Behaviour Recognition in Smart Homes

A dissertation presented in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

in

Computer Science

at Massey University, Manawatu,

New Zealand

Sook-Ling Chua

2012

Abstract

In the context of this thesis, behaviour recognition aims to infer the particular behaviours of the inhabitant of a smart home from a series of sensor readings from around the house. This thesis views the behaviour recognition problem as a task of mapping the sensory outputs to a sequence of activities performed by the inhabitant. The main focus is the development of machine learning methods to find an approximation to the mapping between sensor outputs and behaviours.

While there have been many supervised machine learning methods for identifying behaviours from a sensor stream, they generally assume that the behaviours are either segmented or perform segmentation and behaviour recognition separately. In order to be used in the real world, segmentation and behaviour recognition should not be treated separately. This thesis addresses this problem based on a set of hidden Markov models (HMM) and a variable window length.

As the majority of the methods reported in the literature are based on supervised learning approach, they generally rely on a labelled dataset, where the behaviours of the inhabitant have to be manually labelled. This is often not practical in the real world. Most current unsupervised methods are not suitable for behaviour recognition as they are based on inputs of fixed dimensionality. In the smart home, the behaviours that are to be recognised are variable in length. This thesis introduces an unsupervised learning method that addresses this problem, which is based on compression and the edit distance between words. This includes both the segmentation of the sensor stream into suitable patterns and identification of patterns that correspond to human behaviours. This thesis also shows that the resulting method can be used to provide labels to training data for a supervised method.

However, training a learning algorithm on sensors that are irrelevant and/or redundant becomes crucial as they may affect the recognition performance. This thesis addresses the sensor selection problem for behaviour recognition through an

information-theoretic approach, which is based on information gain, modelled in the form of a decision tree. The main idea is to identify the set of informative sensors that are highly correlated with the behaviours. This thesis also presents solutions to address the 'generalisation' issues of the informative sensors identified across the different trees.

To evaluate the effectiveness of our proposed methods, we use a real smart home dataset obtained from the MIT PlaceLab and compare the labels produced by our methods with the labels assigned by a human to the activities in the sensor stream. We also validated our methods on other benchmark datasets and learning algorithms.

Acknowledgements

I owe the most thanks to my supervisors, Hans Guesgen and Stephen Marsland, for inspiring me in the field of machine learning. Their patience and kind guidance made me feel like I could do anything! They have kept me going, when from time to time, I became convinced that I was not going to make it. I am thankful to have been able to benefit from their combined research experience, wonderful mentorship and friendship. I couldn't have asked for more supportive supervisors and talented teachers.

Thanks to all the members of the Massey University Smart Home (MUSE) research group for input, stimulating discussions and valuable feedback on my research. Thanks to Massey University for the scholarship and doctoral completion bursary to support my degree and thesis writing. Thanks to the School of Engineering and Advanced Technology (SEAT) for the financial support to attend international conferences, which not only helped me in my research but also enriched my life.

There are numerous friends who contributed ideas, gave suggestions and valuable solutions to little problems that pop up unexpectedly. I would specifically like to mention Amissa Arifin, Derek Hao Hu and Frank Xu for their little favors that make a big difference. Thanks to Michele Wagner for all the administrative support throughout my degree. Special thanks must go to Lee Kien, who has made the whole degree more enjoyable, providing the necessary support and distraction when needed.

Thanks to my uncle, Yit Foong, who has been encouraging me to pursue my degree. Finally, I would also like to thank my parents and my sister for their continuous love, support and patience.

For my sister, Joanne, and my parents

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This first chapter provides an overview of the research background, outlines the problems being addressed and details why behaviour recognition in smart homes is in need of further study. It also describes the challenges in behaviour recognition, the aim and objectives of this thesis and the smart home dataset used in this thesis.

1.1 Motivation

It is a well-reported fact that the populations of the world are aging. In Europe, for example, it has been reported that the number of people aged 65 years and over is projected to increase from 10% of the entire population in 1950 to more than 25% in 2050. In New Zealand, the number of people over 65 years has doubled between 1970 and 2005 [28]. Besides the decrease in birth rate, the global life expectancy at birth is projected to increase from 58 years in 1970–1975 to 75 years in 2045–2050 [101]. The group of elderly (aged 65 years and above) is the fastest growing segment of the world's population.

Aging results in some degree of physical disability. Even when the elderly are physically healthy, the aging process is accompanied by cognitive impairment such as diminished sense of touch, slower ability to react, physical weakness and mem-

ory problems. Older adults are more frequently subject to physical disabilities and cognitive impairments than younger people. It is clearly impossible to rely solely on increasing the number of caregivers for the elderly, since even now it is difficult and expensive to find care [125]. Additionally, many people are choosing to stay in their own homes as long as possible, and hope to remain independent.

This has led to a large number of monitoring systems such as the 'smart homes', with the aim being to support the inhabitants, typically the elderly or cognitively impaired who are living alone, by monitoring their behaviours and detecting potentially dangerous behaviours. In this context, the 'smart' part of the smart home is the ability of the system to learn to use the information provided by the sensors (which are installed in the home) to recognise behaviours and identify deviations from the norm.

1.2 The Problem

In a smart home, the observed behaviours are likely to be the standard human behaviours of living, and the observations will depend upon the sensors that the house is equipped with. There are a variety of sensors that could be used to collect information about the inhabitant's activities. These sensors could range from cameras to body-worn, and to touch and motion sensors (a detailed review of these sensors is given in Section 2.3).

Although cameras could provide rich amount of information about the inhabitant and the environment, they are considered obtrusive since they invade the inhabitant's privacy. Sensors such as accelerometers, that can be worn on the body to collect information about body movement and pose, are also considered obtrusive; they are irritating to wear since the inhabitant needs to wear them on a daily basis.

Another sensing method, which is the focus of this thesis, is to recognise behaviours through unobtrusive sensors that are attached to household objects such

as microwave ovens, cupboards, taps, etc. Such sensors are activated when the inhabitant performs his/her daily activities in the home (e.g., opening or closing the microwave oven's door could activate the sensor attached to the microwave oven) and this generates a sequence of sensor observations, which corresponds to the order of events involved in carrying out a particular activity. For example, a sensor sequence such as closing the bathroom door, turning on the heater, opening the shower door, etc., may correspond to the events of a showering activity.

Since sensor observations from the home in some way represent the behaviours of a human inhabitant, this thesis views the behaviour recognition problem as a task of finding a mapping from a stream of sensor information to a sequence of activities performed by the inhabitant of a smart home. We aim for a general method of identifying the mapping that is independent of the particular sensors that are used. Obviously, what information is available will change the potential behaviours that can be detected. In such a smart home, the behaviour itself is performed by the inhabitant, and the sensors detect (potentially noisily) evidence of that behaviour.

There are many ways to learn the mapping between sensor outputs and behaviours. One is to manually label the data, either directly when observing the activities performed by the inhabitant or having the inhabitant keep a record of his activities. These data are usually segmented and labelled into different behaviours. Given a set of activity labels that are mapped to sensor information, we can train any supervised machine learning method. However, data from a sensor stream consists of an unending sequence of sensor readings. This means that there is a temporal ordering in the sensor readings. The nature of such data is similar to the time series data since both have a natural temporal ordering. The main difference is that, in time series, the sequence data point is generally spaced at uniform time intervals, while in a smart home, the sensors are only activated when the inhabitant performs their daily activities. Thus, the sensor readings are not necessarily spaced at uniform time intervals.

The challenge with non-uniformly spaced time intervals data is that the start and end of an activity are unknown. One way to handle this challenge is to segment the sensory stream into appropriate pieces that represent individual behaviours before any behaviour classification can be carried out.

The majority of the methods reported in the literature (see Section 3.2) either assume that activities have been segmented, or handle segmentation and classification of the sensor stream separately. Some approaches attempt to address the problem of segmentation using a fixed window length. However, it is unlikely that all of the sequences in the window belong to one behaviour and this results in other behaviours in the window not being recognised. This thesis addresses this problem by introducing a method that can accurately perform segmentation and behaviour recognition simultaneously on the sensory stream.

Although many behaviour recognition systems have been proposed, they are based on supervised learning algorithms and so training them requires that a dataset is prepared and annotated. One problem with supervised learning approaches is that they require a sufficient number of labelled data for training, which is often done by a person. Unfortunately, this is a rather time consuming task and prone to error; imagine a situation where someone needs to label several weeks of data before the recognition system can be put into use. This is not practical, especially when implementing a home for the elderly. We introduce a method that can automatically identify patterns and segment the data stream into patterns that correspond to human behaviours, without any prior human labelling.

The behaviour recognition problem described above aims to infer the particular behaviours of the inhabitant in a smart home from a series of sensor readings from the house. However, one challenge still remains:– which sensors are useful in order to effectively recognise the behaviours of the inhabitant. To see why this is important, consider the kitchen tap and washing machine as an example. When the kitchen tap

is running, there are many possible behaviours that have caused it; it could be that somebody was in the kitchen washing dishes, making tea (i.e., filling up the kettle for boiling water) or cooking (i.e., filling up the pot for boiling soup). At this stage, it would be rather difficult to determine what exactly the inhabitant is doing in the home. If the next sensor event is the sensor on the burner being triggered then we can be certain that the inhabitant is cooking.

This challenge is one of the main motivations for looking at the sensor selection problem in the smart home. The idea is that if we could find a way to identify one sensor that could tell us exactly what the inhabitant's behaviour is, then we can have a picture of what is going on in the house. The washing machine is one example. When it is running, we can be certain that the inhabitant is doing laundry.

The majority of the methods proposed in the literature (see Section 5.2) aim to identify the set of informative sensors that are useful for recognising human physical activities (e.g., walking, sitting, etc.). However, these methods are not useful when applied to recognising behaviours of daily living (e.g., cooking, doing laundry, washing dishes, toileting, etc.) where the capacity of the sensors attached to the household objects are larger and they span across different room locations in the home as compared to sensors worn on different parts of body.

This thesis addresses the sensor selection problem through an information-theoretic approach, based on information gain, modelled in the form of a decision tree. When a decision tree is trained on each partition of training data from a smart home or data from different smart homes, this will produce a set of different decision trees. This thesis presents a solution to gain insights about the significance of the informative sensors identified across the different trees.

1.3 Challenges in behaviour recognition

Given that we have outlined the problems being addressed, we are now in a position to discuss the challenges that make the behaviour recognition task difficult. These are listed below:

Behaviours are not directly observed The exact activities that the inhabitant is currently involved in are not directly observed. The only information provided is the sensor observations, which could be that the kitchen light is on, the oven is turned on and the burner is on; the inference that therefore somebody is cooking is left to the intelligent part of the system.

Some sensor events may be involved in multiple behaviours There are many behaviours that involve the same sensor activation. For example, the kitchen tap could be involved in washing dishes, making coffee or cooking.

Variation in behaviours The number of sensors that define a behaviour is not fixed. They can vary between behaviours (e.g., showering could involve sensors on the bathroom door and the shower tap while making dinner could involve the fridge, stove, microwave oven, tap and other sensors), and within different instances of the same activity (e.g., making a cup of tea may involve milk, or may not, and the milk could be added before or after the hot water).

Interleaving behaviours People rarely engage in one task and take it to completion before starting something else. For example, while cooking dinner the telephone may ring, or the person may require a trip to the toilet, or the cat may demand to be fed.

Segmentation As described in Section 1.2, a sensor stream consists of an unending sequence of sensor readings where the start and end of an activity is unknown.

This poses a challenge to segment the sensory stream into appropriate pieces that represent individual behaviours before any classification can be performed.

Other challenges Other challenges within the context of a smart home include:

Privacy One important issue with many monitoring systems such as the smart home is the privacy concerns of the inhabitant. However, there is often a trade-off between the privacy of the inhabitant and the benefits that the system can offer.

Lack of benchmark datasets There is still a lack of public benchmark datasets that enable proper evaluation of behaviour recognition algorithms. The majority of the datasets that are made available to the public are collected in a laboratory setting, and often these are constrained to a particular room location such as the kitchen or living room [29, 83, 134]. They do not provide a real representation of the inhabitant's daily behaviours within the home. Although there are other research groups that build their own smart homes and annotate the activities themselves [22, 137], they often utilise their own sensor hardware and the datasets either contain too little data or incomplete annotation about the inhabitant's activities. These make it difficult to test the algorithms on different datasets.

Lack of defined taxonomy of activities Besides the lack of benchmark datasets, there is also a lack of acceptable standard definitions of activities used by researchers in this field. For example, some researchers use the general term 'cooking', while others use specific terms such as 'making tea', 'preparing lunch', 'preparing snacks', etc. This makes the generalisation of the algorithm on these different datasets rather difficult unless researchers are willing to standardise the definitions of the activities when using these datasets.

1.4 Scope of Study

In this thesis, we restrict our scope of research directly to recognising the daily behaviours of an individual who is living independently in his own home. We start by discussing the form in which we expect the data to appear, since this affects which methods are suitable. Following this, we describe the aims and objectives of this thesis.

1.4.1 Tokens: A representation of sensor output

There are many types of sensor used to recognise the behaviours in a home. These sensors range from video cameras and microphones to body-worn sensors, and to touch and motion sensors. Obviously, the degree of preprocessing required to analyse the output of the different sensors, the amount of information that can be obtained, and the amount of privacy intrusion, vary depending on the types of sensors installed in the house. While these are all important considerations, and are an area of ongoing debate, they are not something that we wish to consider in this thesis. We therefore suggest that a useful abstraction from the sensor properties is to assume that activity in the house is presented to the learning system as a set of 'tokens,' which arrive in a sequence over time. Depending on the particular sensor, a token could be the direct representation of the current sensor states being triggered (i.e., kitchen light is turned off, heater is switched on, bathroom door is closed, etc.).

An example of the input that the smart home receives could then be in the form of a timestamp, location of the sensor, the object that the sensor is attached to and a sensor token (shown in Table 1.1), where the particular set of tokens B, X and Y would suggest that somebody spent some time in the evening watching television in their living room.

In this thesis, we will ignore the timestamp part of the data and simply consider the set of sensor tokens 'B, X, Y, H' in the example above. While this seems to be

Activation Date	*Activation Time*	*Location*	*Object Type*	*Sensor Token*
2011-05-15	20:22:19	living room	television	B
2011-05-15	20:22:47	living room	settee	X
2011-05-15	21:30:06	living room	light	Y
2011-05-15	21:48:33	kitchen	tap	H
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Table 1.1: Example of a sequence of tokens.

rather naïve, timing is not necessarily as helpful as it seems at first. For example, in the above example there is a relatively long gap between tokens 'X' and 'Y', but that is not because there is a behaviour change there, just that the particular behaviour can take an arbitrarily long time without sensor activation.

1.4.2 Aims and objectives of this thesis

This thesis zeroes in on the development of machine learning methods to find an approximation to the mapping between the set of sensor outputs and behaviours that is therefore suitable to be used in a smart home to monitor the inhabitant's behaviours, typically the elderly or cognitively impaired. The behaviours that we consider span from simple behaviours such as doing laundry, toileting and showering, where the behaviours are represented by sensor activations from a single sensor (e.g., sensor attached to the washing machine, toilet flush, shower tap), to complex behaviours such as preparing a meal and grooming that involve multiple sensors (e.g., sensor attached to cupboards, toaster, door, etc.).

In order to guide the development of machine learning methods in finding an approximation to the mapping outlined above, a set of objectives has been formulated which this thesis aims to fulfill. They will be revisited in the concluding chapter. There are three main objectives in this thesis. The first two objectives focus on the behaviour recognition problem, one using a supervised learning approach and the other on unsupervised learning. The third objective focuses on the sensor se-

lection problem. The following objectives are formulated based on the discussion in Sections 1.2 and 1.3.

1. *A supervised learning approach to behaviour recognition.* The first research objective in this thesis is to use a supervised learning approach for behaviour recognition. This resulting method will be able to:

    - learn patterns from labelled sensory stream
    - perform segmentation i.e., breaking the sensor sequence into appropriate pieces that represent individual behaviours, and perform behaviour classification
    - recognise variations in the behaviours
    - train on a limited number of training examples

2. *Unsupervised learning of activities.* The second research objective is to develop an unsupervised learning algorithm that learns from an unlabelled sensor data. This includes:

    - segmentation of the sensor stream into suitable patterns
    - identifying and learning patterns of variable lengths in a sensor stream, without prior human labelling
    - recognising variations in the patterns
    - providing labels to training data for a supervised algorithm

3. *Finding informative sensors.* The third research objective is to identify a set of informative sensors (i.e., the sensors that could provide the most information about the inhabitant's behaviours). This includes:

    - identifying a set of informative sensors

- gaining insights about the significance of the identified informative sensors among a set of trees
- recognising a variety of behaviours when trained on the set of identified informative sensors

In all the objectives stated above, we test our methods on a real smart home dataset (which is described further in the next section) to investigate the ability of these methods for performing behaviour recognition. The contents of Chapters 3, 4 and 5 reflect each of the research objectives described above.

1.5 The Smart Home Dataset

We used a real smart home dataset from the MIT PlaceLab [133] for most of the experiments conducted in this thesis. They collected data using a set of 77 state-change sensors that were installed in an apartment and collected data while a person lived in the apartment for a period of 16 days. The sensors were attached to household objects within the home such as the washing machine, toaster, door, etc. The dataset was annotated by the subject herself, meaning that there was a ground truth annotation of the dataset.

Although there were 77 sensors attached to the objects in the home, not many were used for behaviour recognition. Since our interest is to recognise the inhabitant's daily behaviours, we did not consider objects that were rarely used by the inhabitant. Thus, we did not include sensors that were activated less than 20 times throughout the 16 day period, resulting in a total of 24 sensors in the dataset. There were a total of 13 behaviours identified from this set of 24 sensors; these were doing/putting away laundry, dressing, grooming, washing/putting away dishes, preparing breakfast, preparing lunch, preparing dinner, preparing snack, preparing a beverage, toileting, showering, cleaning and putting away groceries.

1.5.1 Changes made to the ground truth data

Although the subject kept a record of her activities, there were notable omissions and inaccuracies in her annotations. We tried to correct these errors in the annotation. First, when an activity was interleaved with another activity, it was often labelled as one activity. For example, 'preparing meal' was often seen interleaved with either the 'toileting' or 'doing laundry' behaviours (see Figure 1.1(a)). Although, it was common to label it as one complete 'preparing meal' behaviour, having to train the instances of 'toileting' or 'doing laundry' as part of 'preparing meal' could result in a poor recognition system. Thus, we corrected the ground truth information to separate out the 'toileting' behaviour from 'preparing meal'.

Second, some observation sequences were labelled as 'toileting' although there was no instances of 'toileting' behaviour observed (see Figure 1.1(b)). To avoid confusion, we relabelled 'toileting' to 'grooming'. Third, there were some occasions when no activity labels were given to describe the sensor readings (see Figure 1.1(c)). For such, we corrected the annotation by labelling them based on our own knowledge of the sensor descriptions.

We have made a few assumptions to guide us when correcting the ground truth data:

1. Washing/putting away dishes. This behaviour is observed when the sensors on the kitchen tap, cupboards, drawers and/or dishwasher are activated.

2. Grooming or dressing. Since dressing behaviour is often interleaved with the grooming behaviour and there are no informative sensors that could discriminate between these behaviours, we combined these two behaviours as 'grooming/dressing'. In the dataset, when the sensor events on the medicine cupboards, sink, door or the light are activated, then these are labelled as 'grooming/dressing' (an example of this is shown in Figure 1.1(b)).

Ground Truth

Date	Time		Sensors	Labelling
	Activation	Deactivation		
8/4/2003	20:50:41	20:50:47	Door	Preparing Meal
	20:50:55	20:50:59	Light Switch	
	20:51:06	21:32:35	Light Switch	
	20:52:19	20:52:26	Freezer	
	20:52:51	20:54:53	Door	
	20:53:05	20:53:43	Cabinet	
	20:53:31	20:53:35	Refrigerator	
	20:53:33	20:53:39	Freezer	
	20:53:47	20:53:49	Drawer	
	20:54:30	21:16:11	Toilet Flush	
	20:54:41	20:54:48	Sink faucet - cold	
	20:54:56	21:09:01	Door	
	20:55:00	20:55:03	Freezer	
	20:55:05	20:55:10	Drawer	
	21:02:01	21:02:03	Freezer	
	21:02:35	21:02:37	Freezer	

corrected →

Corrected Ground Truth

Date	Time		Sensors	Labelling
	Activation	Deactivation		
8/4/2003	20:50:41	20:50:47	Door	Preparing Meal
	20:50:55	20:50:59	Light Switch	
	20:51:06	21:32:35	Light Switch	
	20:52:19	20:52:26	Freezer	
	20:52:51	20:54:53	Door	
	20:53:05	20:53:43	Cabinet	
	20:53:31	20:53:35	Refrigerator	
	20:53:33	20:53:39	Freezer	
	20:53:47	20:53:49	Drawer	
	20:54:30	21:16:11	Toilet Flush	Toileting
	20:54:41	20:54:48	Sink faucet - cold	
	20:54:56	21:09:01	Door	
	20:55:00	20:55:03	Freezer	Preparing Meal
	20:55:05	20:55:10	Drawer	
	21:02:01	21:02:03	Freezer	
	21:02:35	21:02:37	Freezer	

(a)

Ground Truth

Date	Time		Sensors	Labelling
	Activation	Deactivation		
9/4/2003	10:53:42	10:55:30	Door	Toileting
	10:54:58	10:55:07	Sink faucet - cold	
	10:55:01	10:55:10	Sink faucet - hot	
	10:55:25	15:42:12	Medicine cabinet	
	10:55:35	10:55:53	Door	
	10:55:36	10:55:38	Cabinet	

corrected →

Corrected Ground Truth

Date	Time		Sensors	Labelling
	Activation	Deactivation		
9/4/2003	10:53:42	10:55:30	Door	Grooming
	10:54:58	10:55:07	Sink faucet - cold	
	10:55:01	10:55:10	Sink faucet - hot	
	10:55:25	15:42:12	Medicine cabinet	
	10:55:35	10:55:53	Door	
	10:55:36	10:55:38	Cabinet	

(b)

Ground Truth

Date	Time		Sensors	Labelling
	Activation	Deactivation		
4/7/2003	18:48:54	20:38:53	Light Switch	
	18:49:03	18:49:09	Cabinet	
	18:49:16	18:49:28	Cabinet	
	18:49:37	18:55:36	Stove	
	18:49:42	18:49:58	Cabinet	
	18:50:31	18:50:34	Drawer	

corrected →

Corrected Ground Truth

Date	Time		Sensors	Labelling
	Activation	Deactivation		
4/7/2003	18:48:54	20:38:53	Light Switch	Preparing Meal
	18:49:03	18:49:09	Cabinet	
	18:49:16	18:49:28	Cabinet	
	18:49:37	18:55:36	Stove	
	18:49:42	18:49:58	Cabinet	
	18:50:31	18:50:34	Drawer	

(c)

Figure 1.1: Changes made to ground truth data. (a) The sensor sequence is labelled as 'preparing meal' although in between a 'toileting' behaviour is observed. We corrected the labelled to include the 'toileting' behaviour. (b) The sensor sequence is labelled as 'toileting' although no instances of 'toileting' behaviour observed in the sequence. We corrected the activity label to 'grooming'. (c) The sensor sequence is not labelled. We labelled them based on the knowledge gained from the sensor descriptions.

