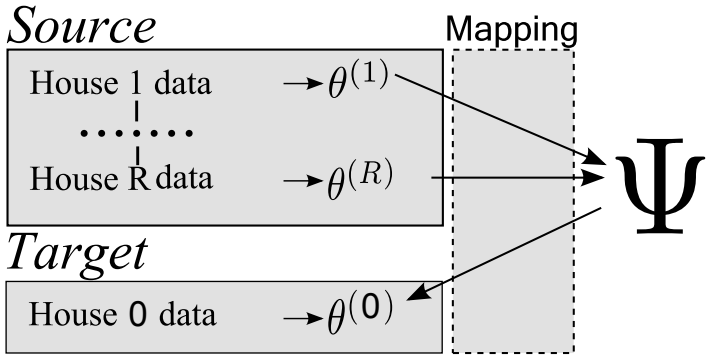


Fig. 6.2: Graphical representation of the transfer learning framework resulting in a specific model. For each source house i training data is used to learn model parameters $\theta^{(i)}$. All the source model parameters are used to learn the hyperparameters Ψ of the prior distributions. Which in turn is used to learn the target model parameters $\theta^{(0)}$ together with any available data from the target house.

the source datasets. In step 2, we estimate the hyperparameters Ψ of the prior distribution, by using the parameters of the source houses as examples. Finally, in step 3, we estimate the MAP model parameters for the target house (Fig. 6.2).

Step 1: Maximum Likelihood parameters of the source house Estimating the maximum likelihood parameters of the source houses is done separately for each house. Because the source house datasets are all fully annotated, we can simply use the equations introduced in Section 3.4.3 to learn the parameters. Note that no meta feature mapping has been applied to the source house datasets at this point of the procedure. Since the model parameters are learned for each house separately, there is no need yet to map to a common feature space.

Step 2: Estimating the prior distributions Before estimating the hyperparameters of the prior distributions, we first have to decide which probability distributions we will use for representing the prior probabilities of the model parameters. In Bayesian statistics, a prior is said to be conjugate if the resulting posterior is of the same functional form as the prior [9]. This property is generally preferred because it allows the use of sequential parameter estimation in which the posterior of a previous iteration is used as a prior for the new iteration.

Recall that our HMM consisted of a number of Bernoulli distributions and a number of multinomial distributions. The conjugate prior of a Bernoulli distribution is the beta distribution, while the conjugate prior of a multinomial distribution is the Dirichlet distribution. We therefore use the following prior distributions. The Dirichlet distribution is used for the initial state distribution parameters π

Factor	**Model Distribution**		**Prior Distribution**	
Name	Name	Parameters	Name	Hyperparameters
Initial State	Multinomial	π	Dirichlet	η
Transition	Multinomial	A	Dirichlet	ρ
Observation	Binomial	B	Beta	ω,υ

Tab. 6.2: Overview of the distributions used in the HMM, parameterized by the model parameters $\theta = \{\pi, A, B\}$. And the corresponding prior distribution, parameterized by the hyperparameters $\Psi = \{\eta, \rho, \omega, \upsilon\}$

and for the state transition distribution parameters a_i. The beta distribution is used for the observation parameters μ. This gives the following equations:

$$p(\pi \mid \eta) = \frac{\Gamma(\sum_{k=1}^{K} \eta_k)}{\Gamma(\eta_1) \dots \Gamma(\eta_K)} \prod_{k=1}^{K} \pi_k^{\eta_k - 1} \tag{6.6}$$

$$p(a_i \mid \rho) = \frac{\Gamma(\sum_{k=1}^{K} \rho_k))}{\Gamma(\rho_1) \dots \Gamma(\rho_K)} \prod_{k=1}^{K} a_{ik}^{\rho_k - 1} \tag{6.7}$$

$$p(\mu_{in} \mid \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu_{in}^{\alpha - 1} (1 - \mu_{in})^{\beta - 1} \tag{6.8}$$

where K is the number of elements in the vector. The hyperparameters α and β are in the sensor feature space and are therefore further parameterized as $\alpha_{in} = g_n \vec{v}_i$ and $\beta_{in} = g_n \vec{\omega}_i$, where g_n is the meta feature row vector for sensor n. The hyperparameters $\vec{v}_i$ and $\vec{\omega}_i$ are column vectors positioned in the meta feature space. An overview of all the distributions and their parameters can be found in Table 6.2.

There are no closed form solutions for estimating the maximum likelihood parameters of the Dirichlet and Beta distributions. Therefore, we use numerical methods for estimating the hyperparameters [105, 172]. It is straightforward to learn the hyperparameters η and ρ of the two Dirichlet distributions, used as a prior for the initial state distribution and the transition distribution. This is because the same number of activities is used in all of the houses, therefore the dimensionality of these model parameters is the same for all houses.

Estimating the hyperparameters of the observation distributions is more involved because of the different sensor feature spaces in each house. The numerical optimization methods for the Beta distributions give us the parameter values of α and β, which are in the sensor feature space. We can find the least square solution to the meta feature space parameters $\vec{v}$ and $\vec{\omega}$ by solving the system of equations defined by $\alpha_i = G\vec{v}_i$ and $\beta_i = G\vec{\omega}_i$, in which G is the mapping matrix and index i indicates the activity the parameters belong to. The values of α and β can be interpreted as the effective number of observations of the sensor outputting

a 1 or a 0, respectively [9]. To guarantee non-negative values, we add a 'bias' meta-feature with a large enough positive value. This bias will influence the parameter values, but because it is consistently applied to all parameter values, it generally does not affect the balance among the parameters that much.

Step 3: Maximum A Posteriori parameters of the target house To learn the MAP model parameters of the target house, we use the EM algorithm. In the E-step, any available unlabeled and/or labeled data from the target house is used to calculate the expectations using the forward-backward algorithm [130]. We define $\xi_t(i,j) = p(y_t = i, y_{t+1} = j \mid \vec{x}_{1:T}, l_{1:T}, \theta)$ which is the probability outputted by the forward-backward algorithm given a set of parameters θ and a set of training data consisting of the sensor feature $\vec{x}_{1:T}$ and the labels $l_{1:T}$. Note that the original sensor feature space is used for the observations. Just like in the generic model, summing over $\xi_t(i,j)$ gives us a quantity which can be interpreted as the expected number of times that a transition from y_t to y_{t+1} is made.

The equations in the M-step for calculating a new set of parameters have to incorporate the prior probability. These equations can be derived by adding the log probability of the prior distributions to the calculations of the expectations and taking the derivative of those terms with respect to each parameter of the HMM [54] . A Lagrange multiplier is used to guarantee the outcome satisfies the rules of probability. This results in the following equations:

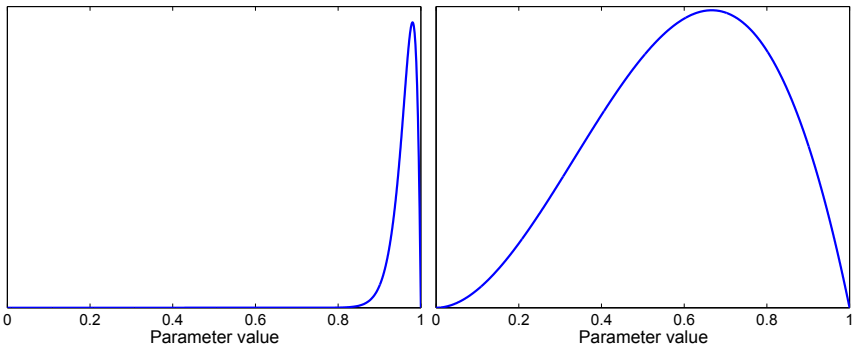
$$\pi_i = \frac{\sum_{j=1}^{Q} \xi_1(i,j) + (\eta_i - 1)}{\sum_{i=1}^{Q} \left\{ \sum_{j=1}^{Q} \xi_1(i,j) + (\eta_i - 1) \right\}} \tag{6.9}$$

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j) + (\rho_{ij} - 1)}{\sum_{t=1}^{T-1} \sum_{j=1}^{Q} \xi_t(i,j) + (\rho_{ij} - 1)} \tag{6.10}$$

$$\mu_{in} = \frac{(\alpha_{in} - 1) + \sum_{t=1}^{T-1} \sum_{j=1}^{Q} \delta(x_t^n, 1) \xi_t(i,j)}{(\alpha_{in} + \beta_{in} - 2) + \sum_{t=1}^{T-1} \sum_{j=1}^{Q} \xi_t(i,j)} \tag{6.11}$$

where $\alpha_{in} = g_n \vec{v}_i$ and $\beta_{in} = g_n \vec{\omega}_i$.

Compared to the generic model equations, we see here that instead of adding all the expectations of the houses together, we take the expectations of the target house and adjust them with the parameter values of the prior. In step 2 we noted that the parameter values of the beta distribution can be interpreted as the effective number of observations made. The parameter values of the Dirichlet distribution have a similar interpretation. For example, the ρ_{ij} can be interpreted as the effective number of transitions made from state i to state j. Because the parameters of the prior distributions are estimated from the source house datasets, we are basically performing a similar operation as summing over the



(a) Beta distribution with $\alpha = 50$ and $\beta = 2$. (b) Beta distribution with $\alpha = 3$ and $\beta = 2$.

Fig. 6.3: Examples of a peaked (left) and wide (right) beta distribution.

expectations of all houses as we did in the generic model case. The difference is that the parameters of the prior are estimated from a combination of the source house expectations, rather than a simple sum of expectations. How the source house expectations are combined depends on how much they are in agreement with each other. For example, if in every house the microwave is used during cooking, then the prior distribution will be strongly peaked over a probability value close to 1 for the microwave sensor firing while cooking. This will result in a large hyperparameter value (Fig. 6.3(a)) which has a strong influence on the expectations of the target house. On the other hand, if the data from the source houses are not in agreement, this will result in a wide prior distribution with low hyperparameter values (Fig. 6.3(b)) and therefore little influence on the expectations of the target house.

6.5 Experiments

Our transfer learning approach provides answers to the questions posed in the introduction section. The use of meta-features allows us to deal with differences in the layout of houses and our methods for learning the parameters of either a generic or specific model provides two ways for dealing with the difference in the behavior of inhabitants. In this section, we present experiments on real world datasets to test the quality of our proposed solutions. Furthermore, our experiments will determine whether transfer learning for activity recognition is possible and whether the inclusion of unlabeled data helps the recognition performance.

In the first experiment, we do not perform transfer learning, but compare the

	House A		House B		House C	
Activity	Num.	Time	Num.	Time	Num.	Time
Leave house	33	50.5%	24	59.6%	47	45.7%
Toileting	114	1.0%	27	0.4%	89	1.0%
Take shower	23	0.8%	11	0.6%	14	0.8%
Brush teeth	16	0.1%	13	0.2%	26	0.4%
Go to bed	24	33.2%	14	29.4%	19	29.2%
Prepare Breakfast	20	0.3%	9	0.5%	18	0.6%
Prepare Dinner	9	0.9%	6	0.5%	11	1.1%
Get drink	20	0.2%	8	0.1%	10	0.1%
Other	-	13.0%	-	8.7%	-	21.1%

Tab. 6.3: The activities that were annotated in the different houses. The 'Num.' column shows the number of times the activity occurs in the dataset. The 'Time' column shows the percentage of time the activity takes up in the dataset. All unannotated timeslices were collected in a single 'Other' activity.

performance of using the meta-feature space to using the original sensor feature representation. The second experiment compares the performance of the generic model and the specific model to a model learned without using transfer learning. Finally, in the third experiment, we compare the performance of performing transfer learning using both labeled and unlabeled data from the target dataset, to the performance of using only labeled data from the target house.