3. Toileting or showering. Since the main difference between 'toileting' and 'showering' is the sensor event on the toilet flush or shower tap, we combined these two behaviours as 'toileting/showering'. The main reason for doing so is to make two relatively similar behaviours and therefore harder (i.e., toileting/showering and grooming/dressing) where both pairs of behaviours shared some common sensors, such as sensors on the bathroom tap, bathroom door, etc. This is important to test how well the algorithm can distinguish between similar be-

haviours.

4. Doing/putting away laundry. This behaviour is observed when the sensor events on the kitchen door, laundry dryer and/or washing machine are activated. They indicate that the inhabitant is either doing or putting away laundry.

5. Preparing meal (breakfast, lunch, dinner), snack or beverage. Since there is no informative sensor that could help to distinguish which sensors are used in preparing breakfast, lunch, dinner, snack and beverages, we grouped all these behaviours and term this behaviour as 'preparing meal/snack/beverages'. This behaviour also includes reheating food, milk and preparing light snacks such as taking biscuits from the cabinet, etc.

6. Cleaning or putting away groceries. Since cleaning and putting away groceries behaviours do not occur frequently and there are no informative sensors that could discriminate between these behaviours, we grouped them together as 'cleaning/putting away groceries'.

Following the assumptions listed above, there are a total of 6 behaviours identified. Although there are only 6 behaviours that we consider in this dataset, there are a number of challenges when using this dataset for behaviour recognition. Firstly, it covers different time scales such as washing dishes and doing laundry which only occur approximately twice a week, while other behaviours repeat daily. Secondly, there are variations in the behaviours e.g., preparing meal/snack/beverages, where this behaviour may not necessarily involve the toaster or coffee machine each time it is seen. Adding in the fact that some behaviours share the same sensor events, e.g., both toileting/showering and grooming/dressing behaviours include the use of the bathroom tap for washing hands.

There are also other publicly available smart home datasets (such as CASAS [21], van Kasteren et al. [137], etc.), which could potentially be used to evaluate our

proposed methods. However, the majority of these datasets focus on basic activities (such as using telephone, eating, preparing meal, etc.), which are identified by a distinct set of sensors. This limits the ability of our algorithms to evaluate the variations in behaviours. Also, since the majority of the subjects were told to perform a sequence of activities rather than freely executing any activity, they did not provide a real world representation of the inhabitant's daily behaviours.

Since MIT PlaceLab dataset covers the majority of the challenges described in Section 1.3 and represents the daily behaviours of the inhabitant, it is therefore suitable to be used for behaviour recognition and hence is applied in most of the experiments in this thesis.

1.6 Thesis Overview

This thesis is organised into 6 chapters (including this chapter). Chapter 2 reviews previous works related to behaviour recognition. Chapters 3, 4 and 5 contain the main contributions of this thesis. An overview of each chapter is given below:

Chapter 2 begins by explaining the background and reviews previous work related to behaviour recognition. The review of the literature was conducted in order to develop an understanding of the different learning approaches proposed by researchers from the machine learning community and the various sensing methods used for behaviour recognition. This chapter ends with a detailed summary of methods used in existing smart home research projects and the approach that this thesis takes for its smart home project under the agenda of the Massey University Smart Environments (MUSE) group.

Chapter 3 introduces a method that can accurately perform segmentation and behaviour recognition simultaneously on the labelled sensor stream using a variable window length and a set of hidden Markov models (HMMs). The variable

window length has the ability to configure its window size based on the sensor observations, which ensures that the majority of the behaviours in the window are recognised. We test our algorithm by comparing it with fixed window length and how the effects of different window sizes affect the accuracy of the algorithm. We also investigate the amount of training data required to train the HMMs.

Chapter 4 addresses the problem of identifying and then learning patterns of variable length in a sensor stream without any prior human labelling. This includes both the segmentation of the sensor stream into suitable patterns and the identification of patterns when they are seen during testing. The resulting method is compared with other unsupervised learning algorithms and is validated against a baseline supervised method (described in Chapter 3). We also show that the proposed unsupervised method can be used as a way to provide labels to training data for a supervised algorithm, which can then be used to recognise behaviours.

Chapter 5 takes a step towards identifying a set of 'informative' sensors for behaviour recognition using information gain, which is modelled in a form of a decision tree. This chapter also introduces methods to gain insights about the significance of the informative sensors identified across a set of decision trees. The supervised and unsupervised methods (which are described in Chapters 3 and 4), are trained on the identified informative sensors for behaviour recognition. The results are then compared with the original baseline sensors used to recognise the inhabitant's behaviours.

Chapter 6 draws the conclusions and summarises the contributions of this thesis, and discusses the directions for future work.

Chapter 2

Literature Review on Behaviour Recognition

This chapter reviews recent work in behaviour recognition. The machine learning approaches for behaviour recognition are first described and then the types of sensor used to recognise behaviours are discussed. This chapter ends with an overview of work on existing smart home research projects, outlining their directions and methods used.

2.1 Introduction

The term 'Ubiquitous computing' was coined by Mark Weiser, with the vision that computers will be available to people in such a way that they are unaware of the presence of such computers, which are mostly embedded unobtrusively in objects such as clothes, cars, handheld devices, etc. [142]. To create such an environment involves bridging the challenges from artificial intelligence, computer vision, machine learning, wireless technologies, and human computer interfaces, so that the environment automatically learns, adapts and responds to the needs of the users.

For the environment to respond intelligently to the user's needs, it must first be able to recognise their behaviours based on the information sensed from the physical environment, which could be in the form of wearable sensors, microphones, video cameras, state-change sensors, etc. These have made behaviour recognition not only

a topic of interest, but also has drawn significant attention from the machine learning research community.

One application of behaviour recognition that supports people in their daily activities is the smart home, which has become a popular topic for research over the past 10 years. The smart home is an automated home, which uses a variety of sensors around the house to recognise the particular behaviours of the inhabitants. This can be useful to monitor and support the inhabitants, typically the elderly or those with diminished cognitive capabilities, in a variety of ways: watching for potential risks (e.g., making sure that the stove is not left unattended for too long), detecting any abnormality (e.g., turning on the heater in summer), reminding the inhabitant of tasks to be performed (e.g., taking medication), and many more.

Other applications of behaviour recognition include healthcare (e.g., monitoring the skills of medical practitioners during medical training [42, 55]), security and surveillance (e.g., detecting unusual events or suspicious behaviours in airports, car parks, subways, etc., [8, 20, 92]), industrial applications (e.g., analysing social patterns in organisations [108] and monitoring activities performed by the workers on assembly lines [68, 84, 129, 141]) and automation (e.g., automate heating, ventilation and air conditioning (HVAC) control [40, 98]).

2.2 Learning the Behaviours of the Inhabitant

There are various directions from which researchers have approached the problem of behaviour recognition, i.e., finding a mapping from a stream of sensor information to a sequence of recognised activities.

At one end of the spectrum, there are symbolic approaches that model the recognition process explicitly. For example, Augusto and Nugent [3] use the concept of Event-Condition-Action rules to detect potentially hazardous behaviours. The current state of the system is updated when an event occurs. If the condition of a

particular event is met, an action is triggered, which could either alert the carers or change the state of the system in the house.

Chen et al. [14] use event calculus for behaviour reasoning and provide assistance to the inhabitant when an abnormal behaviour is detected. The event calculus is formalised using Horn clauses, and the main components consists of events and fluents. An event is an observed action (e.g., add(teabag, mug)), while a fluent is a condition that changes over time (e.g., inside(teabag, mug)). Predicates define the relationship between entities (e.g., initiates(add(teabag, mug), inside(teabag, mug), 10am)). An activity is defined as a combination of multiple predicates. When the system sensed that the water has been boiled for a duration of time and the add(boiledwater, mug) action is not observed, then the system can take the appropriate action by issuing a reminder to the inhabitant. These approaches require a priori knowledge about the environment, its inhabitants, and the possible behaviours that might occur. The approaches are relatively easy to verify and validate, but often suffer from 'brittleness', as it is impossible to cover all possible behaviours.

At the other end of the spectrum, there are sub-symbolic approaches, which try to approximate the mapping with data mining and machine learning methods. One of the advantages of these approaches is that they reduce the need to rely on human effort to define the rules explicitly and once the machine learning algorithm has been trained, classification can be performed automatically without any human intervention. However, in order to train a machine learning algorithm, it needs training data and it is difficult to incorporate domain knowledge.

The main goal of machine learning is to train a classifier from a set of training data, which is then used for classification or prediction of new instances. In the following section, we describe the three commonly used machine learning approaches for behaviour recognition within the context of a smart home. The main distinction between the three approaches is whether the training data is labelled or not.

2.2.1 Supervised learning approach

In supervised learning, the algorithm is trained on labelled data. Prior literature demonstrates that supervised learning has been the primary approach for behaviour recognition. The decision tree [4, 80, 90], statistical classifiers such the naïve Bayes classifier [57, 133] and probabilistic graphical models such as the hidden Markov model and its variants [23, 29, 52, 103] are among the commonly used methods.

Among the earlier work that used decision tree for behaviour recognition is the work of Bao and Intille [4]. They train the decision tree to recognise human physical activities such as walking, sitting, etc based on accelerometers, which are attached to the body. In the work of Logan et al. [80], they apply the decision tree to recognise common activities such as eating, reading, using phone, etc. Maurer, Smailagic, Siewiorek and Deisher [90] use decision trees to identify which body locations are suitable to place the sensors for behaviour recognition. Although decision trees can be used to recognise human behaviours, they may not be suitable to recognise variations in the behaviour. For example, cooking does not always trigger the same sensor events. A decision tree might learn the mapping between some sensors and a particular behaviour, but might miss other sensors that correspond to the same behaviour.

Since there are variations in the behaviours, and the behaviours themselves are not directly observed, only the observations from the sensors, it is no surprise that probabilistic models such as the naïve Bayes classifier and hidden Markov models, have been the most popular method of recognising behaviours. The naïve Bayes classifier has shown a number of successes in machine learning applications such as text classification [76, 91] and spam filtering [120, 93]. Among the works that use the naïve Bayes classifier in behaviour recognition are those of Tapia, Intille and Larson [133] and Sarkar, Lee and Lee [57]. One problems with naïve Bayes is that it does not capture the sequential information of activity events. To achieve a good

recognition performance, temporal information needs to be incorporated into the classifier in order to capture the sequential ordering of sensor readings.

The hidden Markov model, and variants, which all model the temporal information, are popular methods for behaviour recognition. These models estimate the probabilities of unobserved events (i.e., inhabitant's behaviour) given a sequence of observable sensor readings. Given that a behaviour is decomposed into a series of individual activity events, many methods represent behaviours hierarchically and use a more complicated variant of the HMM such as the hierarchical HMM to recognise behaviours at varying levels of abstraction [29, 83, 103, 151]. In this model, the top level represents the individual behaviours to be recognised (e.g., cooking or making coffee) and the low level represents a set of individual activity events that arise from the sensor values. As this model attempts to recognise a complete model of behaviours, it requires more data for training and has a higher computational complexity, which often becomes intractable for learning.

A detailed description of these supervised learning methods and how they are used to recognise behaviours are further described in Section 3.2.

2.2.2 Semi-supervised learning approach

An algorithm in a semi-supervised approach to learning is often trained on a limited set of labelled examples, aided by the unlabelled ones. This is particularly useful in situations when there are limited number of labelled behaviour examples.

Many works reported in the literature apply transfer learning, which attempts to learn from labelled examples in a home and transfer the learned knowledge to a new home by learning from the unannotated ones [50, 135, 114, 155]. Another method is to use active learning, which attempts to find classes that have the lowest confidence, and actively engage the user to label them [131].

Other work includes using ensemble learning that trains multiple classifiers from

a set of limited labelled data. The classifiers are then learned from the unlabelled data, where a behaviour is classified when all the classifiers agree that it belongs to the same class or by majority voting [38]. However, this learning approach still relies on some labelled training data.

2.2.3 Unsupervised learning approach

In unsupervised learning, the algorithm learns from the unlabelled data by grouping data into similar classes that share some form of commonality. The unsupervised learning approach is appealing because it eliminates the need for data labelling and because of the fact that the algorithm does not require any prior knowledge about the behaviours in the home. In fact, unsupervised learning of behaviours in the home is not new and has already been explored by many researchers within the machine learning community.

The unsupervised learning approach is used to identify the set of sensors that are related to the particular behaviours of the inhabitant. One common method is to use the available information from the web, where the objects upon which the sensors are attached to in the home are mined to determine the potential behaviours associating with these objects [105, 109, 110, 149]. There are also works that focus on using standard unsupervised clustering methods such as the self-organising map [69, 70] or k-means clustering [44, 102] to determine if sequences of sensor readings are similar. However, relating these sequences of sensor readings to the activities is difficult as different people have their own ways of performing activities. An additional step is often required to map the resulting output from the clustering or the mined model to a supervised algorithm, which is then used to recognise behaviours.

Another variation to the methods pointed out above is topic model, which is a probabilistic model that has its root in natural language processing [47]. The idea behind topic models is to treat documents as mixtures of topics and each topic is

a probability distribution over words. In the work of Huỳnh, Fritz and Schiele [53], they use topic models to discover daily activity routines. They treat each document as mixture of activities and in each activity (e.g., preparing lunch), it consists of a probability distribution over activity events (e.g., washing vegetables, walking while carrying something, etc.). However, this method relies on an unsupervised algorithm such as the k-means clustering to generate a vocabulary of activities, where the k parameter needs to be known in advance.

In Section 4.2 of this thesis, a detailed description of these unsupervised methods is provided and how they learn from the unlabelled sensor stream. In Chapter 4 of this thesis, an unsupervised method that learns from the unlabelled sensor stream is presented.

Although many machine learning methods have been proposed in the literature, there are a number of factors that make comparison between these methods difficult. First, there are lack of benchmark datasets available for behaviour recognition. Second, the majority of the publicly available datasets are collected in a laboratory setting, which is constrained to a particular room location such as kitchen [29, 134]. Although these datasets could provide interesting information on how each activity is performed by different inhabitants, they do not provide a real world representation of the inhabitant's daily behaviours. Thirdly, researchers often employed different sensor hardware and collect their own proprietary data, and investigate their own research problem [22, 137, 133]. These make the comparison between methods difficult.

2.3 Sensors for Behaviour Recognition

In the previous section, the different machine learning approaches to behaviour recognition are described. Since sensors in some way represent the human behaviours,

knowing the types of sensors used is important in order to select the correct types of sensor for behaviour recognition.

The types of sensors that are described in this section are evaluated according to the criteria described in Section 1.3. Since sensors are to be fitted into the house, it is important to evaluate the level of privacy information collected about the inhabitant. The second criterion is the level of information that could be obtained from the sensors. As described in Section 1.2, an activity consists of a sequence of events. Thus, if the sensors in the home could capture all the possible events that make up an activity, this could provide important information about the inhabitant's behaviours.

In this section, the different types of sensors used to recognise human behaviours in a home setting are described. These sensors are evaluated based on the criteria described above. This review covers a wide range of sensors beginning with the most complex sensing methods used such as video cameras and microphones, followed by sensors with continuous output such as accelerometers and then simple sensors with discrete output, such as the RFID-based sensors, state-change sensors and motion sensors.

2.3.1 Video cameras and microphones

Information extracted from video cameras and microphones can provide a rich amount of information about human behaviours and their environment. Recognising behaviours from video and audio data poses interesting challenges in computer vision, and image and signal processing as well as machine learning.

Gao, Hauptmann, Bharucha and Wactlar [35] recognise eating events from visual data. They first identify the boundaries of individual persons by accumulating the segmented moving regions, and then recognise the faces of each person in the individual regions using a face detection algorithm. The head position is tracked, and the distance between the head and the hand movement is calculated, which is used

to train a hidden Markov model (HMM). The trained HMM is then used to recognise the individual eating events. Recognising dining activity can assist the caregivers in making sure that individuals consume the food or alert them when the individual has not been eating.

Although video data could provide high-dimensional information about the inhabitant's behaviours and the environment, extracting individual events from images is rather difficult. In order to recognise the inhabitant's behaviours in the home, Duong, Bui, Phung and Venkatesh [29] take advantage of the inhabitant's location in the room. Multiple cameras were installed, where each room is divided into equal partitions, with each partition contains some object of interests such as the sink, stove or the cupboard. When the camera detects the movement of the inhabitant occupying a particular partition in the room, events could be extracted that relate to the objects on that partition such as using the stove, taking food from the fridge, etc. The behaviour is recognised based on the partitions in the room that have been visited by the inhabitant.

Instead of using the location to extract individual events, the work of Tenorth, Bandouch and Beetz [134] extracts body poses directly from video data and calculates the trajectories of the body parts such as hand trajectories when opening the cupboard, which are then used to train a conditional random field (CRF). The trained CRF is used to recognise the behaviours of the inhabitant (e.g., setting the dinning table) based on the extracted motion events (e.g., taking a plate out of the cupboard, putting a plate on the table, etc.).

Other work includes detecting human falls from video data. Vishwakarma, Mandal and Sural [138], and Debard et al. [26] first apply a background subtraction method to extract the foreground region, which is then used to obtain the axis of the body. The information about the axis is used to detect falls. Hazelhoff, Han and de With [43] not only use the information obtained from the axis of the body, but also track the

head position to detect falls.

Chen, Zhang, Kam and Shue [13] used microphones to recognise behaviours in the bathroom in order to detect early signs of health deterioration or changes in behavioural patterns such as frequency of toilet use. Each sound such as showering, flushing, urination, etc., is first processed into frames and each frame is represented by a feature vector. A HMM is trained on the extracted features, which is then used to monitor behaviours in the bathroom.

Moncrieff et al. [97] used the signals obtained from microphones to identify contextual information of the inhabitant such as whether they are interacting with any household objects (e.g., turning on the stove, opening the fridge, etc.) or merely walking around the house. The contextual information gained from audio can help to identify any abnormality (e.g., when the stove is left unattended without the user present) and issue warnings to the caregivers or family members.

All these work above have shown that video cameras and microphones have advantages in recognising many different aspects of behaviour in a large environment and can thus provide high amount of information about individual behaviours. Although cameras and microphones could provide high amount of information about the inhabitant and the environment, they are often perceived as invasive, which is one common issues raised in the application of home monitoring. Also, the number of behaviours that can be recognised is limited and specific to the particular location in the home, e.g., behaviours occurring in the kitchen.

2.3.2 Accelerometers

One increasingly popular type of sensor, which is considered less intrusive compared to the use of video cameras or microphone is the accelerometer, which is a device used to detect the direction of the acceleration either along a single axis or multiple axes. Accelerometers are usually embedded within belts, wristbands, bracelets or

garments and are worn on different parts of the body such as the hips, pelvis, limb, etc. The term 'wearable sensors' is often used to refer to sensors that are worn on the body. Signals from accelerometers are either processed directly on the sensing board or transferred wirelessly to a computer. Features which are computed from the acceleration data are used to train a supervised learning method.

Many past works have demonstrated that the signals obtained from accelerometers (such as the pose, position and movement of an individual) are suitable to be used to recognise the human physical activities e.g., walking, running, sitting, standing and lying down [4, 81, 116]. Rather than using accelerometers to recognise the physical activities of a person, the work of Luštrek and Kaluža [85] takes advantage of the coordinates of the body such as the angle between adjacent body parts, inhabitant's locations in the environment and inhabitant's body part to detect falls. Li et al. [77] combine the use of accelerometers with gyroscopes, which measure the angular velocity to recognise whether a fall is intentional or not.

Since the information provided by the acceleration signals is based on body movements, the amount of information that can be used to recognise the inhabitant's daily behaviours such as cooking, showering, grooming, etc. is limited. Although there is work that attempts to recognise the daily behaviours of the inhabitant such as vacuuming, cleaning, bicycling, etc., [4], the range of daily behaviours that could be recognised is still restricted.

In order to increase the number of behaviours that could be recognised, the use of accelerometers is often combined with additional sensors that are attached to objects in the home [48]. For example, when a sensor attached to the cup is triggered, the information of the person's body state obtained from the acceleration data could provide information that the person is drinking while standing. The work of Stikic, Huỳnh, Van Laerhoven and Schiele [130] used sensors from acceleration and tagged objects to distinguish between the different housekeeping tasks (e.g., vacuuming and

brooming).

Although accelerometers have the advantage of recognising the position and body movement, one of the potential drawbacks of using accelerometers is the need to wear them on different locations on the body, which is obtrusive and irritating. This is a major concern in home monitoring, since typically the elderly can forget to wear them. An alternative to this is to attached sensors directly to the objects in the home, which is discussed next.

2.3.3 RFID-based sensors, state-change sensors and motion sensors

Another sensing approach that has been drawing significant attention in home monitoring is to attach sensors directly to the household objects such as microwave, toaster, shower tap, etc. These sensors are activated when the inhabitant performs their daily activities such as opening the microwave door, turning on the shower tap, etc. The main attraction of using this type of sensing approach is that it is less obtrusive since the inhabitant is no longer under the impression that they are being monitored.

One of the common sensing technologies is based on Radio Frequency IDentification (RFID), which uses radio waves to transfer data from RFID-tagged objects through a RFID tag reader. The tagged objects are transferred wirelessly to a computer system for processing. In a smart home, RFID tags are attached directly to the objects in the home and the tag reader, which is worn on the wrist, is used to detect the objects (e.g., opening the microwave door or turning the shower tap on). The information about the tagged object such as object identification, timestamps, location of the sensors, etc., is used to infer the behaviours of the inhabitant in the home. The work in this area was considered in various studies to recognise the inhabitant's daily behaviours in the home [107, 111, 127, 147].

Since the RFID-based sensing technology relies heavily on a tag reader to trigger

the tagged objects, Tapia, Intille and Larson [133] introduced a sensing approach based on state-change sensors, which collects information about the objects used in the same way as the RFID-based sensor, except that the sensors are triggered directly based on the movement of the objects without relying on a tag reader. The sensor is connected to a data collection board and wired to the object. Contact with the object is measured through the use of sensor components such as reed and magnetic switches. Sensors are activated based on the movement of objects when the inhabitant performs their daily activities. There has been a lot of work that recognises behaviours from state-change sensors. However, most of these work address their own specific problem. For example, the work of Hu and Yang [49] recognises interleaving behaviours while the work of Sarkar, Lee and Lee [57] addresses the zero-frequency problem, i.e., sensors that are presented to the learning algorithm during testing but not during training. Zheng, Hu and Yang [155] recognise similarity between activities in different homes.

There is also work that used motion sensors to recognise the inhabitant's behaviours in the home. The motion sensor is based on the passive infrared (PIR) sensor, which measures changes in the heat emitted by surrounding objects. These sensors are mounted on the ceiling and motion is detected when a human is within the sensor range. Motion sensors are used to detect the inhabitant's location in the home and are often used in combination with other sensors such as the RFID-based sensors or state-change sensors for behaviour recognition [80, 126].

The review of the different sensing approaches above has demonstrated that video cameras and microphones are often used to recognise interactions of multiple inhabitants in a large area and fall detection. Accelerometers, which are worn on different parts of the body are used to recognise human physical activities. Sensors based on objects used, on the other hand are commonly used to recognise daily human behaviours.

In terms of privacy concerns, video cameras and microphones are considered as invasive, although they provide lots of information about the inhabitant in the home. Accelerometers are less invasive but they provide too little information about the inhabitant's daily behaviours and the need to wear the sensors around the body makes them unsuitable for home monitoring. Sensors that are based on objects used in the home, which is also the focus of this thesis, are unobtrusive and thus an ideal approach for behaviour recognition in the smart home. However, one disadvantage of using sensors based on objects used is the difficulty in differentiating the identity of the person activating the sensors typically in a home with multiple inhabitants.

Table 2.1 summarises the different types of sensing approaches used for behaviour recognition in the context of privacy, the level of information obtained from the sensor and the type of behaviours that are to be recognised.

	Types of Sensors		
	Video Cameras and Microphones	**Accelerometers**	**RFID-based, State-change and Motion Sensors**
Average No. of Sensors Used	< 5 [35, 134]	≥ 5 and ≤ 10 [4, 130]	> 10 [137, 133]
Level of Information Inferred from Sensors	High-level of information, but limited to the specific room location	Limited information, solely based on body position and movement	Large number of complex combinations of sensor events
Level of Privacy	Obtrusive	Less Obtrusive	Unobtrusive
Types of Behaviour	Human physical activities Fall detection Human daily behaviours (limited)		Human daily behaviours

Table 2.1: Types of sensing approaches for behaviour recognition

2.4 Research in Smart Homes

In this section, the various university-based smart home projects are described. The review on these smart home projects is focused on the development of machine learning methods to learn and support the inhabitant's activities acquired through sensors. These projects include the iDorm [40], Adaptive Home [98], Georgia Tech Aware Home [62], Gator Tech Smart House [46], MIT PlaceLab [133], MavHome [22, 151] and CASAS [21]. All these research work aim to recognise the inhabitant's behaviours

from the observations acquired through sensors, but from there the directions and methods differ markedly.

The iDorm [40] and Adaptive Home [98] are among the earlier prototype university-based research projects with the aim to maximise the comfort of the inhabitant, while minimising the cost of electrical consumption by automatically controlling the lighting, and HVAC of the home.

The iDorm [40], short for Intelligent Dormitory, aims for a system that can automatically learn the inhabitant's behaviours and adapt to their needs. Although the setup of the iDorm is in a dormitory, its usage is similar to a smart home as behaviours such as sleeping, preparing meal or working can be monitored. The iDorm focuses on using intelligent embedded agents to learn the inhabitant's behaviours from the sensor inputs. The system uses a fuzzy-logic approach to learn the rules regarding the inhabitant's needs. The system runs in an on-line mode, and adapts to inhabitant's ongoing behaviours by modifying the rules either directly by the inhabitant themselves or learns from the existing rules. These rules are then used to automatically control the actuators in the room, i.e., turning on/off the fan heater, opening or closing the blinds, etc.

The Adaptive Home [98] learns the inhabitant's behavioural patterns based on sensors information obtained from objects used and other contextual information such as room temperature and the location of the inhabitant in the house. The backbone of the system is reinforcement learning, where a learning agent learns the environment and the inhabitant's behaviours, and decides which action was being performed. The action performed by the learning agent is either rewarded when the output is correct or punished otherwise. The implementation of reinforcement learning in the Adaptive Home enables the agent to learn the inhabitant's behavioural patterns and control the environment without any human intervention to intelligently automate the lights, water heating, and HVAC control. It also employs neural networks for predicting the

next room location of the inhabitant so that control could be raised before the room becomes occupied.

There are also works that consider the mapping between the hardware (sensors and actuators) and the application software [62, 46]. The Georgia Tech Aware Home [62], is one example. They attempt to develop a distributed and network computing infrastructure to support the inhabitant's activities. Their work explore various implementations of sensing approaches (the majority of which have been discussed in Section 2.3), such as motion recognition and location tracking from video, and recognition through floor sensors.

Another example is the Gator Tech Smart House [46] that attempts to create a scalable off-the-shelf assistive technology that provides an assistive environment and remote monitoring of electrical appliances in the home, e.g., smart closet, which has the ability to make suggestion on the types of clothing to wear based on weather conditions, smart mailbox that automatically notifies the inhabitant when the mailbox senses mail arrival, smart laundry that notifies the inhabitant when to do laundry, amongst others. The main focus of both Georgia Tech Aware Home [62] and Gator Tech Smart House [46] is to bridge the gaps between the hardware technologies and software implementation, and how the hardware infrastructure could support the development of machine learning approaches to learning the inhabitant's behaviours.

Similar to the goals of the smart home research at Massey University [89], there are also other smart home research projects that recognise the inhabitant's behaviours from unobtrusive sensors such as the MIT PlaceLab [133], MavHome [22] and CASAS [21]. The MIT PlaceLab [133] introduces a system which is based on a set of state-change sensors that were placed in two different apartments with real people living in them. These sensors are attached to objects in the home and are activated based on the movement of the objects when the inhabitant carries out their daily activities. In their work, they employ a supervised learning approach based on the naïve Bayes

classifier to recognise the inhabitant's behaviours.

The MavHome [22] is a multidisciplinary university-based smart home project that uses the integration of machine learning methods and multiagent technologies to monitor, learn and predict the inhabitant's daily activities from motion and lighting sensors. The MavHome architecture is decomposed into many sub-agents, where each sub-agent is responsible for different subtasks that control the devices in the home, e.g., automating the blinds. Executing each subtask involves gathering of information obtained from the sensors, predicting about the inhabitant's behaviours and then executing the action by controlling the actuators in the home. The MavHome research project includes the integration and implementation of various machine learning methods. One of their prediction algorithms, which is based on LZ78 data compression, calculates the probability of each action occurring based on the history of action sequences, and makes predictions based on the action that achieves the highest probability.

Another smart home research project that is similar to MavHome is the CASAS research project [21] at Washington State University. The CASAS is a 3-bedrooms smart apartment that uses a variety of sensors (such as motion sensors, switch sensors, etc.) to collect information about the inhabitant's behaviours. Their main goal is to create a variety of datasets that cover basic activities (e.g., hand washing, meal preparation, cleaning, etc.) that were performed by undergraduate students. These datasets are annotated and available to the public where researchers can use to test their algorithms.

In similarity to the MIT PlaceLab, MavHome and CASAS, the Massey University Smart Environment (MUSE) research project [89] also focuses on using unobtrusive sensors to identify the behaviours of the inhabitant, and monitor potential abnormality. The development of the MUSE project is still at its infancy stage, i.e., at the stage of sensor installation in a home of an elderly blind man, who has volunteered

himself as a testbed for this project. The work reported in this thesis (the supervised and unsupervised learning approaches in Chapters 3 and 4) will be implemented as a stepping stone towards the behaviour recognition in this smart home project. Since the MUSE project is still at the design stage, the methods described in Chapter 5 will also be employed to help guiding the sensor selection in this project.

2.5 Summary

This chapter has demonstrated that behaviour recognition has a wide range of applications and has been drawing significant attention from the machine research community. Generally, all past works demonstrated that behaviours can be inferred from a sequence of sensor observations output from the house and that a supervised learning method is often used to recognise the behaviours, where the algorithm is trained from a set of labelled sensor data.

This chapter also reviewed the range of sensing methods that could be used to recognise the inhabitant's behaviours from the home, ranging from cameras and microphones to wearables, motion and objects-used sensors. The choice of which sensing method to use depends upon what the aim of the smart home is, the level of information obtained from the sensors and the level of privacy intrusion.

There have been a lot of smart home research projects reported in the literature and some of them have been reviewed. Since each of these projects has its own directions and aims, comparisons between the different smart home projects are very difficult. However, the literature on these smart home projects has demonstrated the need for further machine learning methods to learn the behaviours of the inhabitant, and there is still far more work that needs to be done in this area before the smart home is able to accurately monitor the inhabitant's behaviours.

As this will become clearer in later chapters, using state-change sensors for behaviour recognition raises one problem, i.e., the series of sensor readings output from

the home is unsegmented, where the start and end boundary of a behaviour is unknown. Very few methods described in the literature perform segmentation and behaviour recognition simultaneously on the sensor stream. Addressing this is important in real implementations as the sensor stream needs to be segmented before any classification is performed.

The literature has demonstrated that supervised learning has been the primary approach for behaviour recognition. However, it needs labelled data. Although there are unsupervised methods proposed in the literature, they do not satisfy the criteria that are considered in this thesis, which are described Section 1.4.2.

In the next chapter, we describe one particular behaviour recognition problem that still deserves further attention and our proposed method to approach the problem.

Chapter 3

A Supervised Learning Approach to Behaviour Recognition

This chapter describes a supervised learning approach to behaviour recognition and the segmentation of the sensor stream. The behaviour recognition problem is first described, and then an overview of methods that have been proposed in the literature is provided. An approach to the problem based on simultaneous segmentation and recognition is presented, together with experiments showing the effectiveness of the proposed method for recognising human behaviour.

3.1 Problem Description

As described in Section 1.4.1, the smart home uses sensors to collect information about the inhabitant's activities. Given that we have a sequence of tokens obtained from the sensors, the question is how to recognise behaviours. The challenges in this task are that behaviours are rarely identical on each use; the order in which the individual components happen can change; and components can be present or absent at different times (for example, making a cup of tea may involve milk, or may not, and the milk could be added before or after the hot water). Adding in the fact that the activities are not directly observed, and that sensor observations are themselves intrinsically noisy, the tokens represent only a very partial picture of what the inhabitant is doing.

One common approach to recognising behaviours is to use a supervised learning method. In supervised learning, the algorithm is presented with a set of labelled behaviours that are to be learned. The algorithm is then trained to minimise some error norm between the predicted output of the classifier and the target labels. In this way it should be able to generalise to other, previously unseen, inputs. The benefits of this are a clear delimitation of the problem, and access to many algorithms in machine learning, including decision trees, statistical classifiers such as the naïve Bayes classifier, and graphical models such as the hidden Markov model (HMM). And indeed, there are several examples of these methods implemented in the various smart homes that have been reported in the literature, such as [136, 133, 151].

However, as described in Section 1.3, the sensory stream in a smart home consists of an unending series of sensor readings, which poses an additional challenge to segment the sensory stream in order to determine the start and end points of an activity boundary, before any classification can be performed on the sensor sequences that represent individual behaviours.

Most current approaches assume that the activities have been segmented, or use a fixed window length to partition the input stream. As each behaviour can be described by different numbers of sensor readings, it is inappropriate to rely on a fixed window length, since it is unlikely that all the sequences in the window belong to one behaviour. We want to ensure that other behaviours in the window are also recognised. We approach this problem by proposing a method using a set of hidden Markov models that each recognises a behaviour and compete to explain the current observations, and a method that has the ability to self-determine the window size based on the sensory data.

We first provide an overview of the supervised learning algorithms and describe how they are used to recognise behaviours. We then review methods that are used for activity segmentation. Following this, we introduce our method, which can reliably

detect and segment the sensor stream into behaviours. We use a real smart home dataset to demonstrate the efficacy of our algorithm.

3.2 Relevant Literature

The literature review in this section focuses on supervised learning methods for behaviour recognition. All these methods learn from a set of labelled training data, which is described by a set of features and class labels. The learned model is then used to classify further examples into one of the given set of classes.

In Section 2.2.1, some supervised learning methods were briefly discussed. A detailed description of these methods is given in this section. The first method is to build a tree structure that maps the set of features and the class labels to generate a series of decision rules, which could then be used for classification. The second method is the naïve Bayes classifier, which is a probabilistic classifier based on Bayes' theorem, where the features are assumed to be independent given the class variable. In the third, we look at the hidden Markov model (HMM), which is a probabilistic graphical model where the model is assumed to be a Markov process with unobserved states.

Since the sensor stream consists of an unending series of sensor readings, this section also reviews the methods used to segment the sensor sequence into appropriate pieces that represent individual behaviours. The basic concepts covered in this section will also be used in later chapters of this thesis.

3.2.1 Decision tree

The central idea of a decision tree is to map the set of features and the class labels, with the aim to generate a series of decision rules, which could then be used to classify further examples into one of the classes [94]. A decision tree consists of a set of *nodes*, which correspond to the features, *edges* which descend from the nodes represent the

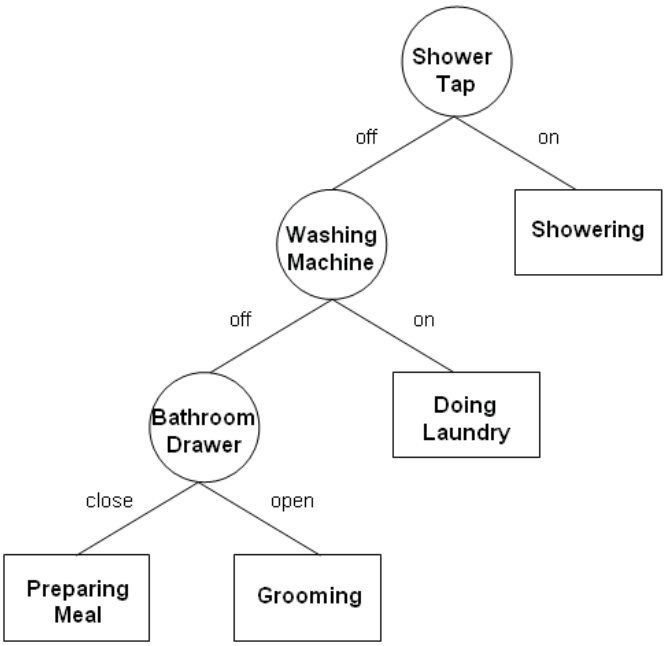


Figure 3.1: A graphical representation of a decision tree. The nodes represent the features to be tested, edges represent the feature values and rectangles represent the class labels.

feature values, and *leaves* represent the classification decision. Figure 3.1 shows an example of a decision tree.

Most decision tree learning algorithms employ a greedy approach, i.e., recursively partition the dataset into smaller subsets by choosing the most informative feature at each step starting at the root of the tree [87, 112]. If all the instances of the feature belong to the same class, then a leaf node is created, otherwise a child node is created for each value of that feature. The process is then repeated for each child node. The problem is how to choose which feature to test at each node in the tree. One common way is to measure the information gain of a given feature, which tells us how well the feature separates out the examples in the dataset according to their class labels [95, 112]. The feature with the highest information gain will therefore be chosen for splitting the tree node.

There has been a lot of work done in using decision trees to recognise human physical activities (such as walking, running, sitting, standing, etc.) based on ac-

celerometers, which are usually attached to the body at locations such as the limb, hip, wrist, etc. [4, 116]. Pärkkä, Cluitmans and Ermes [106] attempt to improve the classifier accuracy by enabling the inhabitant to provide feedback to the recognition system whenever it misclassifies the behaviour. They introduce a personalisation algorithm that enables the inhabitant to provide the corrected example of that activity and the decision tree re-learns from this newly added information.

However, one problem with accelerometers is the need to wear them on the body, which makes the inhabitant feels uncomfortable. Wang et al. [140] turn the problem around by attaching the accelerometers directly on the household objects in the home (e.g. cup, phone, etc.). Acceleration signals are detected from the movements of the objects, which are then used to train a decision tree. There is work done that attempts to recognise activities based on state-change sensors, which are attached to household objects [55, 80]. The work of Hong et al. [48] trains the decision tree using both accelerometers and state-change sensors to improve the recognition accuracy. Some works also progressed towards using the decision tree to identify which particular locations of the body are suitable to place the accelerometers in order to achieve a higher recognition accuracy [4, 90].

Many findings reported in the literature demonstrate that decision trees can be used to recognise human behaviours. The benefits of using a decision tree are that: (1) it can handle both continuous and discrete features, (2) the logical structure in the decision tree makes it easy to understand and (3) it can provide an indication of which features are most important for behaviour recognition. There is also drawback to using a decision tree. Decision tree may have the difficulty in distinguishing behaviours (e.g., cooking and making tea) that share similar sensor activations (e.g., sensors on the cupboard, fridge, kitchen tap, etc.) [4].

3.2.2 Naïve Bayes classifier

In general, a classification problem is to predict class Y given a set of features X_i; $i = 1, 2, \ldots, k$. By applying Bayes' rule, the probability of class Y is

$$P(Y|X_1, X_2, \ldots, X_k) = \frac{P(X_1, X_2, \ldots, X_k|Y)\, P(Y)}{P(X_1, X_2, \ldots, X_k)} \tag{3.1}$$

where $P(X_1, X_2, \ldots, X_k|Y)$ and $P(Y)$ can be calculated from the data during the training of the classifier. For example, we want to build a classifier to determine if a patient shows early signs of health deterioration based on some simple features such as gender (female or male), shortness of breath (yes or no) and whether the patient takes any medication (yes or no). From a given data set, we can calculate $P(Y = \textit{health deterioration})$ by dividing the number of patients that show early sign of health deterioration by the total number of patients in the data set. We can then calculate other probabilities that we are interested in from the data set such as $P(gender = female, shortness\ of\ breath = yes, medication = yes \mid Y = \textit{health deterioration})$ and $P(gender = female, shortness\ of\ breath = yes, medication = yes)$. Following Equation 3.1, we can calculate the probability that a female patient who is on medication and suffers from shortness of breath shows an early sign of health deterioration.

The computation in a Bayes classifier is simple and straight-forward. However, as the number of features increases it becomes impractical to compute the probabilities [95]. Let us consider building a boolean class Bayes classifier with n boolean features. In this case, for each feature, we will need to calculate 2^n probabilities. Since for a particular class, the sum over all probabilities for a given feature must be one, only $2^n - 1$ calculations are necessary. Given a boolean class we need to calculate a total of $2(2^n - 1)$ probabilities, which correspond to each of the distinct features. If

n is a large value e.g., 20 boolean features then we will need to compute more than 10 million probabilities, which is impractical in real world implementations.

One solution to this problem is to assume all features $(X_1, X_2 \ldots, X_k)$ to be independent of each other, given the class Y. The resulting model is called the naïve Bayes classifier. The naïvete in the classifier is that all the features are assumed to be independent of each other and so the computation of $P(X_1, X_2, \ldots, X_k|Y_i)$ can be simplified as $\prod_{j=1}^{k} P(X_j|Y_i)$. The prediction of class Y, given a set of features $X_1, X_2, \ldots, X_k$ is therefore to select the class that maximises the posterior probability, i.e.,

$$Y = \arg\max_{y_i} \; P(Y_i) \prod_{j=1}^{k} P(X_j|Y_i)$$

Figure 3.2 shows an example of a generic naïve Bayes classifier as a graphical model. The class node represents the behaviour, which is the parent to its features (e.g., sensors that are attached to the toaster, television, etc.).

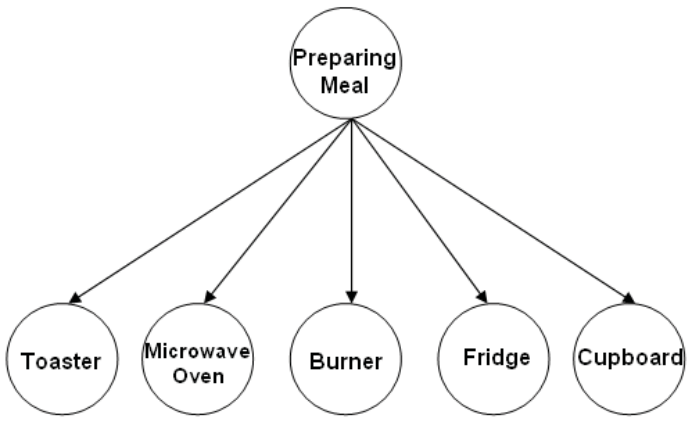


Figure 3.2: A graphical representation of a naïve Bayes classifier. The features are the sensors attached to the toaster, microwave oven, burner, etc., and are independent given the class 'preparing meal' behaviour.