This section is organized as follows, we first give a description of the houses and the datasets recorded and provide details of our experimental setup. Then, we present the results and discuss the outcome.

6.5.1 Datasets used

Experiments were performed using the datasets of the three houses used in earlier chapters. A subset of activities were selected that were annotated in all three houses, an overview of these activities and how often they occur in each dataset can be found in Table 6.3. The activities for houses A and B were annotated using a wireless bluetooth headset, while activities in house C were annotated using a handwritten diary.

The layout of the houses differs strongly, for example, there are two toilets in house C, the toilet in house B is in the same room as the shower, while the toilet and shower in house A are in separate rooms. Furthermore, the inhabitants differ as well, house A was occupied by a 26 year old male, house B by a 28 year old male and house C by a 57 year old male.

6.5.2 Experimental Setup

In all experiments, the HMM was used as activity recognition model. The same meta feature mapping is used in all experiments. Choosing a proper mapping is not straightforward, since the optimal choice is not clear and a wrong decision can result in a feature space in which certain activities cannot be discriminated. In previous work on transfer learning for activity recognition, a comparison of mappings was made [64]. The mapping that combined sensor readings in a single feature based on their function (e.g. sensor used for heating) gave the best results. We use the same mapping which is described in detail in Appendix B.

Sensor data is discretized in timeslices of length $\Delta t = 60$ seconds. This time slice length is long enough to provide a discriminative sensor pattern and short enough to provide high resolution labeling results. After discretization, we have a total of 35486 timeslices for house A, 19968 timeslices for house B and 26236 timeslices for house C.

We split our data into a test and training set using a 'leave one day out' approach. In this approach, one full day of sensor readings is used for testing and the remaining days are, depending on the experiment, either partly or fully used for training. We cycle over all the days and report the average performance measure. In Section 3.7.2, we introduced the precision, recall and F-measure and showed that they are reliable measures for evaluating the performance of our model. The experimental results of Chapters 3 and 4 showed that the changepoint feature representation consistently gives a good performance, therefore, we use that feature representation for the experiments in this chapter. Significance testing between two cases A and B is done at a confidence interval of 95% using a one-tail student t-test and using matching paired days.

6.5.3 Experiment 1: Meta-feature vs. Sensor feature space

In this experiment, we do not use any form of transfer learning. Rather, we perform activity recognition using the same experimental setup as was done in the previous chapters. This means that for each experiment the dataset of only one house is used. Any available training data is used fully labeled according to the leave one day out approach. We wish to determine the impact on the recognition performance when the meta feature mapping is applied to the sensor data. Therefore, we compare the performance of the HMM using the original sensor feature space to the performance of the HMM using the meta feature space. Mapping the sensor data to the meta feature space corresponds to applying step 1 of our generic model approach. Model parameters are learned using maximum likelihood as described in Section 3.4.3.

House	Model	Feature Space	Precision	Recall	F-Measure
A	HMM	Meta	64 ± 16	70 ± 13	67 ± 14
	HMM	Sensor	71 ± 17	74 ± 14	72 ± 15
House	**Model**	**Feature Space**	**Precision**	**Recall**	**F-Measure**
B	HMM	Meta	53 ± 15	60 ± 13	56 ± 13
	HMM	Sensor	55 ± 18	68 ± 15	60 ± 17
House	**Model**	**Feature Space**	**Precision**	**Recall**	**F-Measure**
C	HMM	Meta	48 ± 10	58 ± 11	52 ± 10
	HMM	Sensor	50 ± 12	62 ± 13	55 ± 12

Tab. 6.4: Experiment 1: Precision, recall and F-measure for hidden Markov model (HMM) using the sensor feature space and using the meta feature space.

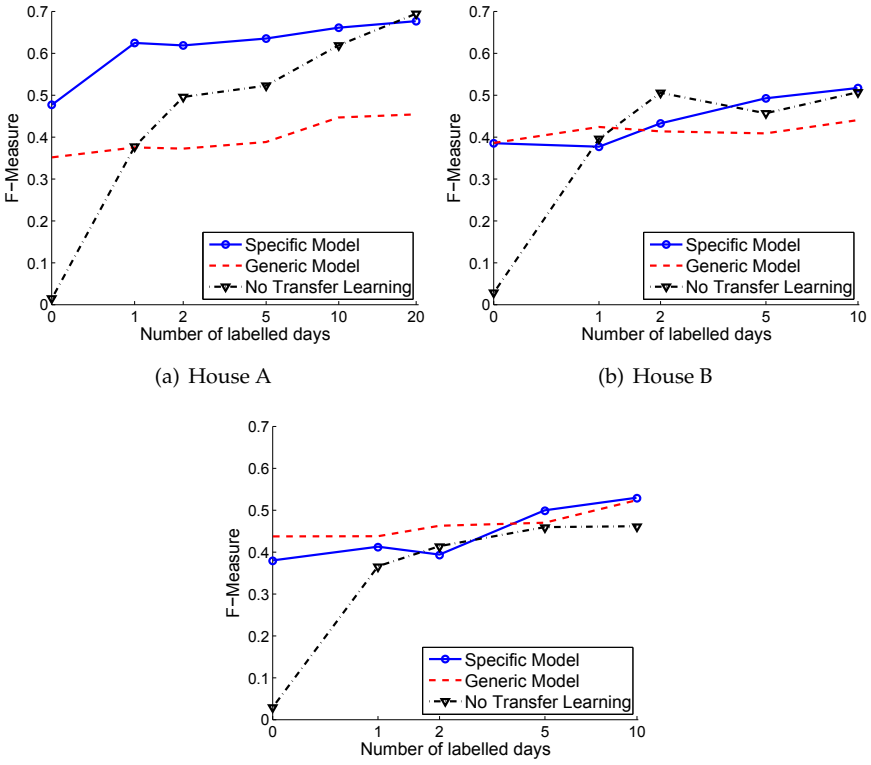
The results for the three houses are shown in Table 6.4. We see that in all three houses, recognition using the sensor feature space on average performs better than when using the meta feature space. To determine the significance of these results, we used a one-tail student t-test with matching paired days. The increase in F-measure performance is significant only for House A at a confidence interval of 95%.

6.5.4 Experiment 2: Generic model vs. Specific model

This experiment compares the performance of the generic model approach to the performance of the specific model approach. One house is used as the target house, while the remaining two houses are used as source houses. We compare the performance of the generic and the specific transfer learning approaches to the performance of the model from the previous experiment in which no transfer learning was done and the original sensor feature space was used. This way we are able to see which transfer learning method works best and what the difference in performance is compared to not doing transfer learning.

Figure 6.4 shows the results for all three target houses. The X-axis show the number of days of labeled data that was used, any remaining unlabeled data was also included during learning. We first compare the two transfer learning approaches to the 'no transfer learning' approach. We see that both transfer learning approaches significantly outperform the 'no transfer learning' approach in all three houses when 0 days of labeled training data are used (i.e. only unlabeled data is used). This is because the 'no transfer learning' approach has no labeled data to learn its parameters, while the two transfer learning approaches can use the labeled data of the source houses.

When more days of labeled training data are used there are no significant differences in performance between any of the methods in the case of Houses B and C. In House A, we see that the specific model approach on average performance



(a) House A

(b) House B

(c) House C

Fig. 6.4: Results of experiment 2, the x-axis are in log-scale and show the number of labeled days of data that were used for training. Any remaining unlabeled data was also included during learning.

better than the 'no transfer learning' approach, but that this difference decreases as more labeled data is included. This shows how the use of a prior distribution can help in learning the model parameters and that its effect decreases as more labeled data from the target house is used.

The generic approach performs worse than the 'no transfer learning' in house A when 2 or more days of labeled training data are used. This phenomenon is called negative transfer [15] which means that transfer learning has a negative effect on the learning process. This is because it is not clearly defined which parts of the data in the source houses are useful for the target house and which or not. Including training data from the source houses during learning can therefore sometimes pull the choice of parameters away from the optimal solution. It is interesting to see that the same collection of data can result in a positive transfer

for the specific model and a negative transfer for the generic model. We will speculate about this occurrence in the discussion section.

Finally, in House A, we see that the specific model on average performs better than the generic model. The specific model is able to learn model parameters that take into account the specific behavior for that house, because its learning method properly balances the influence of the source datasets. The generic model on the other hand only gains a slight performance increase from the extra data, because the data from each house has an equal contribution in the estimation of the parameters. This makes the weight of the labeled data from the target house less than when a prior distribution is used.

6.5.5 Experiment 3: Labeled vs. Unlabeled data

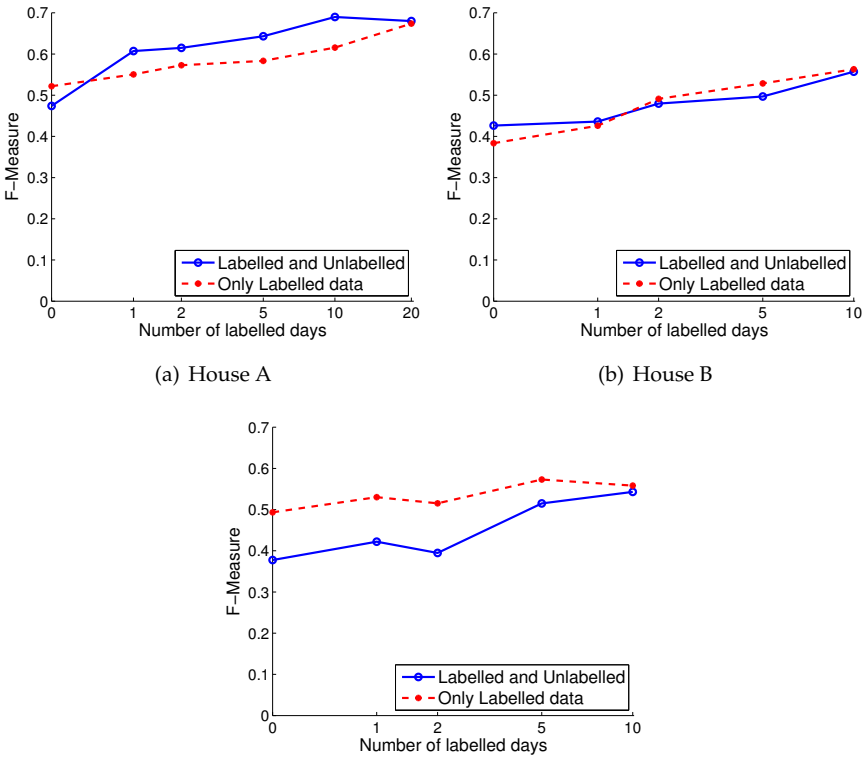
The use of generative models allows us to include unlabeled data during the learning process. In this experiment, we compare performance of using both unlabeled and labeled data to using only labeled data. In both cases, we use the specific model approach to learn the model parameters. The results for the various houses can be found in Figure 6.5. We see that adding unlabeled data results in a slight increases in performance for House A, gives more or less equal performance for House B and decreases performance for House C, compared to the labeled only approach. This tells us that including unlabeled data can have a rather unpredictable effect on the performance of our model. Further experiments on other datasets might provide better insight into the effects of including unlabeled data with respect to the recognition performance. However, based on these experiments it seems best to rely only on labeled data.

6.6 Discussion

Meta features

Our experiments show the recognition performance in the original sensor feature space is better than in the meta feature space. This is not unexpected as the sensor feature space is specific to the house we perform activity recognition in, while the meta feature space is a common feature space that can be used by all houses. Nonetheless, since we need a common feature space to be able to perform transfer learning, the use of meta features serves as a valid solution for dealing with differences in the layout of houses.