In comparison to the decision tree described in Section 3.2.1, the naïve Bayes classifier does not need to perform an explicit search through the training data to test each sensor at every tree node, but instead, counts the frequency of sensor combinations

within the training examples.

Among the earlier work that used the naïve Bayes classifier to recognise behaviours of the inhabitant is the work of Tapia, Intille and Larson [133]. They extend the naïve Bayes classifier to incorporate temporal relationships of the sensor activations, such as whether token 'a' activated within some time interval and whether token 'a' is activated before token 'b'. In order to recognise behaviours, they used a set of feature windows, each represents one behaviour and the length of each feature window is based on the average duration that the inhabitant took to carry out the activity. The probability for the current activity is calculated by shifting the feature window over the sensor sequences. The probability reaches the maximum when the window aligns with the duration of the activity from the sensor readings.

Since some behaviours are rare, this results in the distribution of the sensors not being consistent among the behaviour classes. In order to accurately learn the behaviour classes, the work of Sarkar, Lee and Lee [57] incorporates a smoothing technique by discounting the probabilities of frequently seen sensors and giving some probabilities to unseen sensors when training the naïve Bayes classifier. Long, Yin and Aarts [81] attempt to use principal components analysis (PCA) to first reduce the number of features that will affect the performance classifier before training it. The work of Yang, Lee and Choi [150] introduced a penalise function where the trained classifier is penalised according to mismatch of time, and mismatch between learned model and the observed actions of an activity.

Although the strong independence assumption in the naïve Bayes classifier makes it a tractable approach for learning (i.e., by considering each token in each behaviour separately), this assumption may not hold in real world applications since correlations among features are common. The smart home is one example where some sensors are correlated. For example, sensors attached to kettle and kitchen tap are correlated since making tea involves these two objects. The correlations among sensors introduce

dependencies and as such reduce the influence of other features, which can then affect the overall performance of the algorithm [74, 118].

One problem with naïve Bayes is that it does not take into account the sequential ordering of sensor readings. Thus, to achieve a good recognition performance, temporal information needs to be encoded in the classifier (which can be seen in the work of Tapia, Intille and Larson [133]). Another method that models the temporal information directly into the model is the hidden Markov model (HMM), which is described next.

3.2.3 The Hidden Markov model

The Hidden Markov Model (HMM) is a probabilistic graphical model that uses a set of hidden states to classify a sequence of observations over time [113]. These hidden states evolve over time and are not directly observable. Generally, in the smart home context, the hidden states represent the behaviours of the inhabitant and the observations are the sensor readings. The goal is to infer the hidden states given a sequence of sensor readings. For example, the observations could be that the laundry door is opened and the washing machine is running, where the possible behaviour could be that someone is doing laundry.

HMMs admit tractable algorithms for learning and prediction, and are very commonly used for learning behaviours from labelled data [16, 23, 52]. Figure 3.3 shows a simple representation of a HMM where the nodes represent the variables and the edges represent the conditional dependencies between the nodes. The relationship between the hidden states and the observations in the HMM is similar to the naïve Bayes classifier described in Section 3.2.2. The difference is that the HMM incorporates time dependency into the model by using a transitional probability distribution between the states. This probability distribution represents the transition from one hidden state S_i at time t ($q_t = S_i$), to another hidden state S_j at time $t+1$ ($q_{t+1} = S_j$).

The transition between states allows sequences of behaviours to be modelled.

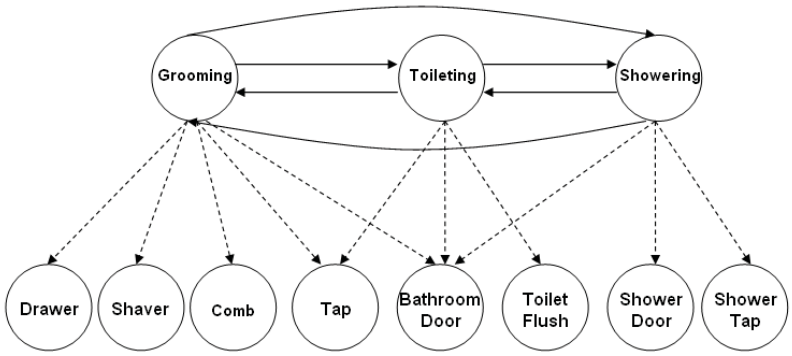


Figure 3.3: The graphical representation of a Hidden Markov Model. The nodes represent the variables (top three nodes are the hidden states and the bottom eight nodes are the observations) and the edges (arrows) represent the conditional dependencies: the solid line represents the conditional dependencies between states and the dashed line represents the conditional dependencies of the observation on the hidden states.

We follow the notation used in Rabiner [113], which uses $\lambda = (N, M, A, B, \pi)$ to indicate the complete parameter set of a model. The description of each parameter is as follows:

N – the number of states.

M – the number of possible observations.

A – the state transition probability distribution $A = \{a_{ij}\}\,;\ i, j = 1, 2, \ldots, N$ where $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$ is the probability of transition from state S_i at time t ($q_t = S_i$) to state S_j at time $t + 1$ ($q_{t+1} = S_j$).

B – the observation probability distribution $B = \{b_i(k)\}\,; i = 1, 2, \ldots, N$ and $k = 1, 2, \ldots, M$ where $b_i(k) = P(O_t = k | q_t = S_i)$ is the probability of observing observation k given that the current state is S_i.

π – the initial state distribution $\pi = \pi_i$, the probability of state S_i being the first state.

The complete specification of the HMM requires the specification of N and M, and the probabilities of A, B and π. However, A, B and π can be learned.

The HMM follows the Markov assignments [113] i.e., the probability of transition to the next state S_j at time $t+1$ ($q_{t+1} = S_j$) depends only on the current state S_i at time t ($q_t = S_i$) and the observation O_t depends only on the state $q_t = S_i$. These properties make the HMM a suitable model for behaviour recognition. Given a sequence of observable sensor readings, HMMs can identify the sequence of behaviours that gave rise to them. For example, when the bathroom door is closed and the shower tap is running, these give rise to the 'showering' behaviour. When the sensor on the shaver is turned on, the behaviour changes to 'grooming'. The changes between behaviours are represented by the transition between the states. Thus when using the HMM for this example, the probability of grooming behaviour at current time is determined only by the previous behaviour, which is showering. Given the complete specification of the HMM, there are three problems that need to be solved i.e., (1) evaluation, (2) decoding, and (3) learning. These are described next.

Problem 1: Evaluation

Given a sequence of observations $O_1, O_2, \ldots, O_T$ and a model λ, the first problem is to find an efficient way to compute the probability that this sequence of observations was produced by λ, i.e., $P(O_1, O_2, \ldots, O_T|\lambda)$. This can be achieved by using the forward algorithm [113].

The computation of the conditional probability in the forward algorithm involves 3 steps: (1) initialisation, (2) induction and (3) termination. Let $\alpha_t(i) = P(O_1, O_2, \ldots, O_t,\ q_t = S_i|\lambda)$, the probability that we have seen the first t observations and ended up in state S_i.

The initialisation step computes the joint probability of state S_i and the initial observation O_1:

$$\alpha_1(i) \;\; = \;\; \pi_i \, b_i(O_1) \, ; \;\; i = 1, 2, \ldots, N \, , \tag{3.2}$$

where b_i is probability of observing observation O_1 given that current state is S_i. The induction step involves the computation of the forward variable $\alpha_{t+1}(j)$. As illustrated in Figure 3.4, $\alpha_{t+1}(j)$ is the probability of reaching state S_j at time $t+1$ from any of the N states S_i at time t, where

$$\alpha_{t+1}(j) \;\; = \;\; \sum_{i=1}^{N} \alpha_t(i) \, a_{ij} \, b_j(O_{t+1}) \, ; \;\; t = 1, 2, \ldots, T-1, \;\; j = 1, 2, \ldots, N \tag{3.3}$$

The product of $\alpha_t(i) \, a_{ij}$ is the probability of the joint event $P(O_1, O_2, \ldots, O_t, \, q_{t+1} = S_j)$ that $O_1, O_2, \ldots, O_t$ are observed and state S_j is reached at time $t+1$ from state S_i at time t. The probability of S_j at time $t+1$ with all the previous partial observations is thus the summation of $\alpha_t(i) \, a_{ij}$ over all the N states at time t. Multiplying this with the probability $b_j(O_{t+1})$ gives the value of $\alpha_{t+1}(j)$. The computation iterates for all the observations $t = 1, 2, \ldots, T-1$ following Equation 3.3.

The termination step is to calculate the probability of the observations given the model $P(O|\lambda)$, which is just the sum of all $\alpha_T(i)$:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i). \tag{3.4}$$

Problem 2: Decoding

The decoding problem can be viewed as a way to find a single state sequence (path) that best describes the observation sequence i.e., to maximise $P(Q, O|\lambda)$. This can be achieved using the Viterbi algorithm [59].

In order to find the single best state sequence $Q = q_1, q_2, \ldots, q_T$ for the given

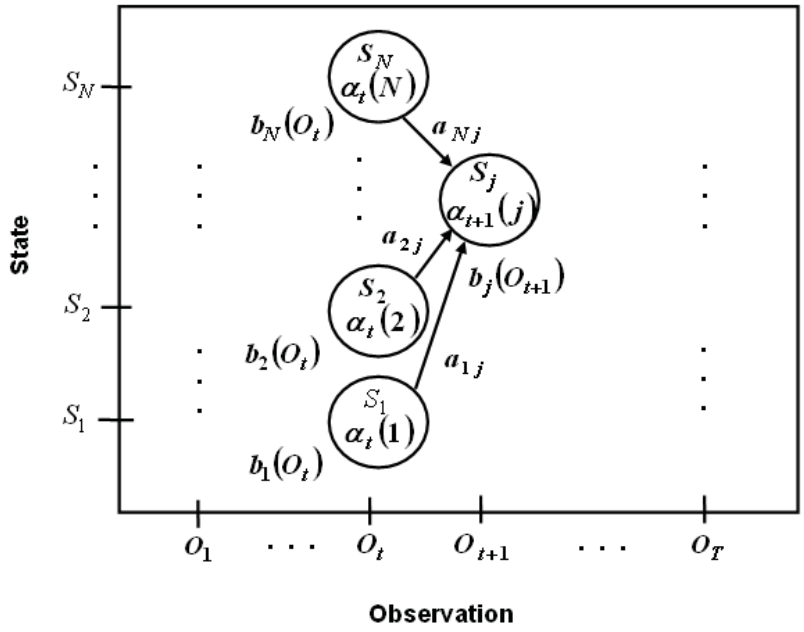


Figure 3.4: Illustration of the forward variable $\alpha_{t+1}(j)$ and the computation of $\alpha_t(i)$ upon the lattice structure of observations O_t and states S_i. This figure is modified from [113] to illustrate how the forward variable is computed in the context of this thesis.

observation sequence $O = O_1, O_2, \ldots, O_T$, let

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t | \lambda)$$

where $\delta_t(i)$ is the highest probability along a single path for the first t observations and ends in state S_i at time t. By induction, we can calculate

$$\delta_{t+1}(j) = \max_i \; \delta_t(i)\, a_{ij}\, b_j(O_{t+1}). \tag{3.5}$$

In order to retrieve the state sequence, an array $\psi_t(j)$ is used to keep track of the argument that maximises Equation 3.5. The Viterbi algorithm involves 4 steps: (1) initialisation, (2) recursion, (3) termination and (4) path backtracking. The initialisation step of the Viterbi algorithm is:

$$\delta_1(i) \;=\; \pi_i \, b_i(O_1) \,; \quad i = 1, 2, \ldots, N$$

$$\psi_1(i) \;=\; 0$$

The recursion step of the algorithm is to recursively compute $\delta_t(j)$ and $\psi_t(j)$ for all the N states, which is

$$\delta_t(j) \;=\; \max_{1 \le i \le N} \, \delta_{t-1}(i) \, a_{ij} \, b_j(O_t)$$

$$\psi_t(j) \;=\; \arg\max_{1 \le i \le N} \, \delta_{t-1}(i) \, a_{ij}$$

where $j = 1, 2, \ldots, N$ and $t = 2, 3, \ldots, T$.

The termination step of the Viterbi algorithm computes

$$p^* \;=\; \max_{1 \le i \le N} \, \delta_T(i)$$

$$q_T^* \;=\; \arg\max_{1 \le i \le N} \, \delta_T(i)$$

where p^* is the highest probability of that most probable path and q_T^* is the most likely state at time T for the given path. Thus, the most probable path can then be traced back using the following equation

$$q_t^* \;=\; \psi_{t+1} \left(q_{t+1}^* \right) ; \quad t = T-1, T-2, \ldots, 1$$

Problem 3: Learning

The third problem deals with adjusting the model parameters in order to maximise the probability of the observation sequence given the model $P(O|\lambda)$. In fact, there is no analytical solution for the model that maximises the probability of the observation sequence, even given any finite training data [113].

However, we can select a model that locally maximises $P(O|\lambda)$ through an iterative process. One of the algorithms to address this is the Expectation-Maximisation (EM) algorithm [27]. The idea behind the EM algorithm is to find the maximum likelihood estimates of the parameters using an unknown latent variable and then maximise the function over those variables [87].

Let $\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda = (A, B, \pi))$ the probability of being in state S_i at time t and state S_j at time $t+1$, given the observation sequence and the model. Summing $\xi(i,j)$ over t, $\sum_{t=1}^{T-1} \xi_t(i,j)$ is the expected number of transitions from state S_i to state S_j. Let $\gamma_t(i) = P(q_t = S_i|O, \lambda)$ the probability of S_i at time t, given the observation sequence and the model. Summing $\gamma_t(i)$ over t, $\sum_{t=1}^{T-1} \gamma_t(i)$ is the expected number of transitions made from state S_i.

There are two main steps in the EM algorithm. The first step (often called the 'Expectation' step) of the algorithm is to estimate the expectations of quantities such as the expected number of times that state S_i is visited, $\sum_{t=1}^{T-1} \gamma_t(i)$ and the expected number of transitions from state S_i to state S_j, $\sum_{t=1}^{T-1} \xi_t(i,j)$. By counting these event occurrences, we can reestimate the parameters A, B, π of a HMM:

$$A = a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$B = b_j(k) = \frac{\sum_{t=1}^{T} \delta(O_t = k)\, \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

$$\pi = \gamma_1(i)$$

where A is the ratio of the expected number of transitions from state S_i to state S_j over the expected number of transitions from state S_i, B is the ratio of the expected number of times an observation k is made in state S_j over the expected number of times in state S_j, and π is the expected number of times in state S_i at time $t = 1$.

Given the current model $\lambda = (A, B, \pi)$ and let $\tilde{\lambda} = (\tilde{A}, \tilde{B}, \tilde{\pi})$ be the new estimated model, the second step (also known as the 'Maximisation' step) is to compute the maximum likelihood estimates of the parameters of these models. If $\tilde{\lambda}$ is more likely than λ, i.e., $P(O|\tilde{\lambda}) > P(O|\lambda)$ then a new model $\tilde{\lambda}$ is produced. The EM algorithm then iterates using the new model $\tilde{\lambda}$ and repeats the reestimation process to improve the probability of the observation sequence O, given the new model $\tilde{\lambda}$ until the likelihood function converges. Further details of this algorithm can be found in [60, 113, 143].

Wu, Dong and Xiao [146] use HMMs to recognise inhabitant's physical activities (e.g., walking, sitting, ascending stairs, etc.) in a home. They trained a bank of HMMs, each representing one physical activity. Behaviour is classified based on the HMM that achieves the highest probabilities. There is also work that used HMMs to identify and distinguish the behaviours between the different inhabitants in a home. This was considered in the work of Crandall and Cook [23]. Al-ani, Le Ba and Monacelli [1] use HMM to detect fall and physical activities of the inhabitant from accelerometers.

Although HMMs can be used to recognise behaviours from the observed sequences, it becomes difficult to model when the number of states grow. To deal with this problem, the Hierarchical Hidden Markov Model (HHMM) was introduced, which

is a generalisation of the HMM [32]. In this model, a top-level representation of behaviours is built up from a set of recognised events that arise from the individual sensor values [83, 103, 151]. This means that each state has its own HMM. An example of a hierarchical HMM for preparing meal behaviour is shown in Figure 3.5. When an end state is reached, it means that the child state returns the control to its parent state.

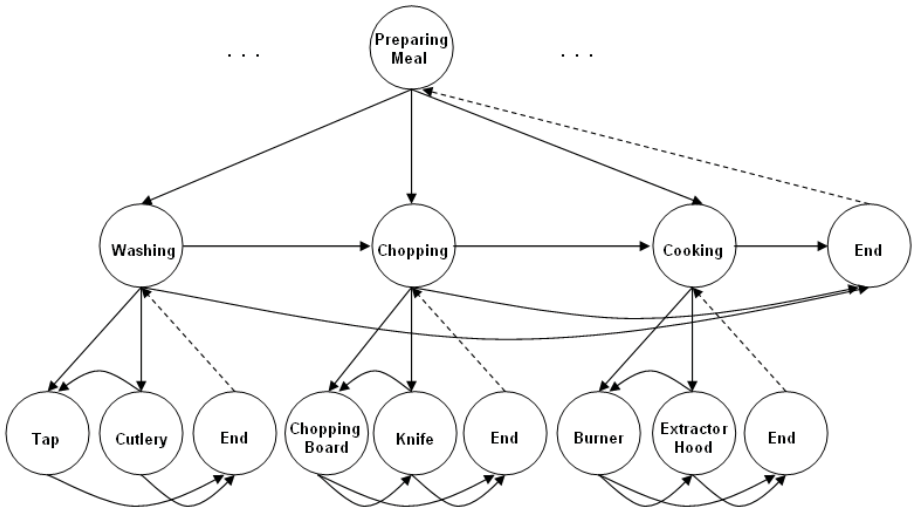


Figure 3.5: The hierarchical HMM. An example of preparing meal behaviour, which can be broken down to represent the events of preparing meal such as washing, chopping and cooking. Each individual event (e.g., cooking) can be broken down to represent the movement of the use of kitchen objects such as the burner and the extractor hood. The 'end' means that a final state is reached where the child state returns the control to its parent state (shown in dashed arrow). For details on this algorithm, see [32].

However, one of the problems with the hierarchical HMM is the high computational cost. The inference algorithm takes approximately $O(T^3N^L)$ time, where T is the length of the sequence, N is the total number of states in each level and L is the number of levels in the hierarchy [32, 99]. Thus, when the model has a large number of layers and/or states, this may become intractable for learning.

Within smart home research, it is also common to explicitly model the duration spent in one activity before the transition to the next activity. A variant to this is called the Hidden Semi-Markov Model (HSMM), where the transition to the next

behaviour is determined by the amount of time spent in each behaviour. In a HMM, the duration is implicitly modelled by the transition between the states, which strictly follows the Markov property where the probability of the transition to the next state S_j at time $t+1$ depends only on the state S_i at time t, i.e., $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$.

However, a state can remain unchanged for some random duration of time before transitioning to a new state. This self-transition, which is a transition to the same state $a_{ii} = P(q_{t+1} = S_i | q_t = S_i)$, occurs when the same sensor observation is observed. The HMM, which does not consider the amount of time spent in a particular state, makes it difficult to detect abnormal behaviours (e.g., taking a shower longer than the usual duration).

The HSMM alleviates this problem by explicitly modelling the duration, where it replaces the self-transition probability a_{ii} with a duration probability density function d_τ, which is the probability of transiting to the next state depending on the length of τ duration in the current state. In the works of Duong et al. [29], Marhasev, Hadad and Kaminka [86], and Natarajan and Nevatis [100], they use HSMM for behaviour recognition and abnormality detection.

The HMMs and variants of the model, which model the joint likelihood of the behaviours and the observations assume conditional independence between sensor observations. They may not be suitable for modelling complex behaviours such as interleaving behaviours. The conditional random field (CRF) [71], on the other hand, is an undirected graphical model that relaxes on this assumption by modelling the conditional likelihood of the behaviours given the sensor observations. There are works that use CRF for behaviour recognition. In the work of Hu and Yang [49], they use the CRF to recognise interleaving behaviours. Liao, Fox and Kautz [78] attempt to recognise activities and the locations of the activities by training a CRF from location data obtained from global positioning system (GPS). However, CRF in general is computationally intractable and often rely on approximation techniques

such as Monte-Carlo Markov Chain (MCMC) for inference.

3.2.4 Methods for activity segmentation

In previous sections, we reviewed those supervised learning methods that have been used to classify behaviour of the inhabitant from the sequence of tokens. Here, we review the methods used for segmenting the sensor stream. Activity segmentation is the process of breaking the sensor sequence into appropriate pieces that represent individual behaviours. Addressing segmentation in behaviour recognition is important so that classification can be performed on the sensor sequence.

Much of the work reported in the literature either assumes that the activities have been segmented [146, 153] or addresses the segmentation problem by using a window that slides across the sensor sequence [36, 133]. Tapia, Intille and Larson [133] used a set of feature windows, each representing one activity, and the length of each feature window was determined by the average duration that the inhabitant took to carry out that activity. The naïve Bayes classifier was used to calculate the probability of the current activity by shifting the feature windows over the sensor sequences, where the class with the highest likelihood was selected as the label for the activity.

The work of Govindaraju and Veloso [36] recognised activities using a set of HMMs, but maintained a single fixed window size, which was determined by averaging the length of the training segments used. However, determining the window this way may result in inaccurate segmentation especially when two similar activities occur close to one another in time. Gu et al. [37] introduced a heuristic algorithm to improve the segmentation boundary identification. This algorithm calculated the relative weight of each instance by taking the difference between the weight of the instances in each of these activities. The boundary between these two similar activities was determined by computing the maximum relative weights, which is the difference between pairs of relative weights.

There are many other places where time series of behaviours are recognised, such as recognising behaviours from posture information from video [61] and motion patterns [63, 104]. Kellokumpu, Pietikäinen and Heikkilä [61] use a set of HMMs, one for each activity, and apply the forward algorithm to monitor likelihood values. However, they do not use a sliding time window, preferring multiple window sizes and thresholding in order to separate out the activities. Niu and Abdel-Mottaleb [104] merge the outputs of the different HMMs using majority voting. A vote is assigned to each window and activity is classified based on the most common classification from the set of HMMs. A similar method is applied in the work of Stikic, Huỳnh, Van Laerhoven and Schiele [130] for the identification of housekeeping activities based on RFID data.

Kim, Song and Kim [63] turn the problem around and perform segmentation before classification, in this case for gesture recognition. The starting point of gestures is detected, and then a window is slid across the observation sequence until an end point is reached. The extracted gestures are then fed to HMMs for gesture recognition, with the final gesture type being determined by majority vote.

3.3 Description of Our Method

We use the Hidden Markov Model (HMM) as the basic representation of a behaviour. Many methods reported in Section 3.2.3 used complicated variants of the HMM such as the hierarchical HMM and the hidden semi-Markov model to recognise behaviours. However, these models have high computational cost in order to recognise a complete model of behaviours, which often becomes intractable for learning.

We approach the behaviour recognition problem by using a set of hidden Markov models that each recognise a behaviour (e.g., we have one HMM to represent the 'showering' behaviour, another HMM to represent 'doing laundry' behaviour, etc.) and they compete to explain the current observations. Figure 3.6 shows an example

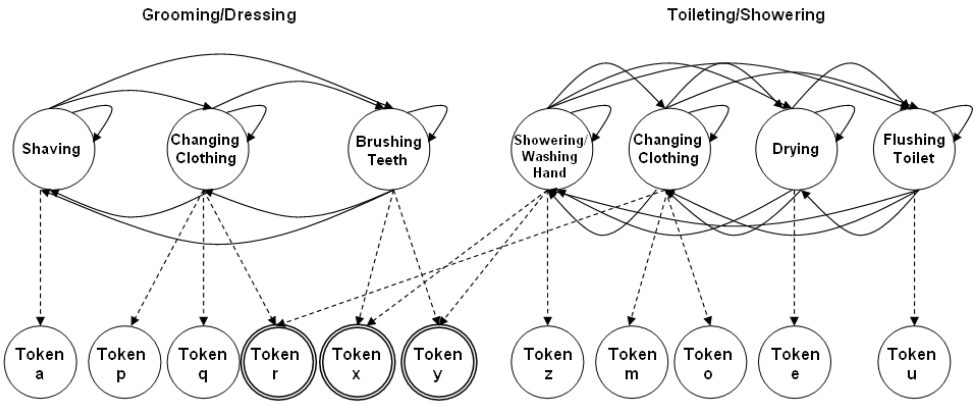


Figure 3.6: An example of two HMMs, one represents the grooming/dressing behaviour and the other HMM represents the toileting/showering behaviour. The nodes with double lines refers to tokens that are shared between two behaviours.

of two HMMs, one HMM represents 'grooming/dressing' behaviour and another represents 'toileting/showering' behaviour. The double lines shown in the figure refer to the sensors that are shared between these two behaviours.

In our work, the observations are the tokens from the sensors and the hidden states are the events that arise from the observations (in contrast to the majority of the works in the literature, the hidden states are the behaviours of the inhabitant). The HMMs in our work are ergodic, which means that in each HMM, every state is reachable from every other state, in a finite number of steps. The HMMs were each trained on the relevant training data using the standard Expectation-Maximisation (EM) algorithm, which is described in Section 3.2.3.

Once we have trained a set of of HMMs $\lambda_1, \lambda_2, \ldots, \lambda_m$, our next task is to recognise the behaviours from the sensor stream. The data that is presented to the HMMs is chosen from the sensor stream using a variable window that moves over the sequence. A winning HMM λ_{winner} is chosen based on the HMM that maximises the likelihood of the sensor observations $O_1, O_2, \ldots, O_T$, i.e., $\lambda_{winner} = \arg\max_j P(O_1, O_2, \ldots, O_T|\lambda_j)$.

Many methods reported in Section 3.2.4 used a fixed window length to partition the sensor stream. However, the choice of the size of this window is important, because it is unlikely that all of the activities in the sequence belong to one behaviour, and

so the HMM chosen to represent it will, at best, represent only some of the activities in the sequence. To see the importance of the problem, consider the three different cases shown in Figure 3.7. In each, a behaviour w takes up much of the window and is the winning behaviour. However, the location of it in the window differs, and we want to ensure that other behaviours in the window are also recognised.

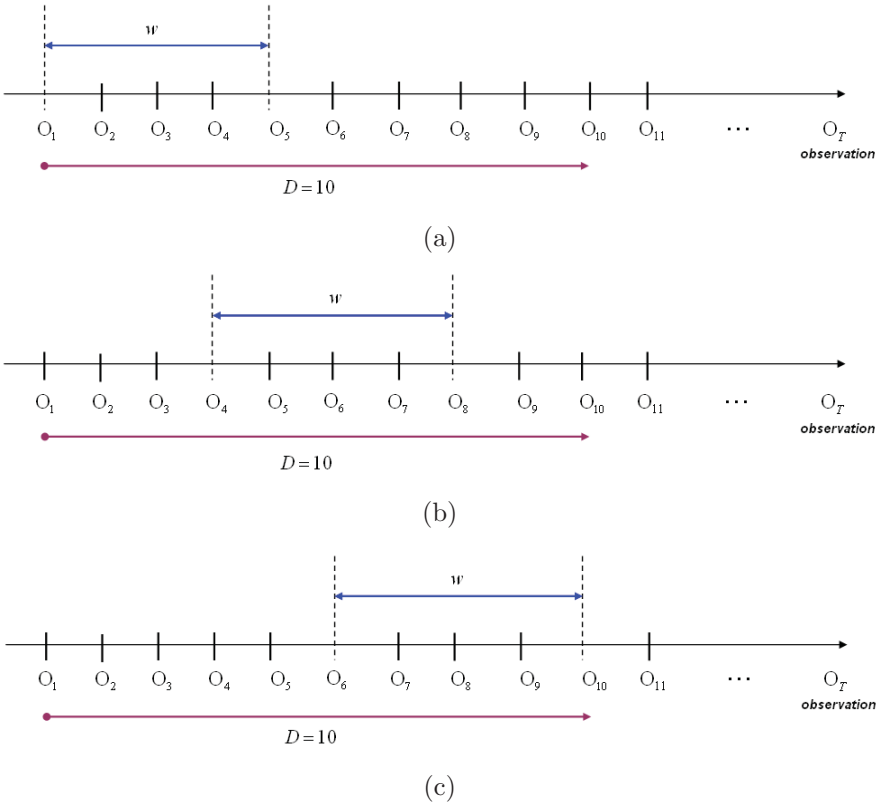


Figure 3.7: An activity w does not need to take up the entire window. Even assuming that the actions in a behaviour are contiguous, it could be (a) at the start of the window, (b) in the middle, or (c) at the end. If the entire window is classified as one behaviour, then a potentially large number of behaviours are missed. D is the window size and $O_1, O_2, \ldots, O_T$ is the observation sequence.

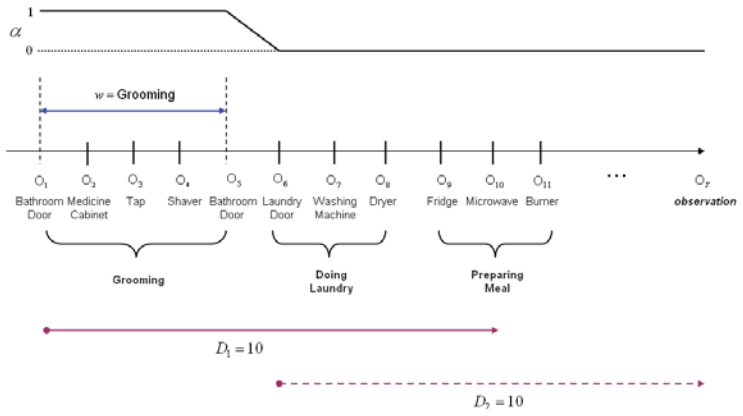
We present an alternative solution to this problem. We use a variable window length that moves over the sequence of observations, where it has the ability to automatically configure the window size based on the sensor observations. We first slide a window of length 10 across the sensor stream, presenting the 10 observations in the window to the sets of trained HMMs for competition (this window size of 10 is

determined experimentally, as described further in Section 3.5.3). A winning HMM is chosen based on the HMM that maximises the likelihood of the 10 observations in the window. Since we want to ensure that the majority of behaviours in the window are recognised, we perform a re-segmentation using the forward algorithm (described in Section 3.2.3). This is achieved by calculating the likelihood of each sensor observation in the window according to the model of the winning HMM, following Equations 3.2–3.4.

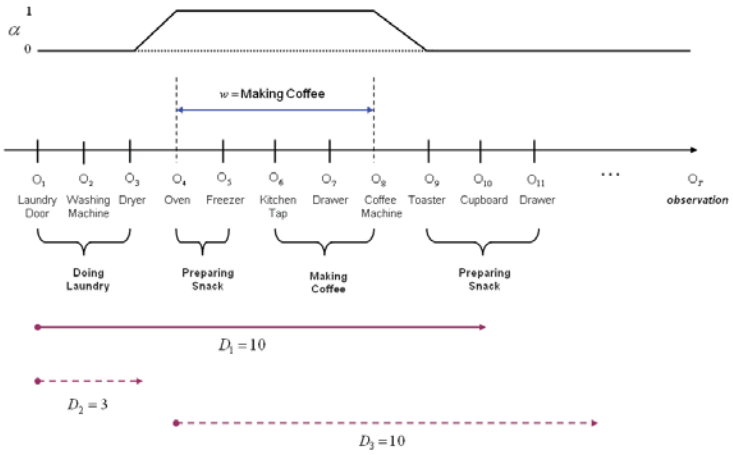
By monitoring the forward variable (α) for each sensor observation, we can determine how well the 'winning' HMM matches a given observation sequence, i.e., choosing an optimal state sequence that explains the sensor observation. When $\alpha > 0$, it means the model of the 'winning' HMM recognises the sensor observations, while $\alpha = 0$ means that the winning HMM does not 'recognise' the sensor observation, i.e., none of the states in the HMM recognises the sensor observation. The changes in α value signify a 'change' of activity from the observation stream. Figure 3.8 shows an interpretation of Figure 3.7 in terms of the α values computed by the forward algorithm applied to one particular HMM, the one selected as the 'winner' for this window. To simplify the illustration, the α value in the figure is quantised into the set $\{0, 1\}$. When $\alpha = 1$, it means that the winning HMM recognises the sensor observations and 0 otherwise.

If the $\alpha > 0$ at the beginning of the observation sequence then it is likely that the case in Figure 3.8(a) is occurring. Following Figure 3.8(a) we see that there is a drop in α value between observations O_5 and O_6, which suggests that the behaviour has changed. We can therefore classify $O_1, O_2, \ldots, O_5$ as belonging to the winning behaviour w (i.e., grooming), and then initialise a new window of default size (D) at O_6. When D is initialised, all the observations within D will then be fed to HMMs for competition and the process iterates.

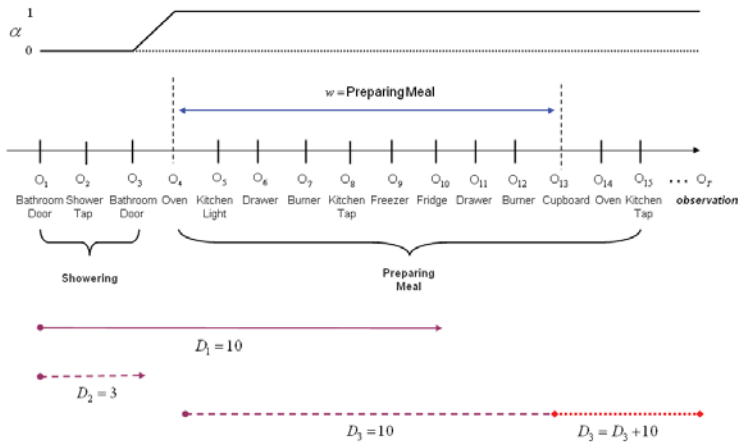
The second case occurs when the winning behaviour best describes observations



(a)



(b)



(c)

Figure 3.8: The solid line above the observation sequence shows the possible representations of a winning sequence using the α values. For simplicity, the α value is quantised into the set $\{0, 1\}$. D is the default window size and $O_1, O_2, \ldots, O_T$ is the observation sequence. The original observation sequence is shown as the down curly brace. In (c), the window is extended (shown by dotted arrow) when the behaviour does not stop during the window. For details, see the text.

that fall in the middle of the window, e.g., $O_4, O_5, \ldots, O_8$ in Figure 3.8(b). Since the winning behaviour (w) does not describe observations O_1, O_2 and O_3, the probability for these three observations is low (i.e, $\alpha = 0$) and we observe a jump in the α value at O_4. When this is observed, a new window (D_2) is initialised that contains only the three observations (i.e., O_1, O_2, O_3) that are not explained by behaviour w. The HMM competition is then rerun on this window, where the winning behaviour for this window is 'doing laundry'.

With regard to the remaining sequence (O_4 and onwards) it would be possible to use HMM w (i.e., making coffee) and continue to monitor the α values. However, it was observed that sometimes there may be an overlap in individual sensor activations (e.g., O_4 and O_5) between 'preparing snack' and 'making coffee' behaviours. If we continue to use the winning HMM and monitor the α values, an additional step would be required by the algorithm to calculate the α value on O_4 and O_5, which is not even described by the winning HMM. For this reason, a new window of default size is started at O_4 and the HMM competition is rerun on this sequence.

Since the two cases have been considered when the winning behaviour is at the beginning and middle of the window, the only possibilities remaining are that the behaviour is at the end and either stops during the window (Figure 3.8(a)) or does not (Figure 3.8(c)). The first case is already dealt with, and in the second case, we could simply classify the activities in the window as w and start a new one at the end of the current window. If we do so, then we will have two examples of the same behaviour abutting one another. Instead we extend the size of the window (shown as a dashed arrow in Figure 3.8(c)) and continue to calculate the α value for each observation until α drops.

3.3.1 The algorithm

Since each HMM represents one behaviour, we trained each HMM separately on the training data using the standard Expectation-Maximisation algorithm, which is described in Section 3.2.3. Once the HMMs were trained, we use the following algorithm for behaviour recognition and segmentation, which is based on competition between HMMs and a variable window length:

(1) *Initialisation*

- a set of trained HMMs, $\lambda_1, \lambda_2, \ldots, \lambda_m$, where m = total number of trained HMMs
- sensor stream $O_1, O_2, \ldots, O_T$, where T = length of sensor stream
- D = window size, s = 1 and e = D

(2) *Competition among HMMs*

- calcLikelihood = 0, firstObservedAlpha = 0
- slide a window across sensor stream so each HMM gets inputs $(O_s, O_{s+1}, \ldots, O_e)$
- calculate $P(O_s, O_{s+1}, \ldots, O_e|\lambda_j)$ according to the model of each HMMs, $\lambda_1, \lambda_2, \ldots, \lambda_m$
- compute winning HMM such that

$$\lambda_{\text{winner}} = \arg\max_j P(O_s, O_{s+1}, \ldots, O_e|\lambda_j)$$

(3) *Calculate likelihood for each sensor observation on the window*

- For each sensor i in $O_s, O_{s+1}, \ldots, O_e$, calculate the likelihood α_i (using Eq. 3.2 and 3.3) according to the winning HMM, λ_{winner}
- if $\alpha_s == 0$
  - firstObservedAlpha = 1

- end if
- if $\alpha_i == 0$ and calcLikelihood $= 0$
  - calcLikelihood $= 1$
  - repeat step (3)

  elseif $\alpha_i > 0$ and calcLikelihood $= 1$
  - reinitialise window, $e = i - 1$
  - rerun HMM competition on this new sequence, $O_s, O_{s+1}, \ldots, O_e$ by returning to step (2)

  elseif $\alpha_i > 0$ and current sensor observation is at the end of the window (i.e., $i = e$)
  - extend window, $e = i + D$
  - repeat step (3)

  elseif $\alpha_i > 0$ and current sensor observation is not at the end of the window (i.e., $i < e$)
  - firstObservedAlpha $= 1$
  - repeat step (3)

  elseif $\alpha_i == 0$ and firstObservedAlpha $== 1$
  - classify $O_s, O_{s+1}, \ldots, O_{i-1}$ as λ_{winner}
  - update $s = i$
  - reinitialise window, $e = s + D$
  - firstObservedAlpha $= 0$
  - rerun HMM competition on this new sequence, $O_s, O_{s+1}, \ldots, O_e$ by returning to step (2)
- end if

3.4 Evaluation Method

To demonstrate our system, we used a real smart home dataset from the MIT PlaceLab, which is described in Section 1.5. We used a leave-two-out cross validation method for each evaluation in order to calculate the confusion matrix and measure the recognition accuracy. From the total of 16 days, we used 14 days for training and the remaining two days for testing (see Figure 3.9). Since we want to ensure that every behaviour (particularly doing laundry and washing dishes behaviours, which do not occur daily) is seen in the test set, we take pairs of days for testing.

The main metric that we are interested in is *recognition accuracy*, which is the ratio of the total number of behaviours correctly identified by the algorithm over the total number of activities used for testing. We repeated the process 8 times, with the final recognition accuracy being calculated by averaging the accuracies in each run.

Table 3.1 shows the different training-testing splits along with the number of activity examples and tokens that we used for training and testing. We also tried other numbers of days in order to investigate the amount of training data required to train the HMMs. The HMMs were each trained on the relevant labelled data in the training set using the standard Expectation-Maximization (EM) algorithm (described in Section 3.2.3).

Training-test Sets	**No. of Activity Examples**		**No. of Sensor Tokens**	
	Training	**Testing**	**Training**	**Testing**
1st Set	279	31	1672	133
2nd Set	256	54	1456	349
3rd Set	290	20	1688	117
4th Set	277	33	1561	244
5th Set	261	49	1563	242
6th Set	276	34	1592	213
7th Set	273	37	1625	180
8th Set	258	52	1478	327

Table 3.1: The different training-testing splits that we tried, along with the total number of activity examples and tokens used in each training and testing set.

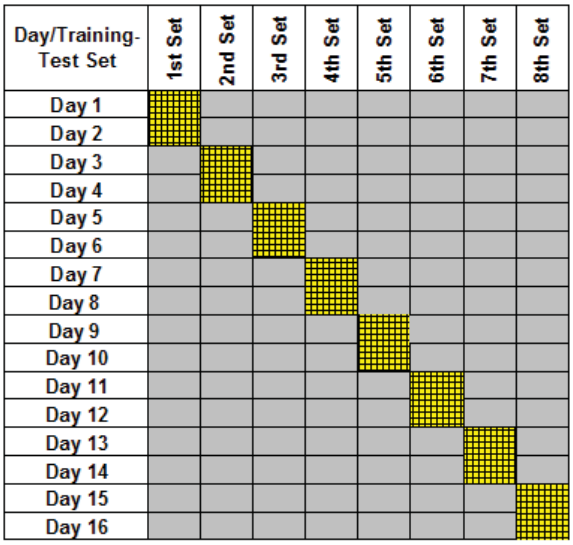


Figure 3.9: Leave-two-out cross validation method. We used 14 days for training (shaded grey) and the remaining two days (crosshatch) for testing.

3.5 Experimental Results

To test our system, we conducted four separate experiments. In the first, we looked at the accuracy of the algorithm to recognise behaviour based on the HMMs competition and variable window length, while in the second we compared the algorithm with fixed window length. In the third experiment, we looked at the effects of window size on the efficiency and accuracy of the algorithm. In the fourth experiment, we looked at how much training data was required for accurate results.

3.5.1 Experiment 1: Competition among HMMs and variable window length

The aim of this experiment is to test the efficacy of our method based on competition among HMMs and a variable window length. In this experiment, we used a default window size of 10, and ran the entire algorithm over the sensor observations on different test sets. The results of sliding this window over the data are shown in Figure 3.10, which displays the outputs of the algorithm, with the winning behaviour at each time being clearly visible.

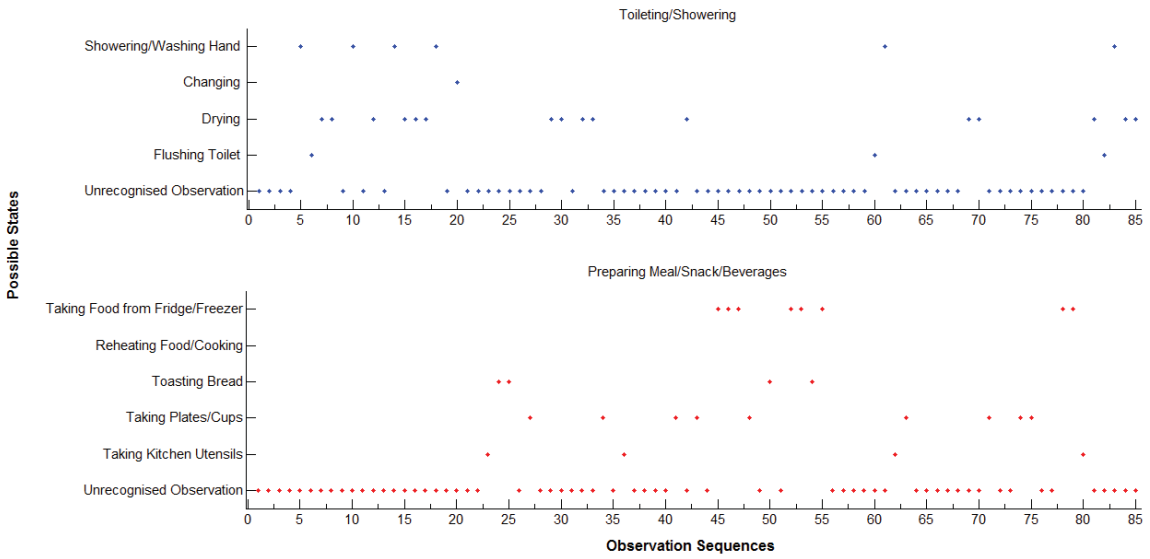


Figure 3.10: Illustration of competition between HMMs based on two different behaviours: toileting/showering and preparing meal/snack/beverages. The example is based on the first 85 observations of the 5th test set. There are other behaviours in this example but they are omitted for clarity. The y-axis shows the different states (events) that arise from the individual sensor values. The 'unrecognised observation' label means that none of the states in the particular HMM recognises the sensor observations.