The set of meta features used in our experiments were manually defined and therefore it is not certain that the optimal mapping was used. An alternative is to learn the mapping automatically from data [25, 183]. However, because we are



(a) House A

(b) House B

(c) House C

Fig. 6.5: Results of experiment 3, the x-axis are in log-scale and show the number of labeled days of data that were used for training. In the case where unlabeled data is also included, the remaining days of unlabeled data are added during training.

working with time series data, the computational complexity can quickly become an issue in developing a feasible approach.

Generic or specific model

The results for all three houses show that both the generic and specific approach to transfer learning work when no labeled data from the target house are used. This is a very important result since it shows that transfer learning for activity recognition is possible and therefore proves that there must be similarities with respect to the way activities are performed between the houses of the experiments. Furthermore, the result is promising as it indicates that by using transfer learning we are able to install activity recognition in any house without the need for any

annotated data from that house.

Using transfer learning when labeled data for the target house is available did not result in a consistent increase in performance. An increase in performance only occurred when the specific model was used in combination with house A as the target house. This raises the question what the difference is between house A and the other houses. Unfortunately these differences are quite a few to be able to make any solid conclusion based on these experiments. There are differences in terms of the number of sensors used and where they were installed, the layout of the houses, the behavior of the inhabitants and the amount of data that was recorded in each house.

In house A we also saw that the use of a generic model resulted in negative transfer, while the use of the specific model gave a positive outcome. The explanation for this lies in the way both models incorporate the source house data. More specifically, the differences lies in the re-estimation equations of the parameters, used in the M-step of the EM algorithm. In the generic model, the model parameters are calculated by summing over the expectations from all houses, while in the specific model the parameters are calculated by adjusting the expectations of the target house with the parameter values of the prior (for further details on these effects see the end of step 3 of the specific model description, below Equation 6.11). For the generic model this means that the contribution of the expectations of the target house depends on how many source houses were used and how large those source house datasets were. This is not a very sensible approach because a large collection of annotated data from the target house can be completely outweighed by a large number of source house datasets. On the other hand, in the specific model the contribution of the prior depends on to what extend the source house datasets are in agreement with each other. If certain patterns of behavior consistently occur in all source datasets, the prior will be very strongly peaked for parameter values related to such patterns. Which will result in a strong influence on the target house expectations. Such an approach is much more sensible, since the influence of the prior depends on evidence found in the data. This explains why the specific model performs so much better than the generic model in House A. Why the same does not happen for the other houses is not clear.

Future work

In terms of future work, it would be interesting to apply transfer learning to several other models. For example, the use of hierarchical models might be better fit for transfer learning because the different levels of the hierarchy allow a better abstraction between houses. Comparing the performance gain due to transfer learning between several models can provide interesting insights on how to accurately model data. It would also be interesting to apply our transfer learning

approach to other sensing modalities such as cameras or wearables. Creating a proper mapping for those modalities will be challenging. Finally, it would be interesting to perform transfer learning across different sensing modalities. For example, using source houses in which camera's and wearables are used to perform activity recognition and a target house in which a wireless sensor network is used.

6.7 Conclusion

We have addressed the problem of learning model parameters when little or no labeled data is available for the house activity recognition is to be performed in. By using activity recognition datasets of other houses, we can transfer the knowledge about activity recognition from one house to another. Differences in the layout of houses results in different sensor feature spaces among datasets, we therefore presented a meta feature representation which allows us to map all sensor data to a common feature space. We presented two methods for learning the model parameters, one approach results in a generic model for activity recognition and the other approach results in a model specifically for the house activity recognition is performed in.

Our experiments on three real world datasets show that it is possible to use transfer learning for activity recognition. The use of a meta feature mapping results in a performance decrease, but allows us to perform transfer learning. When using no labeled training data the use of both the generic and specific model results in a significant increase in performance when compared to not using transfer learning. This is a very important result since it shows that transfer learning for activity recognition is possible and therefore proves that there must be similarities with respect to the way activities are performed between the houses of the experiments. Furthermore, the result is promising as it indicates that by using transfer learning, we are able to install activity recognition in any house without the need for any annotated data from that house. When using one or more days of labeled training data the specific model is able to perform on average better than the no transfer learning case in one of the three houses. The generic model performs on average equally well, or worse than the no transfer learning case in all three houses. Using the specific model therefore seems to be the preferred choice, although further experiments are needed to determine why the specific model is not able to obtain a performance gain in all three houses.

7

Conclusions

The automatic recognition of activities from sensor data allows important applications in healthcare for dealing with the societal problems of an aging population. In this final chapter, we present our concluding remarks and highlight the contributions of this thesis to this area of research. We end this chapter with a discussion on directions for future research.

7.1 Conclusions and contributions

In the introduction of this thesis, we presented a number of research questions that are answered in this thesis. The first question asked which temporal probabilistic model is able to accurately recognize activities from sensor network data. In Chapters 3, 4 and 5, we provided experimental results that allow us to answer this question.

The temporal probabilistic models used in this thesis use a time series representation. This requires us to discretize the sensor data using a constant timeslice length. In Chapter 3, experiments on three real world datasets revealed that the recognition performance is not very strongly affected by the length of the time interval used for discretization. A time interval of 60 seconds provides the proper balance between an accurate representation of the data and a reasonable amount of data needed for this representation.

Transforming sensor data to a different feature representation can sometimes help the recognition performance of a model significantly. We presented three feature representations the raw representation, changepoint representation and the last-

fired sensor representation. Experiments showed that the change and last-fired feature representations perform significantly better than the raw representation. Combinations of representations sometimes resulted in a slight increase in performance, but never significantly outperforms the separate use of the change and last representations.

We presented two probabilistic models for activity recognition in Chapter 3, the hidden Markov model (HMM) and the conditional random field (CRF). Comparing the model performances showed that the CRF generally performs better than the HMM in terms of the accuracy measure. However, in this measure, activities that occur frequently have a larger weight. The CRF favors the recognition of frequent activities, but can completely ignore infrequent activities as a result. We consider the recognition of each activity equally important and therefore rely on the multi-class precision and recall and the F-measure for comparing performances. Comparing these models in terms of F-measure does not show any significant differences in performance between the two models. The performance of these models was used as a recognition baseline in the other chapters.

In Chapter 4, we presented semi-Markov models which explicitly model the duration of an activity. Experiments were run to determine which probability distribution is best suited for modeling the duration of an activity. Three unimodal distributions, a multivariate distribution and a histogram approach with various numbers of bins were tested. The results showed that the Gaussian distribution performed best in the majority of the datasets and was therefore the distribution that was used in the other experiments. We experimented with two semi Markov models, the hidden semi-Markov model (HSMM) and the semi-Markov conditional random field (SMCRF). The HMM and HSMM are both generative models, while the CRF and SMCRF are discriminative models. Experiments showed that accurate duration modeling is important in generative models, but less important in discriminative models. This is because discriminative models are better able to deal with violations of the modeling assumptions. Overall the HSMM and the SMCRF give similar performance on most datasets. Neither model manages to consistently obtain a significantly better performance than the other.

A hierarchical structure for activities consisting of action primitives, actions and activities was presented in the introduction of this thesis. This structure was used in Chapter 5 to create a hierarchical hidden Markov model (HHMM) for activity recognition. Model parameters were estimated by using annotated data for the activities, but no annotation for the actions. As a result, clusters of actions were automatically allocated in the data, in which we are free to choose the number of action clusters to use. We experimented with two observation models for our HHMM. One model in which a separate set of action clusters is used for each activity, and another model in which action clusters are shared among activities. Our experiments showed that the observation model that uses a separate set

of action clusters for each activity works best. Using that observation model, the HHMM significantly outperforms the HMM and the HSMM in two of the three real world datasets, with an increase of 7 percent points in F-measure performance for both datasets.

The experiments from these chapters indicate that the answer to the first question of this thesis is to use a hierarchical hidden Markov model that uses an observation model which uses a separate set of action clusters for each activity. Data should be discretized using the timeslice interval of 60 seconds and should be transformed to the changepoint or last-fired feature representation.

The second question of this thesis addresses the problem of reusing a model for activity recognition in multiple homes. In Chapter 6, we present transfer learning which allows us to learn model parameters despite differences between various homes. This requires the use of a meta feature representation to map sensor data from datasets of different houses to a common feature space. We experimented with a model that learn a generic set of parameters and a model that learns a set of parameters specific to the house activity recognition is performed in. Experiments show that both models allow us to successfully transfer knowledge to a house for which no annotated data is available. When annotated data is available the specific model approach is able to perform on average better than the no transfer learning case in one of the three houses.

Our final question addresses the issue of evaluating the performance of a pattern recognition method to ensure an accurate performance in a real world setting. The experiments in this thesis were performed on three real world datasets. We presented a sensor and annotation system that allows us to create datasets in multiple homes and annotate them accurately. To evaluate the performance of activity recognition models, we suggest the use of the multi-class precision and recall and the F-measure. The software for the models presented in this thesis, together with the datasets on which the experiments were ran are available from: `http://sites.google.com/site/tim0306/`. The entire package can serve as a benchmark for future work on activity recognition.

7.2 Future research

In this section, we present directions for future research, we discuss topics with respect to sensors, the setup activity recognition is performed in, learning methods for activity recognition and the output an activity recognition system should give.

In this work, we used sensor networks to observe the behavior of inhabitants. The sensors used were installed in locations that seemed intuitive to the research team. Certain problems encountered in the classification of activities might easily

be resolved by installing additional sensors. However, because these activity recognition systems will eventually be used on a large scale, it is important to keep the costs of the system to a minimum. It is therefore important to understand the impact on the recognition performance of both the number of sensors used and the location the sensors are installed. Such understanding will allow us to create an activity recognition system using the minimum number of sensors needed.

All the models and experiments performed in this thesis assumed that a household consists of a single person. For a realistic application in a real world setting, an activity recognition system should be able to deal with people visiting a household and with households in which multiple people live. The challenge in performing multi-person activity recognition is that people can perform activities separately, in which sensors from two different locations in the houses are triggered, and activities can be performed jointly, in which a single set of sensors is triggered.

Our models for activity recognition require annotated data to estimate the model parameters. The transfer learning approach we suggested will help in installing activity recognition systems in new houses, but additional annotated data would surely help the recognition performance of the model. Instead of annotating activities for an entire day, we could select a few timeslices for which it would be most informative for the classifier to know which activity was performed. This can be done using active learning [141] in which an algorithm calculates for each timeslice the potential gain in information for knowing the label of that timeslice.

The parameters obtained through learning accurately represent the behavior of an inhabitant at a certain point in time. However, because the behavior of people changes over time, the parameters learned at one point in time may not accurately represent the behavior at a later point in time. In a technique called lifelong learning, parameters are continuously updated, rather than estimated only once during a learning phase [163]. Lifelong learning can be considered as a form of transfer learning, because the recognition of the behavior of a person at two separate moments in time corresponds to two different, but related, classification tasks. The challenge in this approach is that parameters are continuously updated. One hypothesis that can be tested for this approach is whether changes in behavior occur gradually or suddenly. Ideally lifelong learning methods update the model parameters using only sensor data, for which no annotation is available. Parameters could be gradually updated as small changes in the behavior become visible in the sensor data. Such unsupervised methods are not uncommon in the machine learning [9], but the effects of continuously updating parameters over a long period of time have to be studied.

Finally, currently our models for activity recognition simply output a list of activities that were performed during the day. It is unlikely that care givers would like to see long lists of activities for every patient they are caring for.

Instead they might be interested in a summary or in graphs showing the average duration of activities and the frequency with which activities are performed. It is important to gain a better understanding of which information caregivers need in performing their duties. In practice such multi-disciplinary research is challenging to perform and field trials will need to be performed to obtain a better understanding of these principles.