As the figure shows, we can determine that the subject is toileting/showering between observations O_5 and O_{20}, followed by preparing meal/snack/beverages between observation O_{23} and O_{27}. The y-axis shows the different states (events) that arise from the individual sensor values when performing an activity. The 'unrecognised observation' label shown in the figure means that none of the states in the particular HMM recognises the sensor observations. This is observed when $\alpha = 0$.

The recognition accuracy results across different test sets are shown in Table 3.2. Our method achieved an overall accuracy of 91%, which shows that our method based on competition between HMMs and variable window length can be used to recognise and segment the sensor stream into behaviours. A low recognition accuracy is observed on the 8th test set. This is mainly due to the number of cleaning/putting away groceries activities that are observed in that test set where our algorithm is not able to identify this activity since it involves the same sensors that other activities

Test Sets	No. of Activity Examples for Testing	No. of Activities Correctly Identified	Recognition Accuracy
1st Set	31	28	90%
2nd Set	54	49	91%
3rd Set	20	19	95%
4th Set	33	30	91%
5th Set	49	45	92%
6th Set	34	31	91%
7th Set	37	35	95%
8th Set	52	44	85%
Average			**91%**
Standard Deviation			**3.2**

Table 3.2: Results on supervised learning based on competition between HMMs and variable window length on different training/testing splits.

would use.

3.5.2 Experiment 2: Comparison between variable window length and fixed window length

This experiment is designed to compare the algorithm with the fixed window length. In this experiment, we used a fixed window length of sizes 5 and 10, and ran the algorithm over the sensory stream on different test sets for each evaluation. For the fixed window length, each window is shifted over the sensor stream with increments according to the window size used in the experiment. E.g., for fixed window length of size 5, the window is shifted over the sensor stream with increments of 5 sensor observations each time.

The results are shown in Table 3.3. We have conducted a significance test to compare the recognition accuracy of fixed window length and our variable window length method on window size = 10. An F-test was first carried out to determine the equivalence of the variances for these two methods. The test statistic is $F = (\frac{S_1^2}{S_2^2})(\frac{\sigma_2^2}{\sigma_1^2}) = \frac{3.2}{9.3} = 0.3441$ with p-value = 0.09. Thus, the null hypothesis of equal variances is accepted. A Student's t-test is conducted to test the alternative hypothesis that the average recognition accuracy of the variable window length method is significantly higher than the average recognition accuracy of the fixed window length.

Test Sets	Recognition Accuracy			
	Window Size = 5		Window Size = 10	
	Variable Window Length	Fixed Window Length	Variable Window Length	Fixed Window Length
1st Set	90%	55%	90%	48%
2nd Set	91%	69%	91%	54%
3rd Set	95%	75%	95%	75%
4th Set	91%	76%	91%	58%
5th Set	92%	49%	92%	51%
6th Set	91%	68%	91%	62%
7th Set	95%	70%	95%	59%
8th Set	85%	60%	85%	46%
Average	**91%**	**65%**	**91%**	**57%**
Std. Deviation	**3.2**	**9.6**	**3.2**	**9.3**

Table 3.3: Comparison results between the variable window length and fixed window length based on window size = 5 and window size = 10.

The test statistic is $T = \frac{(\bar{x_1}-\bar{x_2})-(\mu_1-\mu_2)}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} = \frac{(91-57)-(0)}{(\sqrt{48.37})(\sqrt{\frac{1}{8}+\frac{1}{8}})} = \frac{34}{3.48} = 9.8$ with p-value $= 5 \times 10^{-8}$, where S_p is the pooled variance. Thus, the null hypothesis is rejected and we can conclude that the average accuracy of our variable window length method is significantly higher than the average accuracy of the fixed window length and that our approach of re-segmentation improves the recognition results, with an average accuracy of 91%.

The problem with fixed window length is that it can only identify one winning behaviour and resulting in other behaviours in the window not being identified. Another problem with fixed window length is to determine the correct window size, since it affects the recognition performance. The size of the fixed window length is often determined empirically. As can be seen in Table 3.3, when the window size = 5 is used, it achieves an accuracy of 65% and drops to 57% when the window size = 10 is used. The accuracy of our proposed method is consistent even when different window sizes are used. We still achieve 91% recognition accuracy on both window sizes = 5 and 10.

3.5.3 Experiment 3: Size of default window

Although our proposed method uses a variable window length that has the ability to automatically configure its window based on the sensor observations, we want to know if the initial window size of the variable window length has any effect on

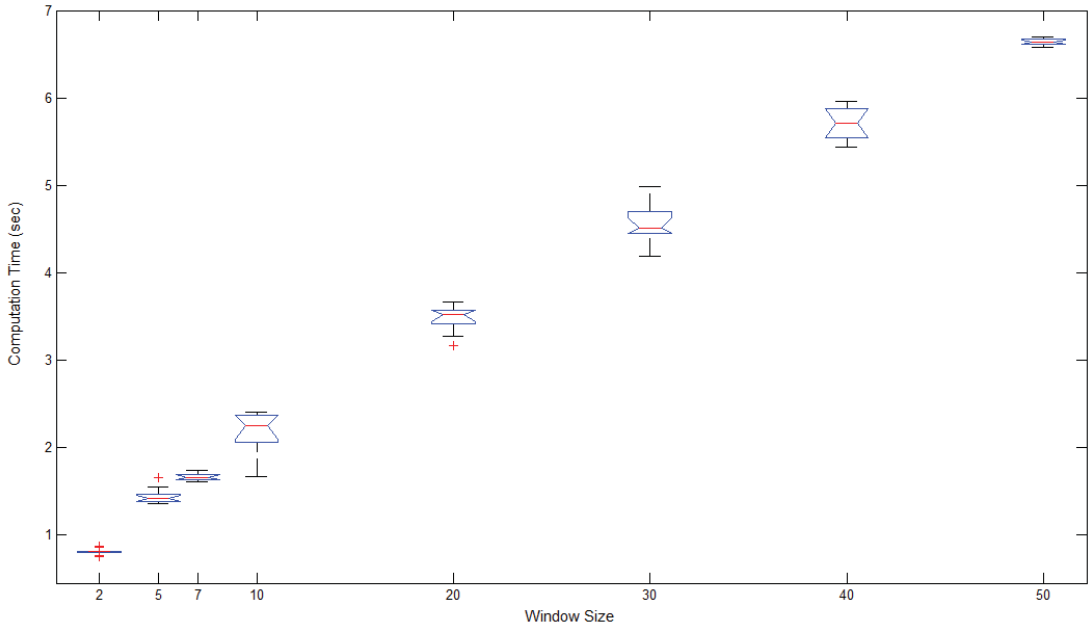


Figure 3.11: Boxplot for computational time (in sec) across different window sizes based on 10 runs each on 2nd test set. For each box, the central line is the median, and the edges of the box are the 25^{th} and 75^{th} percentiles.

the computational and recognition performances. For this reason, this experiment examines if different sizes of window have any effect on computational performance and recognition accuracy. A variety of window sizes ranging from $2, 5, 7, 10, 20, 30, 40$ and 50 was conducted, each with 10 runs on all test datasets.

The results exhibited in Figure 3.11 clearly show that computation time grows approximately linearly with the size of window, and therefore a shorter window length is preferred in order to keep the computational costs low. Although the results presented in the figure are based on the 2nd test set, a similar trend was observed in all other test sets.

We also evaluated these different window sizes for recognition accuracy, i.e., the ratio of the total number of behaviours correctly identified by the algorithm over the total number of activities used for testing. Table 3.4 shows the results of using different window sizes. As the table shows, the results are not significantly different across the different window sizes.

Test Sets	Recognition Accuracy on Different Window Sizes							
	2	**5**	**7**	**10**	**20**	**30**	**40**	**50**
1st Set	90%	90%	90%	90%	90%	90%	90%	90%
2nd Set	91%	91%	91%	91%	91%	89%	89%	89%
3rd Set	95%	95%	95%	95%	90%	90%	90%	90%
4th Set	91%	91%	91%	91%	91%	91%	91%	91%
5th Set	92%	92%	92%	92%	92%	92%	92%	92%
6th Set	91%	91%	91%	91%	91%	91%	91%	91%
7th Set	95%	95%	95%	95%	95%	97%	97%	97%
8th Set	85%	85%	85%	85%	85%	85%	83%	83%
Average	**91%**	**91%**	**91%**	**91%**	**91%**	**91%**	**90%**	**90%**
Standard Deviation	**3.2**	**3.2**	**3.2**	**3.2**	**2.8**	**3.3**	**3.9**	**3.9**

Table 3.4: The effect of using different window sizes on behaviour-level recognition accuracy.

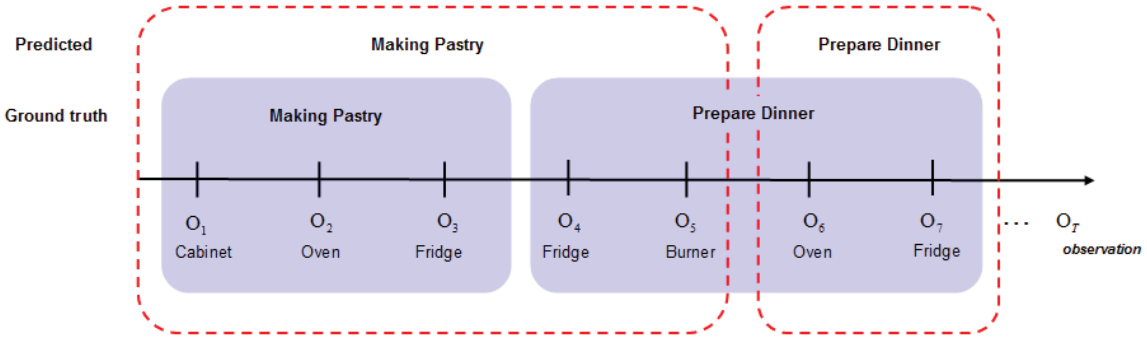


Figure 3.12: Illustration of token-level recognition accuracy, which compares the predicted behaviour (shown in dashed box) with the ground truth label (shown in shaded box) for every token observation. E.g., token observations O_4 and O_5 are misclassified by the algorithm as 'making pastry' rather than 'prepare dinner'.

In view of this, we define another separate measurement of accuracy of our results, which compares the behaviour output by the algorithm with that of the label for every token observation. This is particularly sensitive to times when two behaviours that occur one after the other share the same observations. Using the example shown in Figure 3.12, the token observations O_4 and O_5 are misclassified as 'making pastry' rather than 'prepare dinner'. This means that the token-level accuracy for 'prepare dinner' is 50% (i.e., 2/4). Note that the behaviour-level accuracy in this example is 100% since the algorithm is able to identify the sequence of activities i.e., 'making pastry' and 'prepare dinner.'

The recognition results shown in Table 3.5 drop when the window size is below 5 and above 40. This can be explained by the fact that if the window size is too small

Test Sets	Recognition Accuracy on Different Window Sizes							
	2	**5**	**7**	**10**	**20**	**30**	**40**	**50**
1st Set	73%	76%	76%	76%	74%	74%	74%	74%
2nd Set	84%	85%	85%	85%	85%	84%	84%	84%
3rd Set	79%	85%	85%	85%	84%	85%	85%	81%
4th Set	73%	76%	75%	75%	75%	75%	75%	74%
5th Set	73%	74%	73%	73%	72%	71%	71%	71%
6th Set	83%	84%	84%	84%	84%	83%	83%	84%
7th Set	84%	85%	85%	85%	85%	87%	87%	87%
8th Set	54%	54%	54%	54%	54%	54%	53%	54%
Average	**75%**	**77%**	**77%**	**77%**	**77%**	**77%**	**77%**	**76%**
Standard Deviation	**10**	**10.6**	**10.6**	**10.6**	**10.6**	**10.9**	**11.2**	**10.6**

Table 3.5: The effect of using different window sizes on token-level recognition accuracy.

(e.g., window size = 2), there will be insufficient information to describe the current behaviour and the classification is biased towards the behaviour that has fewer token observations, and if the window size is too large (e.g., more than 30), the chances of misclassification are higher since classification is biased towards the majority of the behaviours that comprise that window. Although the accuracy is the same for window sizes $5, 7, 10, 20$ and 30 on this dataset, we have chosen to use window size 10 for all the experiments conducted in this thesis so that we can examine all the representations of a winning sequence, which could be at the start, middle or at the end, within the window (see Figure 3.7).

3.5.4 Experiment 4: Size of training data

The objective of this experiment is to analyse the amount of training data needed to train the HMMs. The most important thing is that every behaviour is seen several times in the training set to ensure that the HMM acquires a good representation of that behaviour. From the total of 16 days of data, we tried different splits of the data, from 5 days for training (and 11 days for testing) through 8 days, and 11 days for training.

The results on recognition accuracy are presented in Table 3.6. As the table shows, the size of training data does not have an impact on recognition accuracy. Even when trained on 5 days, with a small number of activity examples per behaviour shown in

Training Set		Test Set		Recognition
No. of Days	**No. of Activity Examples**	**No. of Days**	**No. of Activity Examples**	**Accuracy**
5 Days	95	11 Days	215	94%
8 Days	138	8 Days	172	94%
11 Days	203	5 Days	107	93%
Average				**94%**
Standard Deviation				**0.6**

Table 3.6: Recognition accuracy on different size of training data on different training-test sets.

Table 3.7, we are still able to achieve an accuracy of 94%. It seems that the proposed method does not need that large a set of training data, although this may not be true for more complicated behaviours.

Acitivies	**No. of Activity Examples**
Doing/putting away laundry	11
Grooming/dressing	18
Washing/putting away dishes	15
Toileting/showering	26
Preparing meal/snack/beverages	22
Cleaning/putting away groceries	3
Total activity examples	95

Table 3.7: The number of activity examples per behaviour on 5 days training dataset

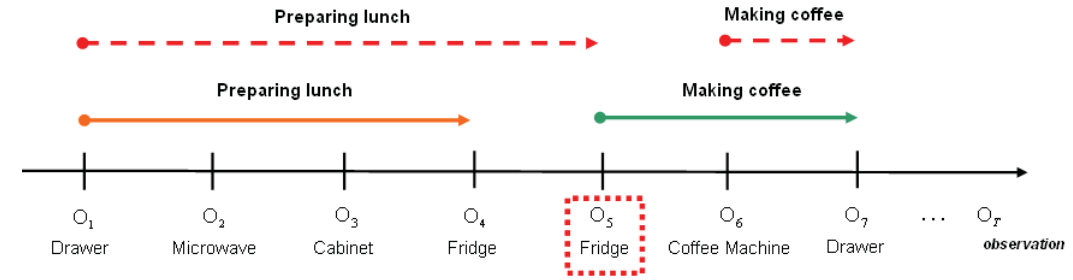
Note that the accuracies presented in Table 3.2 are based on leave-two-out cross validation method, while the accuracies presented in Table 3.6 is trained on 5 days, through 8 days, and 11 days. In leave-two-out cross validation method (Table 3.2), the number of activities in each test set is lower since it only consist of 2 days data and thus the weightage of 1 misclassification is much higher. This explains why the accuracy in Table 3.2 is slightly lower.

3.5.5 Discussion

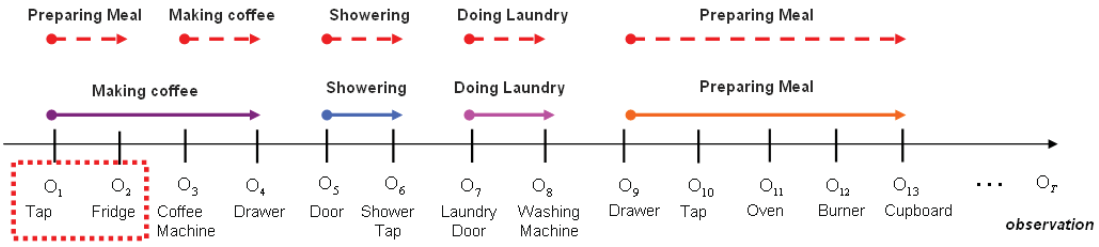
Our algorithms worked very well, producing over 90% recognition. However, it is still instructive to see if there are consistent reasons for the misclassifications that did occur.

We identified one main reason for misclassification, which is that individual sensor

observations can be in several behaviours. There are two places where this can be a problem. The first is when the end of one behaviour contains observations that could be in the start of the next. For example, the last activity for preparing lunch could be to put the leftover food in the fridge. After preparing lunch, the inhabitant proceeds to make a cup of coffee, and the first event to make a cup of coffee is to take the milk from the fridge (see observation O_5 in Figure 3.13(a)). This will not pose a problem if the second behaviour (i.e., making coffee) happens immediately after the first (i.e., preparing lunch). However, if the second behaviour happened two hours after the first, that would be a totally different unrelated behaviour.



(a)



(b)

Figure 3.13: (a) Misclassification (at O_5, shown in dashed arrow) occurs when the end of one behaviour contains the observations that could be in the start of the next behaviour. (b) Observations O_1 and O_2 (shown in dashed box) are misclassified as 'preparing meal'. This occurs when the winning behaviour (i.e., 'preparing meal') is not at the start of the window, but activities at the start could be interpreted as being part of that behaviour. Solid arrows represent ground truth and dashed arrows represent the predicted output.

The second place that this can be seen is where the winning behaviour is not at the start of the window, but those activities at the start could be interpreted as being part of that behaviour. This can be seen in Figure 3.13(b) where observations O_1

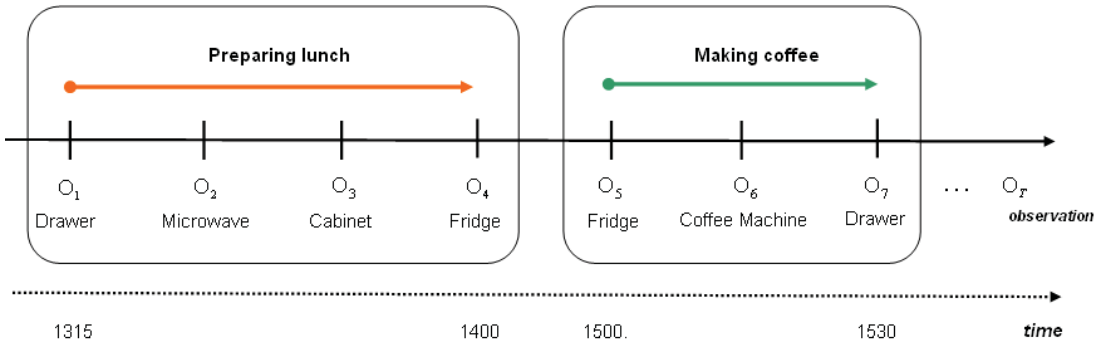


Figure 3.14: Illustration of how temporal information can be included to classify behaviours.

and O_2 are misclassified as 'preparing meal'. It was experimentally observed that this was more likely to happen where the size of the window was large, because more behaviours were observed. One way to reduce the misclassification is by adding extra information in order to improve the classification accuracy. This can be achieved by augmenting the current algorithm with temporal information. If temporal information is included, then places where two behaviours abut one another can be reduced (see Figure 3.14).

Although our method performs well to recognise the behaviours of the inhabitant, achieving an average accuracy of 91%, the algorithm may take a long time to run if there are large number of competiting HMMs. This can be addressed by incorporating spatial information. Since the locations of the sensors implicitly provide some spatial information of where the activity occurs, we can use this spatial information to reduce the number competing HMMs. To see how this could be possible, we use the example presented in Figure 3.15. If the location about where the activity occurs is known and that the activity occurs in the kitchen, then there are only 3 possible behaviours that are more likely to occur, which are either cooking, preparing a beverage or washing dishes. This reduces the number of HMMs that are allowed to compete, which is in this example, to 3.

The current study assumes that actions in a behaviour are contiguous, and that all

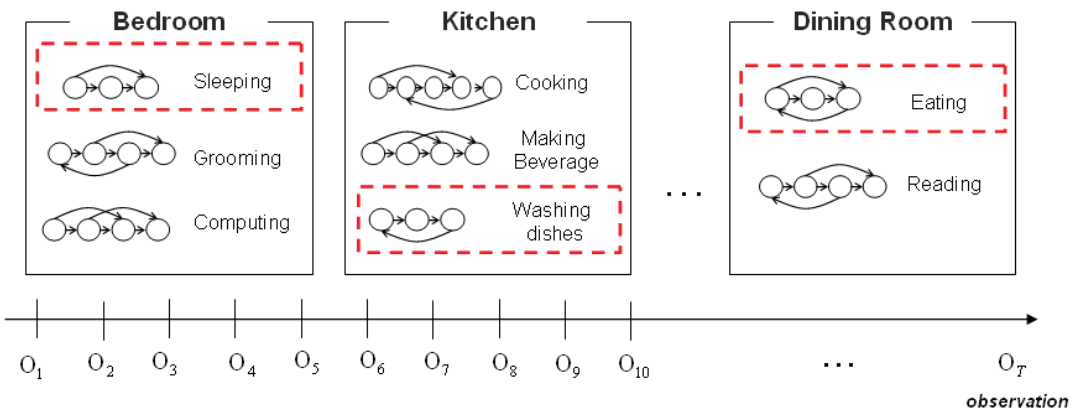


Figure 3.15: Illustration of how each behaviour is defined using separate HMMs in different locations in the home. For example, cooking, preparing a beverage and washing dishes are the representation of separate behaviours in the kitchen. If the location of where the activity occurs is known (e.g., kitchen), then this could reduce the number of competing HMMs to 3 from a total of 8 HMMs.

of the separate parts of the behaviour are different instances of that behaviour. This may not be the case in the real environment, as behaviours are normally interleaved: a person may well make a beverage at the same time as preparing lunch, which could be done while the laundry was running. This system will not deal with these behaviours in any sensible way, which is left for future work.