Appendix A

Inference Algorithms

In this appendix, we describe the Viterbi algorithms for the different models discussed. The Viterbi algorithm finds the sequence of states with the highest probability, given a sequence of observations. In each section, we provide a generalized version of the Viterbi algorithm that works for both generative and discriminative models.

A.1 Viterbi Algorithm for Markov Models

The generalized version of the Viterbi algorithm described in this section uses a factor representation. Two factors are used, an observation factor O and a transition factor $\mathcal{T}$. When using the hidden Markov model (HMM), the observation factor is defined as $O(i, \vec{x}_t) = \prod_{n=1}^{N} \mu_{in}^{x_n}(1 - \mu_{in})^{(1-x_n)}$ and the transition factor as $\mathcal{T}(j, i) = a_{ij}$. We define a special case for $\mathcal{T}$ representing the initial state distribution $\mathcal{T}(i, 0) = \pi_i$. In the case of the conditional random field (CRF), the observation factor is defined as $O(i, \vec{x}_t) = \exp\{\sum_k \lambda_k f_k(y_t = i, \vec{x}_t)\}$ and the transition factor as $\mathcal{T}(j, i) = \exp\{\sum_k \lambda_k f_k(y_t = j, y_{t-1} = i)\}$.

To find the most probable sequence of states, we define the quantity $\mathcal{P}_t(i)$, which represents the best score (be it probability or potential) of ending in state $y_t = i$, taking the first t observations into account. A bookkeeping variable $\zeta_t(j)$ is used to store the optimal state to transition from at each timeslice. The Viterbi algorithm can now be stated as follows:

1. Initialization: Set the first timeslice.

$$\begin{aligned} \mathcal{P}1(i) &= \mathcal{O}(i, \vec{x}_1)\mathcal{T}(i, 0) \\ \zeta_1(i) &= 0 \end{aligned}$$

2. Recursion: Iterate over timeslices $2 \leq t \leq T$.

$$\begin{aligned} \mathcal{P}_t(j) &= \max_{1 \leq i \leq Q} [\mathcal{P}_{t-1}(i)\mathcal{T}(j, i)]\, \mathcal{O}(j, \vec{x}_t) \\ \zeta_t(j) &= \operatorname*{argmax}_{1 \leq i \leq Q} [\mathcal{P}_{t-1}(i)\mathcal{T}(j, i)] \end{aligned}$$

3. Termination: Determine which state has the highest probability in the final timeslice.

$$\begin{aligned} P^* &= \max_{1 \leq i \leq Q} [\mathcal{P}_T(i)] \\ y_T^* &= \operatorname*{argmax}_{1 \leq i \leq Q} [\mathcal{P}_T(i)] \end{aligned}$$

4. Sequence backtracking: Look up the previous state from T to 1.

$$y_t^* = \zeta_{t+1}(y_{t+1}^*)$$

A.2 Viterbi Algorithm for Semi-Markov Models

Semi-Markov models consist of two hidden variables at each timeslice, the hidden state and the duration of that hidden state. For a novel sequence of observations, the values of both hidden variables are unknown and need to be inferred using the Viterbi algorithm. We use a factor representation to present a generalized version of the Viterbi algorithm, applicable to both the hidden semi-Markov model (HSMM) and semi-Markov conditional random field (SMCRF). We use an observation factor $\mathcal{O}$, a transition factor $\mathcal{T}$ and a duration factor $\mathcal{D}$. When using the HSMM the observation factor is defined as $\mathcal{O}(i, \vec{x}_t) = \prod_{n=1}^{N} \mu_{in}^{x_n}(1 - \mu_{in})^{(1-x_n)}$, the duration factor as $\mathcal{D}(l, i) = p_i(l)$ and the transition factor as $\mathcal{T}(j, i) = a_{ij}$. We define a special case for $\mathcal{T}$ representing the initial state distribution $\mathcal{T}(i, 0) = \pi_i$. In the case of the SMCRF, the observation factor is defined as $\mathcal{O}(i, \vec{x}_t) = \exp\{\sum_k \lambda_k f_k(y_t = i, \vec{x}_t)\}$, the duration factor as $\mathcal{D}(l, i) = f_k(y_t = i, d_t = l)$ and the transition factor as $\mathcal{T}(j, i) = \exp\{\sum_k \lambda_k f_k(y_t = j, y_{t-1} = i)\}$.

To find the most probable sequence of states, we define two quantities, $\mathcal{P}_t(i)$ represents the best score (either probability or potential) of ending in state $y_t = i$, taking the first t observations into account. The quantity $\mathcal{P}_t^{\mathcal{D}}(d, i)$ represents the

best score of ending in state $y_t = i$, with a state duration of $d+1$ timeslices, taking the first t observations into account. Three bookkeeping variables are needed, $\zeta_t(j)$ is used to store the optimal state to transition from at each timeslice. The variable $\zeta_t^{\mathcal{D}}(d, j)$ stores the optimal state to transition from, considering a state duration of $d+1$ timeslices. Finally, $v_t(i)$ stores the optimal duration for state i at time t. The maximum duration that a state can take is defined beforehand in a variable D. We use the variable l to represent the duration of a state. The Viterbi algorithm can now be stated as follows:

1. Initialization: Set the probabilities of the initial states, for lengths $l = 1 : D$, and states $i = 1 : Q$.

$$\begin{aligned}
\mathcal{P}_1(i) &= \mathcal{T}(i,0)\mathcal{D}(1,i)\mathcal{O}(i,\vec{x}_1) \\
\mathcal{P}_l^{\mathcal{D}}(l,i) &= \mathcal{T}(i,0)\mathcal{D}(l,i)\prod_{t=1}^{l}\mathcal{O}(i,\vec{x}_t) \\
\zeta_l^{\mathcal{D}}(l,i) &= 0
\end{aligned}$$

2. Recursion: At each timeslice, first iterate over all possible durations. For timeslices $t = 2 : T$, lengths $l = 1 : D$ and states $j = 1 : Q$.

$$\begin{aligned}
\mathcal{P}_t^{\mathcal{D}}(l,j) &= \max_{1\le i\le Q}\left[\mathcal{P}_{t-l}(i)\mathcal{T}(j,i)\right]\mathcal{D}(l,j)\prod_{m=t-l+1}^{t}\mathcal{O}(j,\vec{x}_m) \\
\zeta_t^{\mathcal{D}}(l,j) &= \operatorname*{argmax}_{1\le i\le Q}\left[\mathcal{P}_{t-l}(i)\mathcal{T}(j,i)\right]
\end{aligned}$$

   Then determine which duration gives the highest probability.

$$\begin{aligned}
\mathcal{P}_t(i) &= \max_{1\le l\le D}\mathcal{P}_t^{\mathcal{D}}(l,i) \\
v_t(i) &= \operatorname*{argmax}_{1\le l\le D}\mathcal{P}_t^{\mathcal{D}}(l,i) \\
\zeta_t(i) &= \zeta_t^{\mathcal{D}}(v_t(i),i)
\end{aligned}$$

3. Termination: Determine which state has the highest probability in the final timeslice.

$$\begin{aligned}
P^* &= \max_{1\le i\le Q}\left[\mathcal{P}_T(i)\right] \\
y &= \operatorname*{argmax}_{1\le i\le Q}\left[\mathcal{P}_T(i)\right] \\
l &= v_T(y) \\
t &= T
\end{aligned}$$

4. Sequence backtracking: Backtrack from there, looking up the duration and previous state, until $t = 0$.

$$\begin{array}{lcl} y^*_{t-l+1:t} & = & y \\ y & = & \zeta_t(y) \\ t & = & t - l \\ l & = & v_t(y) \end{array}$$

Appendix B

Meta Features per House

Sensors	Bathroom	Bathroom Entrance	Kitchen Heating	Kitchen Storage	Kitchen	Outside	Bedroom	Bedroom Entrance	Toilet
Front Door	0	0	0	0	0	1	0	0	0
Microwave	0	0	1	0	0	0	0	0	0
Refrigerator	0	0	0	1	0	0	0	0	0
Freezer	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (cups)	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (plates)	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (pans)	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (various)	0	0	0	1	0	0	0	0	0
Dishwasher	0	0	0	0	1	0	0	0	0
Washing Machine	0	0	0	0	1	0	0	0	0
Bathroom door	0	1	0	0	0	0	0	0	0
Toilet door	0	0	0	0	0	0	0	0	1
Toilet Flush Sensor	0	0	0	0	0	0	0	0	1
Bedroom Door	0	0	0	0	0	0	0	1	0

Tab. B.1: House A meta feature matrix, indicating which meta features (columns) correspond to which sensors (rows).

Sensors	Bathroom	Bathroom Entrance	Kitchen Heating	Kitchen Storage	Kitchen	Outside	Bedroom	Bedroom Entrance	Toilet
Front Door	0	0	0	0	0	1	0	0	0
Balcony Door	0	0	0	0	0	0	0	0	0
Window	0	0	0	0	0	0	0	0	0
Microwave	0	0	1	0	0	0	0	0	0
Toaster	0	0	1	0	0	0	0	0	0
Refrigerator	0	0	0	1	0	0	0	0	0
Kitchen Sink	1	0	0	0	0	0	0	0	0
Kitchen Cupboard (various)	0	0	0	1	0	0	0	0	0
Kitchen Cupboard (plates)	0	0	0	0	1	0	0	0	0
Kitchen Drawer (cutlary)	0	0	0	0	1	0	0	0	0
Kitchen Lid (stove)	0	0	1	0	0	0	0	0	0
Kitchen PIR	0	0	0	0	1	0	0	0	0
Bathroom Door	0	1	0	0	0	0	0	0	0
Bathroom PIR	1	0	0	0	0	0	0	0	0
Toilet Flush Sensor	0	0	0	0	0	0	0	0	1
Bedroom Door	0	0	0	0	0	0	0	1	0
Bed Pressure Mat	0	0	0	0	0	0	1	0	0
Bed Pressure Mat	0	0	0	0	0	0	1	0	0
Bedroom PIR	0	0	0	0	0	0	1	0	0
Bedroom Dresser	0	0	0	0	0	0	1	0	0
Chair Pressure Mat	0	0	0	0	0	0	0	0	0
Chair Pressure Mat	0	0	0	0	0	0	0	0	0

Tab. B.2: House B meta feature matrix, indicating which meta features (columns) correspond to which sensors (rows).

Sensors	Bathroom	Bathroom Entrance	Kitchen Heating	Kitchen Storage	Kitchen	Outside	Bedroom	Bedroom Entrance	Toilet
Front Door	0	0	0	0	0	1	0	0	0
Magnetron	0	0	1	0	0	0	0	0	0
Refrigerator	0	0	0	1	0	0	0	0	0
Freezer	0	0	0	1	0	0	0	0	0
Kitchen Drawer (various)	0	0	0	0	1	0	0	0	0
Kitchen Drawer (cutlary)	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (pans)	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (storage)	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (plates)	0	0	0	0	1	0	0	0	0
Kitchen Cupboard (cups)	0	0	0	0	1	0	0	0	0
Bathroom Door	0	1	0	0	0	0	0	0	0
Bathroom Sink	1	0	0	0	0	0	0	0	0
Bathroom PIR	1	0	0	0	0	0	0	0	0
Toilet Flush (upstairs)	0	0	0	0	0	0	0	0	1
Toilet Flush (downstairs)	0	0	0	0	0	0	0	0	1
Toilet Door (downstairs)	0	0	0	0	0	0	0	0	1
Bedroom Door	0	0	0	0	0	0	0	1	0
Bedroom Dresser	0	0	0	0	0	0	1	0	0
Bed Pressure Mat	0	0	0	0	0	0	1	0	0
Bed Pressure Mat	0	0	0	0	0	0	1	0	0
Couch Pressure Mat	0	0	0	0	0	0	0	0	0

Tab. B.3: House C meta feature matrix, indicating which meta features (columns) correspond to which sensors (rows).