3.6 Problems with Supervised Learning

This chapter introduces a supervised learning method to learn the mapping between sensor outputs and behaviours. However, as supervised learning, it needs ground truth, which means that somebody has to go through several weeks to manually label their activities; either directly when observing the activities performed by the inhabitant or having the inhabitant keep a diary of what she does. Regardless of whether using direct or indirect observations, manual labelling is a time consuming process. This is rather impractical, especially when implementing a home for the elderly.

Considerations such as these have led us to the conclusion that unsupervised

learning may be a more sensible learning paradigm. In this approach, the algorithm is asked to identify structure within the inputs without any benefit of data labels. Thus, the algorithm tries to categorise the data into similar classes that exhibit some form of commonality. The challenge is to ensure that the algorithm identifies the structures that we want it to find.

Motivated by these thoughts, the next chapter provides a solution to the problem of automatically learning the mapping between the sensor information and the behaviours of the inhabitant, without any prior human labelling or knowledge about the sensor stream.

3.7 Summary

Algorithms for behaviour recognition generally fall into two categories:– those that are based on an explicit representation of behaviours together with the events that characterise them, and those that mine them from sensor streams. The second has the advantage that we don't need to know what events constitute a behaviour, and therefore it is the preferred approach by many researchers and also in the work described in this chapter.

This chapter has presented a system that performs behaviour recognition and segmentation of the sensor stream based on competition between a set of trained Hidden Markov Models and a variable window length. Our experimental results show that our method works effectively, achieving an average accuracy of 91%. We have also shown a comparison between variable window length and fixed window length, and that the variable window length works best. We have investigated different window sizes, and found that relatively short ones work best. The experimental results also shown that our method does not need a large amount of training data. With the above strengths, the proposed method is thus a promising approach for behaviour recognition and segmentation in a smart home.

Chapter 4

Unsupervised Learning of Activities

In this chapter, an unsupervised learning solution to the problems described at the end of the previous chapter is provided. An overview of unsupervised learning methods is described and a description of our approach is given. The efficacy of the method is demonstrated with experiments on two benchmark datasets and compared with other unsupervised learning algorithms.

4.1 The Problem

At first thought, it seems clear that behaviour recognition is a supervised learning problem. However, as supervised learning, it needs labelled training data. There are several problems with labelled data. To begin with, labelling the data is rather tedious, and prone to error. Even if the inhabitants are asked to keep a journal of what tasks they are engaged in at a very detailed level (and given facilities to help, such as a smart phone or personal organiser), actually doing it accurately is very hard, and irritating to perform. Certainly, it is not possible to imagine that anybody who wishes to live in a smart home would willingly go through several weeks of this before the home starts to work! Furthermore, labelling only works for the subject under study and may not generalise to other inhabitants in different home settings. This means that re-labelling is needed for different inhabitants, since people have

variations in their activities and the sensors installed in each home could be different.

Considerations such as these have led to the approach of training a recognition system from a limited number of labelled examples, aided by a larger set of unlabelled data in a semi-supervised approach to learning. There have been works that attempt to transfer learned knowledge to a new physical domain [114] or a new activity task [155], or use active learning to engage users to label classes that have the lowest confidence [131]; they do, however, rely on some partially labelled data.

An alternative approach is to automatically learn the mapping between sensor information and behaviours in an unsupervised manner. While there are many methods of performing unsupervised learning (e.g., clustering) described in the literature, very few of them can cope with data sequences of variable length. Essentially, most work by taking input vectors of known dimensionality and identifying prototypes or exemplars that best represent clusters within the data using a distance metric in the vector space of inputs; examples include the Self-Organising Map [75, 156] and hierarchical clustering [123]. While useful for a great many applications, these solutions may not be practical in applications where the inputs are not of fixed dimensionality, so that data sequences are variable in length, such as speech recognition, behaviour recognition and text classification. In a smart home, the number of sensors that define a behaviour is not fixed. They can vary:

(a) *between behaviours* (Referring to the example in Figure 4.1, the number of sensors that defined the 'preparing meal' is 4, while the 'grooming' behaviour is of length 2.)

(b) *within different instances of the same behaviour.* (For example, 'preparing meal' could involve sensors on the stove, tap and fridge in one day, and the microwave the next day when the person decides to reheat the leftover food from yesterday.)

Even where data are of fixed length, they may be embedded into a data stream,

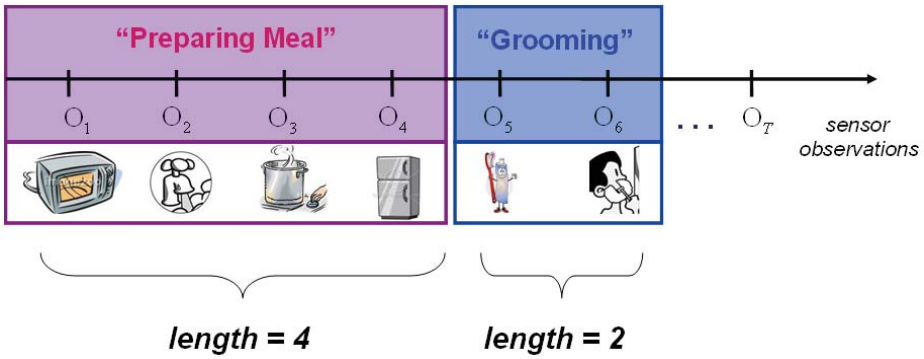


Figure 4.1: An example of variable length sequences. There are 4 sensor readings in 'preparing meal' behaviour and 2 sensor readings in 'grooming' behaviour.

which requires segmentation before the individual input vectors can be assembled. This chapter addresses the problem of identifying and then learning patterns of variable length in a data stream without any prior knowledge about them. This includes both the segmentation of the data stream into suitable patterns and the identification of patterns when they are seen during testing.

We first provide an overview of relevant unsupervised learning algorithms and review the methods that are used to recognise human behaviours. We then introduce our approach, which can reliably cluster the unlabelled data stream and segment the data stream into behaviours that we have identified. We evaluate the effectiveness of our proposed method using both fixed-length and variable-length benchmark datasets, and compare the labels produced by our approach with the labels assigned by a human.

4.2 Relevant Literature

The literature review in this section focuses on the unsupervised learning methods that automatically find the mapping between the sensor information and the behaviours, without any prior human labelling. The approaches that we discuss are conceptually rather different from each other. All these methods take only the unannotated token stream as input, but from there the methods differ markedly. In the

first, this information is extended with additional information from another source, namely the Internet, gained through data mining. The second approach is to use standard machine learning methods for unsupervised learning, such as the k-means algorithm or Self-Organising Map, to identify structure in the data.

4.2.1 Mining available knowledge from the Web

As a source of data the World Wide Web has obvious benefits. There is a vast amount of information readily available through search engines, and there are obvious disadvantages. Getting the information that is actually wanted is far from trivial, and what information there is can be contradictory. The application of machine learning methods to extract meaningful knowledge from the web is called 'web mining'.

The idea is that by extracting information from the web, parts of the token sequence could be separated out and automatically labelled. Of course, this would require knowing the mapping between sensor tokens and objects. An example of a website that might then be useful is *http://ehow.com*, which helpfully illustrates not only the sequence of steps involved to perform an activity (such as making coffee), but also the objects used (e.g., coffee machine, coffee beans, cup) [105, 110]. For example, suppose that token 'K' appears in the sequence, and that this is a token representing 'kettle is switched on.' By searching for the word 'kettle', various Web pages are found that include mention of tea making.

However, there are also lots of other Web pages and the different possibilities from these Web pages need to be selected judiciously, for example by applying information retrieval to extract useful text in these Web pages. A weighting scheme such as term frequency-inverse document frequency (TF–IDF) [155] is often used to determine the importance of the text. TF–IDF is commonly used in information retrieval to determine the importance of a word to a document in a corpus. This is what the web will find for us such as that the relation between kettle and 'tea making' will have a

high TF–IDF score since the word kettle has a high frequency that pertains to tea making. Of course, bridging the gap between the website and the token evidence is far from trivial, and could involve interesting challenges in natural language processing and image analysis as well as web mining itself.

Another variation of this is to use the web to learn behaviours that share some form of commonality [50, 155]. For example, suppose that there are labelled examples of 'sweeping the floor' in the source domain and other unlabelled cleaning activities in the target domain. In this case, given some sensor readings from the target domain, web mining can try to extract similar tasks such as 'vacuuming' or 'doing laundry' and give those as possible behaviours to be identified. Various similarity functions (such as the cosine similarity metric [155], maximum mean discrepancy [51, 157], Kullback-Leibler divergence [50, 149] or Google similarity distance [50]) can be employed to measure the similarity between two different activities.

Given a set of activities, the basic idea of using the web is to identify objects that are used when performing particular activities. However, relating these clusters of sensor readings to the activities is difficult, as different people have their own ways of performing activities. In order to address this, the resulting output from the mined model is usually mapped to a supervised algorithm (such as the hidden Markov model [110] or a dynamic Bayesian network [149]), which is then used to recognise behaviours. Some work also proceeded to add common-sense information in the reasoning about the human state (e.g., 'the inhabitant is in the kitchen making tea') [109].

Although mining activity models from the web could identify sensors that are related to activities, this approach assumes that the inhabitant performs the activities according to the model defined by the Web, which may not always be true since different people have their own preferences for doing things. Take the tea making for example, some people don't put in any milk at all and add sugar instead, while others

make the tea without any of them. So, training a learning algorithm on the activity models mined from the web may not be the actual representation of the inhabitant's behaviours.

4.2.2 Finding structure in the sensor stream

Another method to automatically learn the mapping between sensor outputs and behaviours is to use standard machine learning approaches to unsupervised learning, namely the k-means algorithm and Self-Organising Map (SOM) [65]. Both algorithms have the same aim, which is to find sequences of sensor readings that are 'similar'.

The k-means algorithm attempts to identify a set of k centroids or cluster centres that best represent the data, where k is a parameter that is chosen in advance. The positions of the centroids are initially chosen randomly. The algorithm then updates the positions according to the data and iterates until the centroid locations stop moving, whereupon each datapoint is represented by its closest centroid. For behaviour recognition, the centroid would represent the 'idealised' behaviour.

Huynh and Schiele [54] use the output of clustering to determine the selection of features for each activity. Nguyen et al. [102] employ the clustering method to separate out the usual and unusual events while Hein et al. [44] use clustering to identify abstract high-level activities. Although this method is able to identify clusters of behaviours, the difficulty lies in determining the k parameter beforehand, where prior knowledge of the number of behaviours needs to be known.

Another variation to the k-means clustering algorithm is the Self-Organising Map (SOM) [65], which is an unsupervised learning method. The basic idea of the SOM is to produce a representation of the input in a low dimensional set of map nodes or neurons (often 2D), which are activated proportional to the distance between their weights and the input. The learning process is competitive in the sense that neurons similar to the inputs are trained, while the others are ignored. The algorithm

modifies the weights of the winning neurons and its neigbours in such a way that, after training, the outputs are arranged in a topological order so that neurons that are closer to one another represent similar features. During learning, the weights are iteratively adjusted until the maximum number of iterations is reached.

In the smart home context, the inputs to the SOM could be a set of tokens that represent particular behaviours. When a new sequence is presented to the SOM, the output can be either the entire pattern of neuronal activations, or just the index of the most active node (as in the k-means algorithm), which can be used to represent the behaviour [63, 154, 156].

In the work of both Laerhoven [70] and Krause et al. [69], they first apply the Kohonen self-organising map to reduce the number of dimensions of sensor data and further perform k-means clustering for behaviour classification. Dahmane and Meunier [24] used the SOM to identify abnormal behaviours, which were determined when the distance between the winning neuron under normal condition and the current winning neuron exceeded a predefined threshold value. An example of a SOM is shown in Figure 4.2.

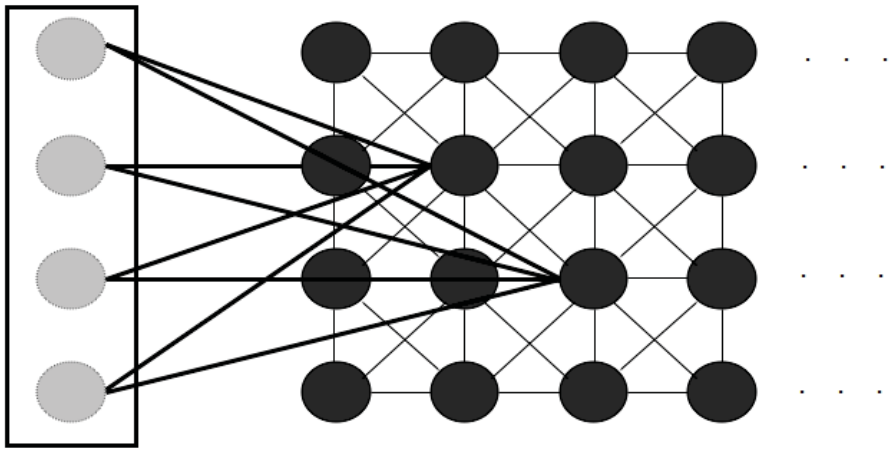


Figure 4.2: A schematic of the Self-Organising Map. Inputs (in grey) are fed to each neuron (in black) using weighted connections. There are connections between neurons in a local neighbourhood to ensure that neurons that are close together represent similar inputs. Learning consists of modifying the weights.

Although the SOM preserves the topological structure (i.e., the neighbourhood relations) of the data, it is not suitable where the inputs are not of fixed dimensionality since it uses a topology of fixed dimensionality. This means that prior to training, the dimensions of the inputs need to be defined. This may not be practical for applications where the data sequences are of variable length.

In the next section, we describe a method that can address the variable-length data sequences without any prior knowledge about the inputs. This method, which is also our method of choice, exploits the fact that a behaviour is a pattern that is a repeated set of tokens, and can thus be identified using any algorithm that exploits redundancy in the input stream, i.e., a compression algorithm.

4.2.3 Exploiting redundancy in the sensor stream

Compression has been a topic of interest since the birth of information theory in the work of Shannon [122], with the aim of reducing the size of data for storage and/or for transmission. Compression exploits the repetition in the data by building a dictionary of codewords, and then replacing each incidence of the word with a codeword in the dictionary, with shorter codewords being used for frequent words, and longer codewords for less frequently used words. Provided that the codewords are shorter than the words in the codebook, compression is achieved. Most compression algorithms require no prior knowledge about the input data stream and can deal with codewords of different lengths without problem.

Compression algorithms can be broken down into two encoding schemes: lossless and lossy. The former allows the perfect reproduction of the input from the compressed stream, while the latter does not. Clearly, for most computer files, lossless compression is what is required. However, for images and sound, where knowledge of the type of data to be stored and the capabilities of humans to process their sight and sound stimuli enables more compression to be achieved by suppressing data that

would not be detected by humans anyway, such as very high frequencies in sound recording. It can also help to deal with noise and variability, provided that the component that is lost is the noisy part of the data.

Lossless compression

In lossless compression, the decompressed data is the exact replication of the original data. One common application of lossless compression is the ZIP file, which effectively compresses large document files in order to reduce the file size. Two popular lossless compression techniques used for text are the statistical-based approach, which is based on context modeling, and the dictionary approach.

(a) The statistical approach

The main idea behind the statistical approach is to predict the occurrence of the next symbol based on the symbols that precede it. One well-known statistical approach is *Prediction by Partial Matching (PPM)*, which makes use of finite-context models of order k (where k is the number of preceding symbols used) for character prediction [19, 18].

PPM uses a table to keep track of the number of times the characters are seen in the input stream. The table is updated adaptively as each character is read from the input stream. Since prediction probabilities are used to predict the upcoming character, each context model maintains its own probability distributions.

The encoding of a character begins at the highest k model to see if the character has occurred in the current context. If it has occurred before, then the count is used to encode the character in that context. However, if the character has never been encountered before in the context, then an 'escape' symbol is transmitted to tell the decoder to use the model of $k-1$, and the process is repeated until a model is reached and the character is predicted according to the probability distribution of that model [18]. There are many methods to determine the probabilities for the

Order $k = 2$			Order $k = 1$			Order $k = 0$		
predictions	*freq*	*prob*	*predictions*	*freq*	*prob*	*predictions*	*freq*	*prob*
mo → u	3	3/4	m → o	3	3/4	→ m	3	3/20
→ *esc*	1	1/4	→ *esc*	1	1/4	→ o	3	3/20
ou → s	3	3/4	o → u	3	3/4	→ u	3	3/20
→ *esc*	1	1/4	→ *esc*	1	1/4	→ s	3	3/20
us → e	3	3/4	u → s	3	3/4	→ e	3	3/20
→ *esc*	1	1/4	→ *esc*	1	1/4	→ *esc*	5	1/4
se → m	2	2/3	s → e	3	3/4			
→ *esc*	1	1/3	→ *esc*	1	1/4			
em → o	2	2/3	e → m	2	2/3			
→ *esc*	1	1/3	→ *esc*	1	1/3			

Figure 4.3: Illustration of Prediction by Partial Matching (PPM) method based on order-2 context models. The 'esc' symbol stands for 'escape' event and is used when a novel character is encountered and is not seen in the context of the current model.

'escape' probability [5, 19, 96, 121], but this is beyond the scope of this thesis.

If a character has never occurred in any context at all then it is processed at the bottom-level, i.e., $k = -1$, where all characters are assigned with equal probabilities. For example, if there are 10 characters, then each character in the $k = -1$ model is assigned a probability of 0.1. Figure 4.3 illustrates the PPM method of three models with $k = 2, 1$ and 0 for the input string *mousemousemouse*. As the figure shows, all the previously seen contexts are shown along with the frequency counts and probabilities for each k model.

(b) The dictionary approach

The dictionary approach replaces phrases in the text with a pointer to an earlier occurrence of the same phrase, effectively creating a 'codebook' of common phrases. One popular dictionary-based compression algorithm is the Lempel-Ziv-Welch (LZW) algorithm [144], a family of Lempel-Ziv compression (LZ78) [159]. LZW begins with single characters and adds them into the dictionary, and then examines the token stream character by character until a string that is not in the dictionary is found.

This is added into the dictionary, and the search for dictionary strings recommences, starting with the last letter of the string that has just been added.

As an example, the second time the phrase '*mo*' is seen in the input string *mouse-mousemouse*, it will take the index of '*mo*' found in the dictionary and extend the phrase by concatenating it with the next character from the sequence to form a new phrase ('*mou*'), which is later added to the dictionary. The search then continues from the token '*u*'. The phrases in the dictionary are basically indexed by a code and the code representing the phrase can be found by looking it up in the dictionary. For a complete description of LZW, see [144].

Within the context of smart homes, Das et al. [25] use compression to predict the inhabitant's movement (location) in the home. They partitioned the home into different zones (rooms), with each zone represented by a symbol. A sequence of zones are generated when the inhabitant moves from one room to another. They first used the LZ78 method to build a dictionary that represents the inhabitant's movements in the home. The statistical method is then used to predict the inhabitant's next room's location based on the probability distributions of the phrases in the dictionary.

Cilibrasi and Vitányi [17] propose a similarity distance based on the length of compressed data files, the 'normalised compression distance (NCD)' and then use hierarchical clustering to identify clusters within the data based on the most dominant shared feature, which is computed from the lengths of the compressed data files. Their approach aimed at rather longer sequences such as music files and genome data.

4.3 Description of Our Approach

We present an unsupervised learning approach based on compression and text analysis that can automatically cluster the unlabelled data and segment the data stream into behaviours, without any prior labelling of the data stream. The main reason why a set of activities form a behaviour is because they are repeated. For example, making

a beverage is a behaviour, since it might well be repeated several times a day, and showering is a behaviour because it is probably repeated daily. However, answering a phone call while having dinner is not a behaviour since it does not occur frequently. Based on this reasoning, it seems clear that we can identify behaviours as a set of sensor patterns that are seen repeatedly in the data, which can be considered as 'redundant' in the representational sense and therefore detectable. Compression can be used to exploit the redundancy in the sensory stream, which we define to be behaviours.

To illustrate how compression can be achieved, we represent the data stream as a sequence of tokens (e.g., letters from the Roman alphabet), where a token could be the direct representation of the current sensor states being triggered (e.g., bathroom light is turned off, kitchen door is opened, microwave is turned on, etc.). Hence, a behaviour can be identified in the data stream as a repeated set of 'words', albeit with variations, during the inhabitant's daily activities.

Both dictionary and statistical approaches can be applied to exploit the redundancy in the unlabelled sensor stream. Since our interest is to identify common longest repeated subsequences and the dictionary approach can process several characters at once, unlike the statistical approach which can only use higher-order context models, we have therefore chosen to use the dictionary-based approach. The method that we used is based on the Lempel-Ziv-Welch (LZW) algorithm [144], which exploits the repetitions by creating a codebook of potential patterns (i.e., 'words'), which is defined by a set of prototype vectors of clusters.