Summary

As people age their need for care increases, but with an increasing population of elderly worldwide, it becomes important that such care is provided in an efficient way. Automatic health monitoring provides an inexpensive way for obtaining information needed to give efficient and accurate care to these elderly. A common method in healthcare for assessing the cognitive and physical wellbeing of elderly are activities of daily living (ADL), a collection of activities that are performed on a daily basis such as cooking, toileting and showering. Recent developments in sensing technology make it possible to easily equip existing homes with sensors, therefore allowing a continuous observation system. However, automatically interpreting this sensor data to recognize ADLs is an unsolved problem.

We use wireless sensor networks to observe actions performed by inhabitants in their homes. Sensors used are contact switches to measure whether doors and cupboards are open or closed; pressure mats to measure sitting on a couch or lying in bed; mercury contacts for movement of objects such as drawers; passive infrared sensors to detect motion in a specific area and float sensors to measure the toilet being flushed. Recognizing activities from such sensor data is challenging because the data is ambiguous, unsegmented, noisy and because activities can be performed in a large number of ways.

In this thesis, we answer some fundamental pattern recognition questions with respect to activity recognition. Issues with respect to discretization and feature representation are discussed and solutions to these issues are proposed and empirically supported. We compare the recognition performance of two of the most basic temporal probabilistic models, the hidden Markov model and the conditional random field, both providing a baseline for the recognition performance. The two models used present an important dichotomy in the field of temporal probabilistic models, namely the choice between generative and discriminative models. The use of the Markov assumption in these models has important limitations with respect to modeling long term dependencies. Semi-Markov models relax the Markov assumption and provide a solution for modeling long term dependencies, but require computationally expensive inference algorithms. We

compare conventional Markov models with their semi-Markov counterparts and discuss their impact on the computation time and recognition performance. To model the internal structure of activities more accurately we present a hierarchical model for activity recognition. The use of a hierarchy allows abstraction of the data on multiple levels. We compare two approaches to modeling activities using hierarchical models and compare the recognition performance of these models to the performance of the Markov and semi-Markov models. Finally, we introduce a method for applying activity recognition models in multiple homes without the need for labeled training data from each home. Models used in activity recognition require labeled data to learn the model parameters. But a model trained for one home can not automatically be used in another home, due to differences in the layout of the homes and the behavior of the inhabitants. We present a transfer learning method that allows us to use labeled data from other homes to learn the model parameters for a new home. This makes it possible to apply the models discussed on a large scale, therefore providing a broadly applicable solution to efficient care giving of elderly.

Samenvatting

Naarmate mensen ouder worden krijgen zij een grotere behoefte aan zorg. Vanwege de vergrijzing neemt het aantal ouderen in de samenleving in rap tempo toe en daarom wordt het steeds belangrijker dat zorg efficiënt wordt toegepast. Door automatisch de gezondheid van mensen te monitoren, is het mogelijk op een goedkope manier informatie te verzamelen die nodig is om efficiënt zorg aan ouderen te leveren. Binnen de gezondheidszorg wordt het uitvoeren van dagelijkse activiteiten gebruikt als maatstaf voor de cognitieve en fysieke toestand van ouderen. Activiteiten zoals koken, toiletteren en douchen zijn typische voorbeelden van dergelijke activiteiten. Vanwege recente ontwikkelingen op het gebied van sensor technologie is het nu mogelijk om eenvoudig een bestaand huis met diverse sensoren uit te rusten. Dit zorgt ervoor dat er continu in het huis geobserveerd kan worden welke sensoren geactiveerd zijn. Echter, om vanuit deze sensor-informatie te achterhalen welke activiteiten er in het huis worden uitgevoerd is een openstaand probleem.

In dit werk gebruiken wij draadloze sensor netwerken om de handelingen van bewoners in hun huizen te observeren. Wij gebruiken hiervoor contact-schakelaars die meten of een deur of kastje open of dicht is; drukmatten om te meten of iemand op een bank zit of in bed ligt; kwik-schakelaars die beweging van ladekastjes kunnen waarnemen; infrarood-sensoren die beweging in een bepaalde ruimte kunnen meten en niveau-schakelaars die meten of het toilet wordt doorgespoeld. Het automatisch herkennen van activiteiten aan de hand van dergelijke sensor data is uitdagend omdat de betreffende data ambigu en ongesegmenteerd zijn, ruis kunnen bevatten en omdat activiteiten op veel verschillende manieren uitgevoerd kunnen worden.

In dit proefschrift beantwoorden we een aantal fundamentele vragen op het gebied van patroonherkenning toegepast op het herkennen van activiteiten. Vraagstukken op het gebied van discretisatie en data representatie komen aan bod en oplossingen worden voorgesteld en empirisch getoetst. Twee veel gebruikte temporele probabilistische modellen worden besproken, het 'Hidden Markov Model' en het 'Conditional Random Field' en hun nauwkeurigheid in

het herkennen van activiteiten wordt vergeleken. Deze prestaties vormen een basis voor vergelijkingen met andere modellen. Beide modellen gaan uit van de Markov assumptie, een aanname die belangrijke beperkingen met betrekking tot het modelleren van lange termijn afhankelijkheden als gevolg heeft. Semi-Markov modellen verlichten deze aanname enigszins en bieden mogelijkheden om lange termijn afhankelijkheden te modelleren. Een gevolg hiervan is echter dat er computationeel intensievere algoritmen gebruikt moeten worden. Wij vergelijken de standaard Markov modellen met hun semi-Markov varianten en bespreken verschillen op het gebied van de benodigde rekenkracht en op het gebied van de nauwkeurigheid om activiteiten te herkennen.

Om de interne structuur van activiteiten nauwkeuriger te modelleren gebruiken we hiërarchische modellen. Het gebruik van een hiërarchie maakt het mogelijk om data op diverse abstractie niveaus te representeren. Er wordt een vergelijking tussen twee hiërarchische aanpakken gegeven en de prestaties van deze modellen wordt vergeleken met die van de Markov- en semi-Markov modellen. Tenslotte introduceren we een methode, die het mogelijk maakt om probabilistische modellen voor activiteiten herkenning toe te passen in meerdere huizen. Normaal gesproken heeft een probabilistisch model geannoteerde data nodig zodat de parameters van het model geleerd kunnen worden. Een model dat ontwikkeld is met een bepaald huis in gedachte kan niet zo maar gebruikt worden voor een ander huis. Dit omdat er verschillen zijn in de indeling van een huis en in het gedrag van de bewoners. Wij presenteren een 'transfer learning' methode die het mogelijk maakt geannoteerde data van huizen te gebruiken om de model-parameters van een ander huis te leren. Dit maakt het mogelijk om modellen voor activiteiten herkenning op grote schaal toe te passen en maakt daarmee de realisatie van een breed toepasbare oplossing voor efficiënte zorg voor ouderen haalbaar.

Bibliography

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC, pages 304–307, London, UK, 1999. Springer-Verlag.

[2] J. K. Aggarwal and Sangho Park. Human motion: Modeling and recognition of actions and interactions. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization, and Transmission*, 3DPVT, pages 640–647, Washington, DC, USA, 2004. IEEE Computer Society.

[3] KH Asberg. Assessment of adl in home-care for the elderly. change in adl and use of short-term hospital care. *Scandinavian Journal of Social Medicine*, 14(2):105–111, 1986.

[4] L. Atallah and G.Z. Yang. The use of pervasive sensing for behaviour profiling–a survey. *Pervasive and Mobile Computing*, 5(5):447–464, 2009.

[5] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[6] E.M. Bertera, B.Q. Tran, E.M. Wuertz, and A. Bonner. A study of the receptivity to telecare technology in a community-based elderly minority population. *Journal of telemedicine and telecare*, 13(7):327–323, 2007.

[7] Michael D. Beynon, Daniel J. Van Hook, Michael Seibert, Alen Peacock, and Dan Dudgeon. Detecting abandoned packages in a multi-camera video surveillance system. In *Proceedings of the international conference on Advanced Video and Signal Based Surveillance*, AVSS, pages 221–228, Washington, DC, USA, 2003. IEEE Computer Society.

[8] Jeff Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, International Computer Science Institute, 1997.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, August 2006.

[10] Bosch Health Buddy. http://www.healthbuddy.com/.

[11] B. Bouchard, S. Giroux, and A. Bouzouane. A logical approach to adl recognition for alzheimer's patients. In *Proceedings of the 4th international conference on Smart Homes and Health Telematics*, ICOST, pages 122–129, 2006.

[12] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity recognition from accelerometer data on a mobile phone. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks*, IWANN, pages 796–799, Berlin, Heidelberg, 2009. Springer-Verlag.

[13] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th international conference on Mobile Computing and Networking*, MobiCom, pages 85–97, New York, NY, USA, 1998. ACM.

[14] Richard H. Byrd, Jorge Nocedal, and Robert B. Schnabel. Representations of quasi-newton matrices and their use in limited memory methods. *Math. Program.*, 63(2):129–156, 1994.

[15] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[16] F. Casacuberta. Some relations among stochastic finite state networks used in automatic speech recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(7):691–695, 2002.

[17] Joseph Chamie. World population ageing. Technical report, Department of Economic and Social Affairs, Population Division, United Nations, 2009.

[18] Nait Charif and J. McKenna. Tracking the activity of participants in a meeting. *Mach. Vision Appl.*, 17(2):83–93, 2006.

[19] P. Charniak Robert et al. A bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.

[20] Liming Chen, Chris Nugent, Maurice Mulvenna, Dewar Finlay, Xin Hong, and Michael Poland. Using event calculus for behaviour reasoning and assistance in a smart home. In *Proceedings of the 6th international conference on Smart Homes and Health Telematics*, ICOST, pages 81–89, Berlin, Heidelberg, 2008. Springer-Verlag.

[21] Bill Chiu, Eamonn Keogh, and Stefano Lonardi. Probabilistic discovery of time series motifs. In *Proceedings of the 9th international conference on Knowledge discovery and data mining*, KDD, pages 493–498, New York, NY, USA, 2003. ACM.

[22] N. Chomsky. Three models for the description of language. *IRE Trans. on Information Theory*, 2(3):113–124, 2002.

[23] Calvin Chow. Parzen-window network intrusion detectors. In *Proceedings of the 16th International Conference on Pattern Recognition*, ICPR, pages 385 – 388, Washington, DC, USA, 2002. IEEE Computer Society.

[24] Sunny Consolvo, David W. McDonald, Tammy Toscos, Mike Y. Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis

LeGrand, Ryan Libby, Ian Smith, and James A. Landay. Activity sensing in the wild: a field trial of ubifit garden. In *Proceedings of the 26th international conference on Human Factors in Computing Systems*, CHI, pages 1797–1806, New York, NY, USA, 2008. ACM.

[25] Wenyuan Dai, Yuqiang Chen, Gui-Rong Xue, Qiang Yang, and Yong Yu. Translated learning: Transfer learning across different feature spaces. In *Advances in Neural Information Processing Systems 21*, NIPS, pages 353–360, Vancouver, Canada, 2008.

[26] Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Transferring naive bayes classifiers for text classification. In *Proceedings of the 22nd Conference of the Association for the Advancement of Artificial Intelligence*, AAAI, pages 540–545, 2007.

[27] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, SenSys, pages 171–180, New York, NY, USA, 2003. ACM.

[28] Raghu Das. Rfid explained. Technical report, Free IDTech White paper, 2002.

[29] Thomas Degen, Heinz Jaeckel, Michael Rufer, and Stefen Wyss. Speedy: A fall detector in a wrist watch. In *Proceedings of the 7th International Symposium on Wearable Computers*, ISWC, pages 184 – 187, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[30] A.P. Dempster, N.M. Laird, D.B. Rubin, et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[31] A. DiCenso, N. Cullum, and D. Ciliska. Implementing evidence-based nursing: some misconceptions. *Evidence Based Nursing*, 1(2):38, 1998.

[32] Thi Duong, Dinh Phung, Hung Bui, and Svetha Venkatesh. Efficient duration and hierarchical modeling for human activity recognition. *Artificial Intelligence*, 173(7-8):830–856, 2009.

[33] 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 international conference on Computer Vision and Pattern Recognition*, CVPR, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society.

[34] Jonny Farringdon, Andrew J. Moore, Nancy Tilbury, James Church, and Pieter D. Biemond. Wearable sensor badge and sensor jacket for context awareness. In *Proceedings of the 3rd International Symposium on Wearable Computers*, ISWC, pages 107–113, 1999.

[35] Silvia Ferrando, Gianluca Gera, and Carlo Regazzoni. Classification of unattended and stolen objects in video-surveillance system. In *Proceedings of the international conference on Advanced Video and Signal Based Surveillance*, AVSS, page 21, Washington, DC, USA, 2006. IEEE Computer Society.

[36] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, 32(1):41–62, 1998.

[37] Carl Fischer, Kavitha Muthukrishnan, Mike Hazas, and Hans Gellersen. Ultrasound-aided pedestrian dead reckoning for indoor navigation. In *Proceedings of the 1st international workshop on Mobile entity localization and tracking in GPS-less environments*, MELT, pages 31–36, New York, NY, USA, 2008. ACM.

[38] Kenneth P. Fishkin, Matthai Philipose, and Adam Rea. Hands-on rfid: Wireless wearables for detecting use of objects. In *Proceedings of the 9th International Symposium on Wearable Computers*, ISWC, pages 38–43, Washington, DC, USA, 2005. IEEE Computer Society.

[39] James Fogarty, Carolyn Au, and Scott E. Hudson. Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition. In *Proceedings of the 19th symposium on User interface software and technology*, UIST, pages 91–100, New York, NY, USA, 2006. ACM Press.

[40] D. Freitag and A. McCallum. Information extraction with hmms and shrinkage. In *Proceedings of the AAAI Workshop on Machine Learning for Information Extraction*, pages 31–36, 1999.

[41] Mykola Galushka, Dave Patterson, and Niall Rooney. *Temporal Data Mining for Smart Homes*, chapter 6, pages 85–108. Springer, 2006.

[42] D. M. Gavrila. The visual analysis of human movement: a survey. *Comput. Vis. Image Underst.*, 73(1):82–98, 1999.

[43] B. Georis, M. Maziere, F. Bremond, and M. Thonnat. A video interpretation platform applied to bank agency monitoring. *IEE Seminar Digests*, 2004(10426):46–50, 2004.

[44] Asela Gunawardana, Milind Mahajan, Alex Acero, and John C. Platt. Hidden conditional random fields for phone classification. In *Proceedings of the European Conference on Speech Communication and Technology*, Interspeech, pages 1117–1120, 2005.

[45] Karen Zita Haigh and Holly Yanco. Automation as caregiver: A survey of issues and technologies. In *Proceedings of the AAAI Workshop on Automation as Caregiver'*, pages 39–53, 2002. AAAI Technical Report WS-02-02.

[46] R. K. Harle and A. Hopper. The potential for location-aware power management. In *Proceedings of the 10th international conference on Ubiquitous computing*, Ubicomp, pages 302–311, New York, NY, USA, 2008. ACM.

[47] 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.

[48] K.A. Heller, Y.W. Teh, D. Görür, and G. Unit. Infinite hierarchical hidden markov models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2009.

[49] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[50] J. Hoey, P. Poupart, C. Boutilier, and A. Mihailidis. Pomdp models for assistive technology. In *Proceedings of the AAAI Workshop on Caring Machines: AI in Eldercare*, 2005.

[51] Yu-Jin Hong, Ig-Jae Kim, Sang Chul Ahn, and Hyoung-Gon Kim. Mobile health monitoring system based on activity recognition using accelerometer. *Simulation Modelling Practice and Theory*, 18(4):446 – 455, 2010. Modeling and Simulation Techniques for Future Generation Communication Networks.

[52] Toshio Hori, Yoshifumi Nishida, and Shin'ichi Murakami. Pervasive sensor system for evidence based nursing care support. In *Proceedings of the International Conference on Robotics and Automation*, ICRA, pages 1680 – 1685, 2006.

[53] Derek Hao Hu, Sinno Jialin Pan, Vincent Wenchen Zheng, Nathan Nan Liu, and Qiang Yang. Real world activity recognition with multiple goals. In *Proceedings of the 10th international conference on Ubiquitous computing*, Ubicomp, pages 30–39, New York, NY, USA, 2008. ACM.

[54] Q. Huo, C. Chan, and C.H. Lee. Bayesian adaptive learning of the parameters of hidden markov model for speech recognition. *IEEE Trans. on Speech and Audio Processing*, 3(5):334–345, 1995.

[55] Intel Health Guide. http://www.intel.com/healthcare/ps/healthguide/.

[56] S.S. Intille. A new research challenge: persuasive technology to motivate healthy aging. *IEEE Trans. on Information Technology in Biomedicine*, 8(3):235–237, 2004.

[57] Stephen S. Intille, Kent Larson, Emmanuel Munguia Tapia, Jennifer Beaudin, Pallavi Kaushik, Jason Nawyn, and Randy Rockinson. Using a live-in laboratory for ubiquitous computing research. In *Proceedings of the*

4th international conference on Pervasive Computing, Pervasive, pages 349–365, 2006.

[58] Stephen S. Intille, Emmanuel Munguia Tapia, John Rondoni, Jennifer Beaudin, Chuck Kukla, Sitij Agarwal, Ling Bao, and Kent Larson. Tools for studying behavior and technology in natural settings. In *Proceedings of the 5th international conference on Ubiquitous computing*, Ubicomp, pages 157–174, 2003.

[59] AM Jehad Sarkar, Y. Lee, and S. Lee. A smoothed naive bayes-based classifier for activity recognition. *IETE Technical Review*, 27(2):107, 2010.

[60] ShengYi Jiang, Xiaoyu Song, Hui Wang, Jian-Jun Han, and Qing-Hua Li. A clustering-based method for unsupervised intrusion detections. *Pattern Recognition Letters*, 27(7):802–810, 2006.

[61] Alvin H. Kam, Wenmiao Lu, and Wei-Yun Yau. A video-based drowning detection system. In *Proceedings of the 7th European Conference on Computer Vision*, ECCV, pages 297–311, London, UK, 2002. Springer-Verlag.

[62] Irek Karkowski, Marius Wulffers, and Paul van Wezenberg. Intelligent observatiesysteem voor zelfstandig wonende ouderen. *TNO Magazine*, Maart:10, 2007.

[63] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse. An activity monitoring system for elderly care using generative and discriminative models. *Journal of Personal and Ubiquitous Computing*, Special Issue on Pervasive Technologies for Assistive Environments:1–10, 2010.

[64] T.L.M. van Kasteren, G. Englebienne, and B.J.A. Kröse. Recognizing activities in multiple contexts using transfer learning. In *Proceedings of the AAAI Fall Symposium on AI in Eldercare: New Solutions to Old Problems*. AAAI Press, 2008. ISBN=978-1-57735-394-2.

[65] T.L.M. van Kasteren, G. Englebienne, and B.J.A. Kröse. Activity recognition using semi-markov models on real world smart home datasets. *Journal of Ambient Intelligence and Smart Environments*, 2(3):311–325, 2010.

[66] T.L.M. van Kasteren, G. Englebienne, and B.J.A. Kröse. Transferring knowledge of activity recognition across sensor networks. In *Proceedings of the 8th international conference on Pervasive Computing*, Pervasive, pages 283–300, 2010.

[67] T.L.M. van Kasteren, G. Englebienne, and B.J.A. Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In Liming Chen, Chris Nugent, Jit Biswas, and Jesse Hoey, editors, *Activity Recognition in Pervasive Intelligent Environments*, Atlantis Ambient and Pervasive Intelligence series. Atlantis Press, 2010.

[68] T.L.M. van Kasteren and B.J.A. Kröse. Bayesian activity recognition in residence for elders. In *Proceedings of the 3rd International Conference on Intelligent Environments*, IE, pages 209–212, Ulm, Germany, 2007.

[69] T.L.M. van Kasteren and B.J.A. Kröse. Care: Context awareness in residences for elderly. *IEEE Pervasive Computing*, 6:59–63, January-March 2007.

[70] T.L.M. van Kasteren and B.J.A. Kröse. A sensing and annotation system for recording datasets in multiple homes. In *Proceedings of CHI workshop: Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*, 2009.

[71] T.L.M. van Kasteren, A. Noulas, G. Englebienne, and B.J.A. Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, Ubicomp, pages 1–9, New York, NY, USA, 2008. ACM.

[72] S. Katz. Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living. *Journal Am. Geriatrics Soc.*, 31(12):721–726, 1983.

[73] Henry A. Kautz. *A formal theory of plan recognition and its implementation*, chapter 2, pages 69–126. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.

[74] Cory D. Kidd, Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth D. Mynatt, Thad Starner, and Wendy Newstetter. The aware home: A living laboratory for ubiquitous computing research. In *Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, CoBuild, pages 191–198, London, UK, 1999. Springer-Verlag.

[75] U. Kjaerulff. dhugin: A computational system for dynamic time-sliced bayesian networks. *International journal of forecasting*, 11(1):89–111, 1995.

[76] A. Krogh, M. Brown, I.S. Mian, K. Sjolander, and D. Haussler. Hidden markov models in computational biology. applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 1994.

[77] Kristof van Laerhoven, Hans-W. Gellersen, and Yanni G. Malliaris. Long-term activity monitoring with a wearable sensor node. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, BSN, pages 171–174, Washington, DC, USA, 2006. IEEE Computer Society.

[78] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th international conference on Machine learning*, ICML, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[79] John Lafferty, Xiaojin Zhu, and Yan Liu. Kernel conditional random fields: representation and clique selection. In *Proceedings of the 21st international conference on Machine learning*, ICML, pages 64–71, New York, NY, USA, 2004. ACM.

[80] James Landay and Anthony LaMarca. Intel's seattle lab making an impact. Technical report, Intel, 2005.

[81] Niels Landwehr, Bernd Gutmann, Ingo Thon, Matthai Philipose, and Luc De Raedt. Relational transformation-based tagging for human activity recognition. In *Proceedings of the 6th International Workshop on Multi-relational Data Mining*, MRDM, pages 81–92, Warsaw, Poland, September 2007.

[82] K. Lari and S.J. Young. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech & Language*, 5(3):237–257, 1991.

[83] Su-In Lee, Vassil Chatalbashev, David Vickrey, and Daphne Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning*, ICML, pages 489–496, New York, NY, USA, 2007. ACM.

[84] Jonathan Lester, Tanzeem Choudhury, Nicky Kern, Gaetano Borriello, and Blake Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *Proceedings of the 19th international joint conference on Artificial intelligence*, IJCAI, pages 766–772, 2005.

[85] SE Levinson. Continuously variable duration hidden markov models for automatic speech recognition. *Computer Speech & Language*, 1(1):29–45, 1986.

[86] Lin Liao, Dieter Fox, and Henry Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, Vol. 26:No. 1, 119–134, 2007.

[87] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, 1989.

[88] Beth Logan, Jennifer Healey, Matthai Philipose, Emmanuel Tapia, and Stephen Intille. A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of the 9th international conference on Ubiquitous computing*, Ubicomp, pages 483–500. Springer Berlin / Heidelberg, 2007.

[89] E.F. LoPresti, A. Mihailidis, and N. Kirsch. Assistive technology for cognitive rehabilitation: State of the art. *Neuropsychological Rehabilitation*, 14(1-2):5–39, 2004.

[90] W. Lu and Y.P. Tan. A vision-based approach to early detection of drowning

incidents in swimming pools. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(2):159–178, 2004.

[91] Sebastian Lühr, Hung H. Bui, Svetha Venkatesh, and Geoff A. W. West. Recognition of human activity through hierarchical stochastic learning. In *Proceedings of the 1st International Conference on Pervasive Computing and Communications*, PerCom, pages 416 – 422, Washington, DC, USA, 2003. IEEE Computer Society.

[92] P. Lukowicz, G. Pirkl, D. Bannach, F. Wagner, A. Calatroni, K. Förster, T. Holleczek, M. Rossi, D. Roggen, G. Troester, et al. Recording a complex, multi modal activity data set for context recognition. In *Proceedings of the ARCS Workshop on Context-Systems Design, Evaluation and Optimisation*, ARCS, 2010.

[93] Paul Lukowicz, H. Junker, M. Stäger, T. von Büren, and G. Tröster. Wearnet: A distributed multi-sensor system for context aware wearables. In *Proceedings of the 4th international conference on Ubiquitous computing*, Ubicomp, pages 361–370, London, UK, 2002. Springer-Verlag.

[94] M.S. Magnusson. Discovering hidden time patterns in behavior: T-patterns and their detection. *Behavior Research Methods Instruments and Computers*, 32(1):93–110, 2000.

[95] Dhruv Mahajan, Nipun Kwatra, Sumit Jain, Prem Kalra, and Subhashis Banerjee. A framework for activity recognition and detection of unusual activities. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pages 15–21, 2004.

[96] Mihai Marin-Perianu, Clemens Lombriser, Oliver Amft, Paul Havinga, and Gerhard Tröster. Distributed activity recognition with fuzzy-enabled wireless sensor networks. In *Proceedings of the 4th international conference on Distributed Computing in Sensor Systems*, DCOSS, pages 296–313, Berlin, Heidelberg, 2008. Springer-Verlag.

[97] Uwe Maurer, Asim Smailagic, Daniel P. Siewiorek, and Michael Deisher. Activity recognition and monitoring using multiple sensors on different body positions. In *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks*, BSN, pages 113–116, Washington, DC, USA, 2006. IEEE Computer Society.

[98] W.W. Mayol, A.J. Davison, B.J. Tordoff, N.D. Molton, and D.W. Murray. Interaction between hand and wearable camera in 2d and 3d environments. In *British Machine Vision Association*, BMVC, 2004.

[99] Iain McCowan, Daniel Gatica-Perez, Samy Bengio, Guillaume Lathoud, Mark Barnard, and Dong Zhang. Automatic analysis of multimodal group

actions in meetings. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):305–317, 2005.

[100] Susan McKeever, Juan Ye, Lorcan Coyle, and Simon Dobson. Using dempster-shafer theory of evidence for situation inference. In *Proceedings of the 4th European conference on Smart sensing and context*, EuroSSC, pages 149–162, Berlin, Heidelberg, 2009. Springer-Verlag.

[101] E. Meara, C. White, and D.M. Cutler. Trends in medical spending by age, 1963-2000. *Health Affairs*, 23(4):176, 2004.

[102] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *Proceedings of the 12th International Conference on Computer Vision*, ICCV, Washington, DC, USA, 2009. IEEE Computer Society.

[103] Juan Carlos San Miguel and José M. Martínez. Robust unattended and stolen object detection by fusing simple algorithms. In *Proceedings of the international conference on Advanced Video and Signal Based Surveillance*, AVSS, pages 18–25, Washington, DC, USA, 2008. IEEE Computer Society.

[104] A. Mihailidis, J.C. Barbenel, and G. Fernie. The efficacy of an intelligent cognitive orthosis to facilitate handwashing by persons with moderate to severe dementia. *Neuropsychological Rehabilitation*, 14(1-2):135–171, 2004.

[105] Thomas P. Minka. Estimating a dirichlet distribution. Technical report, Microsoft Research, 2000.

[106] F.G. Miskelly. Assistive technology in elderly care. *Age and ageing*, 30(6):455, 2001.

[107] Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, 104(2):90–126, 2006.

[108] David Molnar and David Wagner. Privacy and security in library rfid: issues, practices, and architectures. In *Proceedings of the 11th conference on Computer and communications security*, CCS, pages 210–219, New York, NY, USA, 2004. ACM.

[109] Louis Philippe Morency, Ariadna Quattoni, and Trevor Darrell. Latent-dynamic discriminative models for continuous gesture recognition. In *Proceedings of the international conference on Computer Vision and Pattern Recognition*, CVPR, pages 1–8, 2007.

[110] J. Muncaster and Y. Ma. Hierarchical model-based activity recognition with automatic low-level state discovery. *Journal of Multimedia*, 2(5):66, 2007.

[111] Kevin P. Murphy. Hidden semi-markov models (hsmms). Technical report, University of British Columbia, 2002.

[112] K.P. Murphy and M.A. Paskin. Linear time inference in hierarchical hmms. In *Advances in Neural Information Processing Systems 14*, NIPS, 2001.

[113] E.D. Mynatt, I. Essa, and W. Rogers. Increasing the opportunities for aging in place. In *Proceedings on the conference on Universal Usability*, pages 65–71, 2000.

[114] E.D. Mynatt and W.A. Rogers. Developing technology to support the functional independence of older adults. *Ageing International*, 27(1):24–41, 2001.

[115] Hammadi Nait-Charif and Stephen J. McKenna. Activity summarisation and fall detection in a supportive home environment. In *Proceedings of the 17th International Conference on Pattern Recognition*, ICPR, pages 323–326, Washington, DC, USA, 2004. IEEE Computer Society.

[116] Futoshi Naya, Ren Ohmura, Fusako Takayanagi, Haruo Noma, and Kiyoshi Kogure. Workers' routine activity recognition using body movements and location information. In *Proceedings of the 10th International Symposium on Wearable Computing*, ISWC, pages 105–108, 2006.

[117] A.Y. Ng and M.I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems 14*, NIPS, pages 841–848, 2001.

[118] Nam T. Nguyen, Dinh Q. Phung, Svetha Venkatesh, and Hung Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In *Proceedings of the international conference on Computer Vision and Pattern Recognition*, CVPR, pages 955–960, Washington, DC, USA, 2005. IEEE Computer Society.

[119] N.T. Nguyen, S. Venkatesh, and H.H. Bui. Recognising behaviours of multiple people with hierarchical probabilistic model and statistical data association. In *British Machine Vision Association*, BMVC, pages 1239–1248, 2006.

[120] T. Oates. Peruse: An unsupervised algorithm for finding recurring patterns in time series. In *Proceedings of the International Conference on Data Mining*, ICDM, pages 330 – 337, 2002.

[121] S. Ohta, H. Nakamoto, Y. Shinagawa, and T. Tanikawa. A health monitoring system for elderly people living alone. *Journal of telemedicine and telecare*, 8:151–156, 2002.

[122] Daniel Olguín Olguín and Alex Sandy Pentland. Human activity recognition: Accuracy across common locations for wearable sensors. In *Proceedings of the 10th International Symposium on Wearable Computing (Student Colloquium Proceedings)*, ISWC, pages 11–14, October 2006.

[123] Nuria Oliver, Ashutosh Garg, and Eric Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Comput. Vis. Image Underst.*, 96(2):163–180, 2004.

[124] Donald J. Patterson, Dieter Fox, Henry A. Kautz, and Matthai Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *Proceedings of the 9th International Symposium on Wearable Computers*, ISWC, pages 44–51. IEEE Computer Society, 2005.

[125] William Pentney, Ana-Maria Popescu, Shiaokai Wang, Henry Kautz, and Matthai Philipose. Sensor-based understanding of daily life via large-scale use of common sense. In *Proceedings of the 21st Conference of the Association for the Advancement of Artificial Intelligence*, AAAI, pages 906–912. AAAI Press, 2006.

[126] Mike Perkowitz, Matthai Philipose, Kenneth P. 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, pages 573–582. ACM, 2004.

[127] Susanna Pirttikangas, Kaori Fujinami, and Tatsuo Nakajima. Feature selection and activity recognition from wearable sensors. *Ubiquitous Computing Systems*, 4239:516–527, 2006.

[128] R. Plamondon and S.N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):63–84, 2000.

[129] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28:976–990, June 2010.

[130] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[131] M.A. Rahman, M.F. Alhamid, W. Gueaieb, and A. El Saddik. An ambient intelligent body sensor network for e-health applications. In *Proceedings of the International Workshop on Medical Measurements and Applications*, MeMeA, pages 22–25, 2009.

[132] Rajat Raina, Andrew Y. Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning*, ICML, pages 713–720, New York, NY, USA, 2006. ACM.

[133] Rajat Raina, Yirong Shen, Andrew Y. Ng, and Andrew McCallum. Classification with hybrid generative/discriminative models. In *Advances in Neural Information Processing Systems 16*, NIPS, Cambridge, MA, 2003. MIT Press.

[134] Uwe E. Reinhardt. Does the aging of the population really drive the demand for health care? *Health Affairs*, 22(6):27–39, November 2003.

[135] RFM. http://www.rfm.com/.

[136] I. Rish. An empirical study of the naive bayes classifier. In *Proceedings of the 17th international joint conference on Artificial intelligence*, IJCAI, pages 41–46, 2001.

[137] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, et al. Collecting complex activity data sets in highly rich networked sensor environments. In *Proceedings of the 7th International Conference on Networked Sensing Systems*, INSS, 2010.

[138] Alberto Rugnone, Francesco Poli, Enrico Vicario, Chris D. Nugent, Elena Tamburini, and Cristiano Paggetti. A visual editor to support the use of temporal logic for adl monitoring. In *Proceedings of the 5th international conference on Smart Homes and Health Telematics*, ICOST, pages 217–225, 2007.

[139] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17*, NIPS, pages 1185–1192, 2004.

[140] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, IDA, pages 309–318, London, UK, UK, 2001. Springer-Verlag.

[141] B. Settles. Active learning literature survey. *Machine Learning*, 15(2):201–221, 1994.

[142] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL, pages 134–141, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[143] Priya Narasimhan Shahriyar Amini. Twitterjacket: An automated activity and health monitoring solution for the elderly. Technical report, CMU, 2009.

[144] S.T. Shivappa, M.M. Trivedi, and B.D. Rao. Hierarchical audio-visual cue integration framework for activity analysis in intelligent meeting rooms. In *Proceedings of the CVPR Workshop*, pages 107–114, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[145] Amit Sinha and Anantha Chandrakasan. Dynamic power management in wireless sensor networks. *IEEE Des. Test*, 18(2):62–74, 2001.

[146] Marios Skounakis, Mark Craven, and Soumya Ray. Hierarchical hidden markov models for information extraction. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI, pages 427–433, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[147] K. Sohrabi, J. Gao, V. Ailawadhi, and G.J. Pottie. Protocols for self-organization of a wireless sensor network. *IEEE Personal Communications*, 7(5):16–27, 2000.

[148] Thomas Stiefmeier, Georg Ogris, Holger Junker, Paul Lukowicz, and Gerhard Tröster. Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario. In *Proceedings of the 10th International Symposium on Wearable Computing*, ISWC, pages 97–104, 2006.

[149] Maja Stikic, Tâm Huynh, Kristof Van Laerhoven, and Bernt Schiele. Adl recognition based on the combination of rfid and accelerometer sensing. In *Proceedings of the 2nd International Conference on Pervasive Computing Technologies for Healthcare (Pervasive Health 2008)*, pages 258–263, Tampere, Finland, January 2008. IEEE.

[150] Maja Stikic and Kristof Van Laerhoven. Recording housekeeping activities with situated tags and wrist-worn sensors: Experiment setup and issues encountered. In *Proceedings of the INSS workshop on Wireless Sensor Networks for Health Care*, WSNHC, 2007.

[151] Amarnag Subramanya, Alvin Raj, Jeff Bilmes, and Dieter Fox. Hierarchical models for activity recognition. In *Proceedings of the international conference on Multimedia Signal Processing*, MMSP, Victoria, CA, October 2006.

[152] S. Sundaram and W.W. Mayol Cuevas. High level activity recognition using low resolution wearable vision. In *Proceedings of the CVPR Workshop on Egocentric Vision*, pages 25–32, 2009.

[153] Charles Sutton and Andrew McCallum. *Introduction to Statistical Relational Learning*, chapter 1: An introduction to Conditional Random Fields for Relational Learning, page (Available online). MIT Press, 2006.

[154] Tony Tam, Alf Dolan, Jennifer Boger, and Alex Mihailidis. An intelligent emergency response system: Preliminary development and testing of a functional health monitoring system. *Gerontechnology*, 4:209–222, March 2006.

[155] Toshiyo Tamura, Tatsuo Togawa, Mitsuhiro Ogawa, and Mikiko Yoda. Fully automated health monitoring system in the home. *Medical Engineering and Physics*, 20:573–579, 1998.

[156] E. Munguia Tapia, S.S. Intille, L. Lopez, and K. Larson. The design of a portable kit of wireless sensors for naturalistic data collection. In *Proceedings*

of the 4th international conference on Pervasive Computing, Pervasive, pages 117–134. Springer, 2006.

[157] Emmanuel Munguia Tapia, Tanzeem Choudhury, and Matthai Philipose. Building reliable activity models using hierarchical shrinkage and mined ontology. In *Proceedings of the 4th international conference on Pervasive Computing*, Pervasive, pages 17–32. Springer, 2006.

[158] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Proceedings of the 2nd international conference on Pervasive Computing*, Pervasive, pages 158–175, Vienna, Austria, April 2004.

[159] R. Tavenard, A.A. Salah, and E.J. Pauwels. Searching for temporal patterns in ami sensor data. In *Proceedings of the AmI Workshops*, AmI, November 2007.

[160] T. Teixeira and A. Savvides. Lightweight people counting and localizing in indoor spaces using camera sensor nodes. In *Proceedings of the International Conference on Distributed Smart Cameras*, ICDSC, pages 36–43, 2007.

[161] David Minnen Thad, David Minnen, Thad Starner, Irfan Essa, and Charles Isbell. Discovering characteristic actions from on-body sensor data. In *Proceedings of the 10th International Symposium on Wearable Computing*, ISWC, pages 11–18, 2006.

[162] S. Thrun. Is learning the nth thing any easier than learning the first? In *Advances in Neural Information Processing Systems 08*, NIPS, pages 640–646, 1995.

[163] S. Thrun. A lifelong learning perspective for mobile robot control. In *Proceedings of the International Conference on Intelligent Robots and Systems*, IROS, pages 23–30, 2002.

[164] A. Tolstikov, X. Hong, J. Biswas, C. Nugent, L. Chen, and G. Parente. Comparison of fusion methods based on DST and DBN in human activity recognition. *Journal of Control Theory and Applications*, 9(1):18–27, 2011.

[165] Tran The Truyen, Dinh Q. Phung, Hung H. Bui, and Svetha Venkatesh. Hierarchical semi-markov conditional random fields for recursive sequential data. In *Advances in Neural Information Processing Systems 21*, NIPS, pages 1657–1664, 2008.

[166] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Trans. on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.

[167] 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, 2007.

[168] Vicente Luis Atienza Vanacloig, Juan Alfonso Rosell Ortega, Gabriela Andreu García, and José Miguel Valiente González. People and luggage recognition in airport surveillance under real-time constraints. In *Proceedings of the 19th International Conference on Pattern Recognition*, ICPR, pages 1–4, 2008.

[169] H. Wallach. Efficient training of conditional random fields. In *Proceedings of the 6th Computational Linguistics UK Research Colloquium*, 2002.

[170] Liang Wang, Tao Gu, Xianping Tao, and Jian Lu. Sensor-based human activity recognition in a multi-user scenario. In *Proceedings of the European Conference on Ambient Intelligence*, AmI, pages 78–87, Berlin, Heidelberg, 2009. Springer-Verlag.

[171] Sy Bor Wang, Ariadna Quattoni, Louis-Philippe Morency, and David Demirdjian. Hidden conditional random fields for gesture recognition. In *Proceedings of the international conference on Computer Vision and Pattern Recognition*, CVPR, pages 1521–1527, Washington, DC, USA, 2006. IEEE Computer Society.

[172] Murray Todd Williams. Beta-binomial distribution for proportional confidence intervals. Technical report, University of Leeds, 1998.

[173] Daniel Wilson and Chris Atkeson. Simultaneous tracking and activity recognition (star) using many anonymous binary sensors. In *Proceedings of the 3rd international conference on Pervasive Computing*, Pervasive, pages 62–79, Munich , Germany, 2005.

[174] Daniel H. Wilson. *Assistive Intelligent Environments for Automatic Health Monitoring*. PhD thesis, Carnegie Mellon University, 2005.

[175] Daniel H. Wilson, Sunny Consolvo, Kenneth P. Fishkin, and Matthai Philipose. Current practices for in-home monitoring of eldersŠ activities of daily living: A study of case managers. Technical report, Intel Research Seattle, 2005.

[176] C. R. Wren and E. Munguia Tapia. Toward scalable activity recognition for sensor networks. In *Proceedings of the 2nd International Workshop in Location and Context-Awareness*, LoCA, 2006.

[177] Jianxin Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg. A scalable approach to activity recognition based on object use. In *Proceedings of the 11th International Conference on Computer Vision*, ICCV, pages 1–8, December 2007.

[178] Danny Wyatt, Matthai Philipose, and Tanzeem Choudhury. Unsupervised activity recognition using automatically mined common sense. In *Proceedings of the 20th Conference of the Association for the Advancement of Artificial Intelligence*, AAAI, pages 21–27. AAAI Press; AAAI Press / The MIT Press, 2005.

[179] Ya Xu, John Heidemann, and Deborah Estrin. Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th international conference on Mobile Computing and Networking*, MobiCom, pages 70–84, New York, NY, USA, 2001. ACM.

[180] J. Ye and S. Dobson. Exploring semantics in activity recognition using context lattices. *Journal of Ambient Intelligence and Smart Environments*, 2(4):389–407, 2010.

[181] W. Zajdel, J. D. Krijnders, T. Andringa, and D. M. Gavrila. Cassandra: Audio-video sensor fusion for aggression detection. In *Proceedings of the international conference on Advanced Video and Signal Based Surveillance*, AVSS, 2007.

[182] Piero Zappi, Clemens Lombriser, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. Activity recognition from on-body sensors: accuracy-power trade-off by dynamic sensor selection. In *Proceedings of the 5th European conference on Wireless sensor networks*, EWSN'08, pages 17–33, Berlin, Heidelberg, 2008. Springer-Verlag.

[183] Jian Zhang, Zoubin Ghahramani, and Yiming Yang. Learning multiple related tasks using latent independent component analysis. In *Advances in Neural Information Processing Systems 18*, NIPS, pages 1585–1592, Cambridge, MA, 2005. MIT Press.

[184] Vincent Wenchen Zheng, Derek Hao Hu, and Qiang Yang. Cross-domain activity recognition. In *Proceedings of the 11th international conference on Ubiquitous computing*, Ubicomp, pages 61–70, New York, NY, USA, 2009. ACM.

Acknowledgements

First of all I would like to thank my direct colleagues with whom I worked during the PhD. I am very happy Ben Kröse chose me for this PhD project and taught me so many things about how to do good science. Gwenn Englebienne, you made a wonderful addition to our team and I hope we will get the opportunity for further collaboration in the future. My promotor Frans Groen has been actively involved in the final stages of the PhD, it was great to have been part of your group and I am grateful for your insightful feedback.

Out of my personal circle of friends three people in particular come to mind, who have been a great support in the different stages of the PhD. Athanasios Noulas, you helped me so much in the first stages of the PhD, your dedication in working with me and your support will never be forgotten. Ana Marcheva, your endless positive energy, beautiful expressive nature and confidence in my capabilities have helped me so much and continue to help me to this very day. Thomas Bernard, my PhD soul mate, I have never felt so in sync with someone as I do with you, our discussions, debates and sharing of experiences makes me feel part of a team and gives me the strength to take on any challenge.

Throughout my years in the UvA I have met a lot of people with whom I have shared experiences on a personal or professional level. Carsten Cibura, you are a great blend of rationalism and fun, thanks for all the great breaks. Bas Terwijn no discussion is complete without a perspective given from you. Gerben de Vries, always a pleasure, let's see where our paths cross next time. Everybody from the Borrel commissie, thanks for all the good times. Thanks to my old office mates Frans Oliehoek and Bram Bakker. And thanks to all the rest of the group, Olaf Booij, Michael Hoffman, Martijn Liem, Julian Kooij, Shimon Whiteson, Leo Dorst, Arnoud Visser, Gregor Pavlin and Patrick de Oude.

During my trips to conferences I have also met some great fellow researchers that have been an inspiration for continuing my work in science. Alex de Luca, Xueli An, Fahim Kawsar, Dawud Gordon, attending a conference feels incomplete without you guys, thanks for all the good times. Also many thanks to everyone that I have met along the way and that took the time to feedback on my work.

My recent move to Turkey has made me part of a family of researchers at the Boğaziçi University in Istanbul. Cem Ersoy, thank you for giving me the opportunity to work in your group and for your inspiring words towards my work method. Hande, Remzi, Bilgin, Ozan, Özlem and Rabun thank you so much for giving me such a warm welcome in joining your group, changing work places has never been so comfortable. Many thanks to all the people I have met during my stay in Turkey that have helped me to relax and forget the PhD every now and then. And many many thanks to Taylan Cemgil for giving me a great perspective on science and getting me in touch with such a wonderful group of researchers.

Moving abroad can be exciting and challenging, but I won't forget the support I received from my friends back home. Robert Gigengack, you are my buddy for life man, thanks for always being there. Zeynep Baldemir, my sister, thanks for always looking out for me. Ercan Tunc, life feels safer knowing you are keeping an eye out for me. Merijn Veken, thanks for always supporting me and keeping me sharp. Tim van den Dool, thanks for all the good times, both serious and casual. Loes Wenink, your positive energy always lifts my spirit. Sjoerd, IJsbrand, Frank, Kim, Casper and Xander, thanks for always being interested in my progress. And many thanks to so many other people from Amsterdam and Alkmaar for all the laughter and support.

Last but definitely not least, many thanks go out to my family. Thanks to my parents for always supporting me unconditionally. Your understanding and help are really invaluable to me. My brother Paul, many thanks for being there at any time, I know I can always rely on you. My uncle Willy, your contribution to my thesis is extremely valuable and I am enormously grateful for the effort you have put into realizing that contribution. Thanks to my other uncles for always being interested and keeping me sharp with your finely tuned questions and debates. Thanks to my aunts who always manage to find the energy to be curious about my research and my well-being. Thanks to all my cousins, you always make me feel so appreciated.

Thanks to each and every one of you I truly feel loved and blessed to have so many great people around me.