However, patterns (e.g., 'hello') often do not repeat perfectly each time they are seen, such as certain tokens being additionally present (e.g., 'he$y$llo') or absent (e.g., 'helo'), tokens being in a different order (e.g., 'helol'), or minor variation in a token (e.g., 'hallo'). We hence want to recognise variations in the patterns. Unfortunately, the LZW method does not generalise to variations of the input. To allow for these

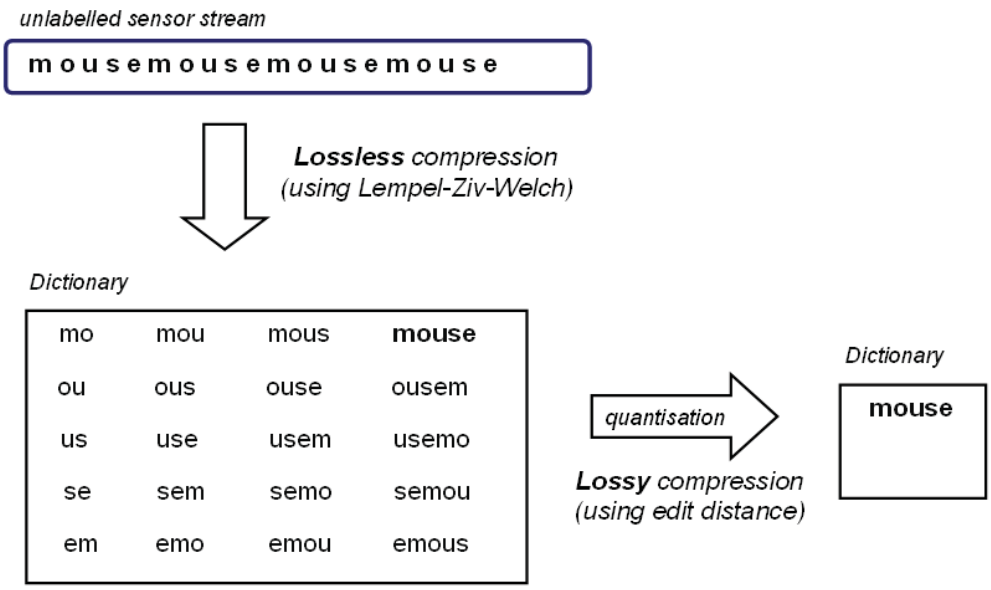


Figure 4.4: Lossless compression is first performed on the unlabelled data stream using the Lempel-Ziv-Welch (LZW) method, which creates a dictionary of phrases. There are codewords associated with each substring in the dictionary, but they are omitted for clarity. The dictionary is then quantised using lossy compression, which is based on edit distance. The word '*mouse*' after quantisation represents a pattern.

kinds of variability, a lossy compression is more suited to our problem that allows variations of a word to be matched in the dictionary. We do this by extending the LZW method to perform lossy compression based on edit distance [139].

Figure 4.4 shows an overview of our approach. Standard lossless compression of the data stream is performed using the Lempel-Ziv-Welch (LZW) algorithm [144], which produces a dictionary of phrases that are seen most frequently in the data. This will include a number of variations of each word, and we then edit this dictionary to produce individual prototype datapoints. Based on this reduced dictionary, lossy compression (i.e., allowing some relatively minor changes between the input and the dictionary words) is used to find the closest matching word in the dictionary.

4.3.1 Selecting patterns from an unlabelled data stream

The only input that we expect to see for our approach is the unannotated data stream. The LZW algorithm is used to parse this and to identify potential sequences that can be added to the dictionary. The dictionary produced by LZW is typically large, since it contains everything that has been learned during training, including all the substrings of each dictionary word (see Figure 4.4).

To identify patterns, we are only interested in the longest common words in the dictionary. To illustrate this, for now we use the word 'mouse' to represent the tea making behaviour, where token '*m*' could be a sensor event on the kitchen door, '*o*' that the tap was running, '*u*' that the kettle was switched on, '*s*' that the fridge was opened and '*e*' that the teapot was in use. Since LZW organises around a dictionary by concatenating a phrase found in the dictionary with the next character from the token sequence, this will result in the dictionary containing many similar phrases such as '*mo*', '*ou*', '*us*', '*mou*', '*ous*', etc. We want to identify the longest common 'words', arguing that they represent patterns; thus we want '*mouse*' to represent one complete tea making behaviour, rather than '*mo*' and '*use*' separately. Thus, the next step is to perform a reduction of the dictionary so that it contains only the prototype words.

Our approach to this problem is to consider a modification to the LZW algorithm that enables it to perform lossy compression, so that the dictionary after reduction is biased towards words which are as long as possible, while still being common in the data stream.

The aim of the dictionary reduction is to find a single prototype vector for typical data entries. In fact, we also wish to use the quantisation to ensure that allowable variations on the 'word' in the dictionary are recognised as being equivalent to the dictionary exemplar during use of the algorithm, which is a related problem. We have chosen to approach both of these problems by using the edit distance [139] (also known as Levenshtein edit distance), which is a measure of the similarity between

pairs of strings.

The Edit Distance

The edit distance can be efficiently computed by dynamic programing and is commonly used for biological sequence analysis [128], spell checkers [10], and plagiarism detection [158]. It works by computing the minimum number of actions required to transfer one string into the other, where an action is a *substitution*, *deletion*, or *insertion* of a character into the string. For example, given $p =$'rains' and $q =$'rein', the edit distance to change 'rains' into 'rein' is 2 since we only need to substitute the second letter 'a' in 'rains' with 'e', and delete the letter 's' from 'rains'.

An algorithm for computing the edit distance usually uses a two-dimensional matrix (i.e., size $(|p|+1) \times (|q|+1)$, where $|p|$ and $|q|$ are the lengths of the two strings p and q) to keep track of the edit distance values. The algorithm begins by first initialising the first column to have values $[0, 1, 2, \ldots, |p|]$ and likewise for the first row to have the values $[0, 1, 2, \ldots, |q|]$, as shown in Figure 4.5. The entry for each remaining cell in the matrix is computed by following Equation 4.1:

$$\min \begin{cases} dist_{i-1,j-1} + \begin{cases} 0 & \text{if } \; p[i] = q[j] \\ 1 & \text{otherwise} \end{cases} \\ dist_{i-1,j} + 1 \\ dist_{i,j-1} + 1 \end{cases} \tag{4.1}$$

where $dist_{i,j}$ is the element of the i^{th} row and j^{th} column in the $dist$ matrix. The value on the last row and column in the matrix (shaded in Figure 4.5) represents the distance to change 'rains' into 'rein'.

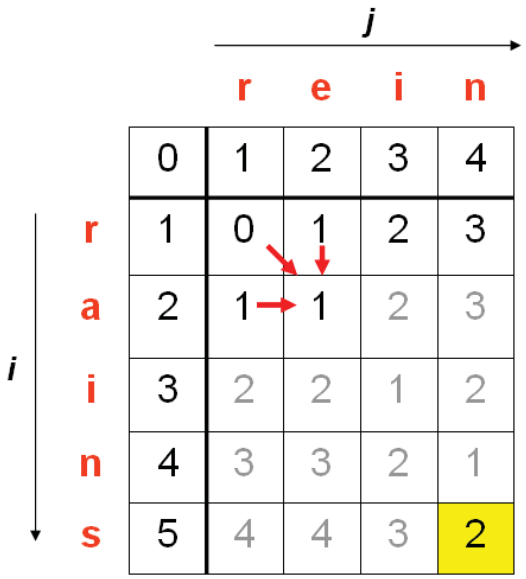


Figure 4.5: An illustration detailing how edit distance can be efficiently computed using dynamic programming to change the string 'rains' into 'rein'. The entry in each cell is computed by following Equation 4.1.

The algorithm

Based on this distance, we are now in a position to quantise the dictionary. We do this by picking the first phrase from the dictionary, and finding its 'neighbouring' phrases, i.e., those that are edit distance 1 away. We have found experimentally that the most effective way to quantise the dictionary is to use edit distance = 1 since increasing the edit distance, so that for values of edit distance > 1, very noisy examples are likely to be incorrectly identified as words. This is described further in Section 4.4.1.

From this set, the word with the highest frequency count and longest word length is selected as the potential pattern, and the algorithm iterates until the pattern does not change. However, when there is a tie between words that have the same frequency count and longest word length, the algorithm will select the word with the highest frequency count. Algorithm 1 shows the steps of using edit distance for lossy compression. The average computational complexity of our method is $O(Tn^2)$, where T is the number of words, and n is the maximum length of a word.

Once the prototype 'words' for the dictionary have been selected, the next task is

Algorithm 1 Lossy Compression using Edit Distance

Input: LZW dictionary D
Initialisation: m = number of words in D and EndOfDictionary = false
  while (EndOfDictionary == false) **do**
    for $k = 1$ to m **do**
      for $i = k + 1$ to m **do**
        $\omega_{array} \leftarrow$ Using Eq. 4.1, find words where $dist(w_k, w_i) = 1$
      end for
      if $\omega_{array} \neq \phi$ **then**
        $\omega_{maxArray}$ = get word from ω_{array} that has max(freq count + word length)
        if length ($\omega_{maxArray}$) > 1 **then**
          delete all words in $\omega_{maxArray}$ from D except the word with max(freq count)
        end if
        delete words in ω_{array} from D
        m = number of words in D
      end if
    end for
    if $k == m$ **then**
      EndOfDictionary = true
    end if
  end while
  output quantised dictionary D

to use these prototypes to identify words in the data stream. This is described next.

4.3.2 Recognising patterns in the unlabelled data stream

Given a dictionary, we need to parse the data stream to recognise dictionary exemplars and allowable variations on them. To formulate the problem, given the data stream $S = \{d_1, d_2, d_3, \ldots d_n\}$ and the quantised set of words $D = \{w_1, w_2, w_3, \ldots w_l\}$ in the dictionary, we try to find a match w_i for some subset of S, and then make that subset maximal given w_i, so that the distance between w and d is minimal.

One of the challenges in segmentation is that the presentation of a pattern will almost always vary between instances of the same behaviour. Using the same example of the word '*mouse*' to represent the tea making behaviour, both words, '*moue*' (i.e., missing token, which literally means that the tea is made without the milk) and '*mosue*' (different orders of token activations, where somebody takes the milk out

from the fridge before turning on the kettle) could well represent the same behaviour. For this reason, we use the edit distance to identify the matches between the token sequence and the set of words – that correspond to behaviours – in the quantised dictionary. Segmentation of the data stream can be summarised as a three-step procedure:

1. Compute the matches between each w_i in the quantised dictionary and the data stream S using edit distance.

We compute the matches for each w_i in the quantised dictionary and the data stream S using edit distance. The distance values are stored in a two-dimensional matrix. The computation of the edit distance is the same as we use to perform lossy compression, as described in Section 4.3.1. The only difference is that, here, the value for all the entries on the first column are initialised as 0 (shown in Figure 4.6). This is to enable an approximate match for some subset of words in S.

2. Select the maximal 'word' in S with edit distance below some threshold ϵ.

A threshold ϵ is chosen to control how much variation is allowed between word samples. Based on experiments, the ϵ value that we used is half the word length of the word in the dictionary and we round down those values which are not integer. Referring to the example in Figure 4.6, the ϵ value for the dictionary word 'mouse' is floor(5/2) = 2. Looking at the figure, there are 6 columns in the last row with distance less than 2, which are columns 3, 4, 5, 10, 11 and 12. This does not mean that there are 6 'words' that match for the word 'mouse'.

For that, we need to determine among the identified columns (i.e., last row of columns 3, 4, 5, 10, 11 and 12) the exact ending point of the word boundary. We first determine if the columns are in steps of 1. Looking at the sequence, there is a gap

between columns 5 and 10, which means that there are 2 words that match the word 'mouse'. The end point of the first word (i.e., the last row of columns 3, 4 and 5) can be found by taking the minimum distance, which is column 4 with the minimum distance of 1. As for the second word (i.e., the last row of columns 10, 11 and 12), the end point is therefore column 12, which has a minimum distance of 0.

An array is used to keep track of word boundary. This array is first initialised as '0' and the length of the array is set according to the length of the sensor stream. When an end point of a word is found, the index in the array is assigned a value according to the position of the matched word in the dictionary. For example, if the word matches the first word in the dictionary then a value '1' is assigned, '2' for the second word and so on. Using the example shown in Figure 4.6, a value 1 is assigned for indices 4 and 12 in the array since both words match the first word in the dictionary.

3. Perform backward-traversal to determine the starting point of the word boundary.

Given that we have found a match, the question is then how to determine the starting point of the word. We can distinguish two types of match: perfect matches (i.e., matches with a distance value of 0, such as the last row of column 12 in Figure 4.6) and matches with errors (i.e., those with a distance value greater than 0, but less than ϵ, such as the last row of column 4 in Figure 4.6). When a perfect match is found, the starting point of word boundary is determined by moving backwards through the word length. In the example, the word length for '*mouse*' is 5 and thus we can move backward 5 steps. Since the starting point and end point for this word are at columns 8 and 12, a value '1' is assigned for the indices 8 through 12 in the array.

However, if the edit distance is 1, i.e., there is an error, then this approach is not sufficient, as it is hard to know if there is a missing letter (e.g., '*mose*') or extra letter

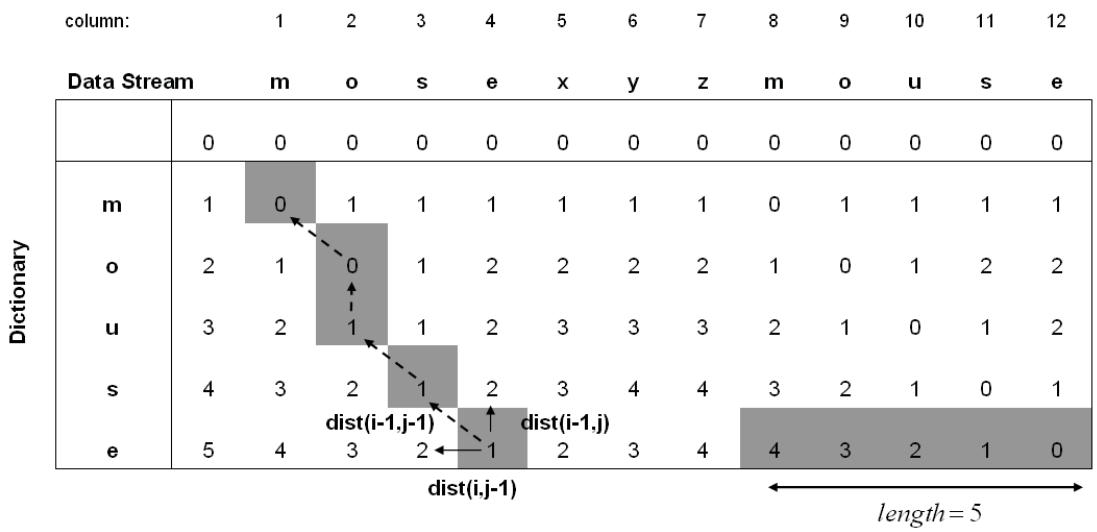


Figure 4.6: Illustration of how backward traversal is performed on the distance matrix to identify the starting point of the word boundary. When a perfect match is found i.e., when the distance is 0 (column 12), the number of steps to move backward is based on word length. When there is an error (column 4), the algorithm recursively traverses the distance matrix back and upward by finding the minimum distance (shown in dashed arrow). For details, see the text.

included (e.g., '*mo*p*se*'). An example of this is shown in column 4 of Figure 4.6. In this case the starting point of word boundary can be identified by traversing the *dist* matrix back and upward by finding the minimum distance of $\min(dist[i-1, j-1], [i-1, j], [i, j-1])$. By traversing back and upward i.e., following the shaded paths in Figure 4.6, we can identify the starting point of word boundary and thus segment the data stream according to the 'words' in the quantised dictionary. Note that when traversing the matrix, priority is given to following the minimum distance on the diagonal (i.e., $dist[i-1, j-1]$) path, followed by $dist[i-1, j]$ and $dist[i, j-1]$.

Given that we have identified the start and end point of the word (i.e., columns 1 and 4), the indices 1 through 4 in the array are assigned a value of '1', which corresponds to the first word in the dictionary. The array for the example shown in Figure 4.6 would be: $[1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]$. The values for the indices in the array remain as '0' if there is no match (e.g., the word 'xyz' in columns 5, 6 and 7).

The use of edit distance in this thesis is two-fold: (1) to perform lossy compression on the dictionary produced by the LZW algorithm (Section 4.3.1), and (2) to segment the unlabelled token stream into behaviours by identifying the matches between the 'words' in the quantised dictionary and the token stream (Section 4.3.2).

4.4 Experimental Results

In this section, we describe our experiment setup and the datasets used. In fact, the datasets that we have used do have labels, and thus are suitable for supervised learning. In all the experiments reported here, we used the annotation in the training set only to attach a recognisable label to the words in the quantised dictionary, and used the annotation of the test set as a ground truth.

The first experiment is to investigate how the edit distance threshold can be selected for dictionary quantisation. The objective of the second experiment is to test the proposed unsupervised approach based on compression and edit distance to identify words that are seen frequently in the sensor stream and then segment the sensor stream into words by finding the closest words in the dictionary using edit distance. We evaluate the performance of our method by comparing it with an unsupervised method based on the self-organising map (SOM) on two datasets: one where the inputs are of fixed dimensionality (Fisher's iris data [33]) and another on smart home dataset where the behaviours are of variable length (MIT PlaceLab [133]). In the final experiment, we use the unsupervised compression-based method as a way to provide labels to training data for a supervised algorithm. This is done by training hidden Markov models based on the words in the quantised dictionary. We evaluate the effectiveness of our method by training a baseline supervised classifier using HMMs as in Section 3.5.1.

The main statistic that we are interested in is *recognition accuracy*, which is the ratio of the total number of activities correctly identified by the algorithm divided by

Test Sets	Length of Token Stream (Training Set)	Size of Dictionary Before Quantisation	Edit Distance = 1		Edit Distance = 2	
			Size of Dictionary After Quantisation	Recognition Accuracy	Size of Dictionary After Quantisation	Recognition Accuracy
1st Set	1672	852	57	81%	15	35%
2nd Set	1456	768	54	76%	13	52%
3rd Set	1688	865	53	90%	19	70%
4th Set	1561	813	49	91%	15	61%
5th Set	1563	806	47	84%	17	59%
6th Set	1592	826	47	79%	15	44%
7th Set	1630	846	50	86%	15	65%
8th Set	1478	779	43	79%	15	57%
Average				**83%**		**57%**
Standard Deviation				**5.4**		**11.4**

Table 4.1: Results on the effect of edit distance for dictionary quantisation. The size of dictionary refers to the number of words in the dictionary.

the total number of activities used for testing.

4.4.1 Experiment 1: The effect of edit distance for dictionary quantisation

Since we want to ensure that the quantised dictionary acquires a good representation of the behaviours, we need to control the amount of loss in the entries in the dictionary. Thus, in this experiment we investigate what threshold value is best for dictionary quantisation.

Using the MIT PlaceLab dataset, we ran the LZW algorithm on the training data and the output is a set of words in a dictionary, including the number of variations of each word. Based on this dictionary, we quantise the dictionary by setting the edit distance threshold = 1. Once we have the quantised dictionary, we parse the tokens from the test set into the set of quantised words in the dictionary. The token streams which have been segmented are validated against the ground truth annotations on the test set. We then repeat the same experiment by increasing the edit distance threshold to 2. The results are shown in Table 4.1.

As the table shows, an increase in the edit distance threshold results in a higher amount of information loss and a greater reduction on the dictionary size. The poor percentage accuracy can be explained by the fact that many good representative examples have been discarded and that the very noisy examples have a higher chance

of being incorrectly identified as words, which means that the quantised dictionary does not acquire a good representation of the behaviours. Thus, the most effective way to quantise the dictionary is to use an edit distance threshold = 1, which can be seen from the results shown in Table 4.1, producing an average accuracy of 83%.

4.4.2 Experiment 2: Comparison with the Self-Organising Map (SOM)

We report experiments on two datasets. The first is a dataset where the inputs are of fixed dimensionality (Fisher's iris data [33]). This enables us to compare the results with other unsupervised learning algorithms, here the Self-Organising Map (SOM). The second dataset is the sensor readings for a smart home where the behaviours that should be identified are of variable length (MIT PlaceLab [133]). In this experiment, we also use the SOM as a baseline to understand how effective our proposed method is.

We chose the SOM because it can also be considered as building a codebook of prototype vectors. The difference is that the SOM determines the similarity between the input vector and codebook vectors by minimising the Euclidean distance, while our method minimises the edit distance between the input vector and the 'words' in the dictionary.

Experiment 2.1: Fixed-length Data Sequences: Fisher's Iris Dataset

Fisher's iris data [33] is widely used for cluster analysis and is available through the University of California, Irvine (UCI) machine learning database [34]. The data has four attributes, which are measured on fifty iris specimens from each of three species: Setosa, Vericolor and Virginica.

We partitioned the iris data into 5 equal partitions and used 5-fold cross validation,

training 4 partitions and using the last one for testing. The recognition accuracy is calculated by averaging the accuracies on each run. Since the iris data are numeric, to use it with our compression approach, we transformed the data into a series of tokens using quantisation by treating each dimension separately and breaking it into ten equally spaced steps, with different letters for each dimension. Thus, attribute instances whose values were in the range $[0, 0.1)$ were assigned token 'a', those in $[0.1, 0.2)$ were assigned token 'b' and so on.

The training of the SOM was in batch mode, the learning rate was set to 0.05 and it was trained for 100 epochs. We have tried different learning rates and 0.05 works best for this dataset. The SOM is a 6×6 map shown in Figure 4.7. Since the size of the training and testing examples are small, the map size is determined empirically so that the differences in the clusters can be observed. Once we have trained the SOM and labelled the nodes according to the target classes (as can be seen in Figure 4.7), we test the SOM using the test set. Recognition accuracy in the SOM is calculated by determining the nodes that best match the inputs from the test set i.e., the ratio of the total number of class labels from the test set that match to the labels in the map divided by the total number of instances used for testing.

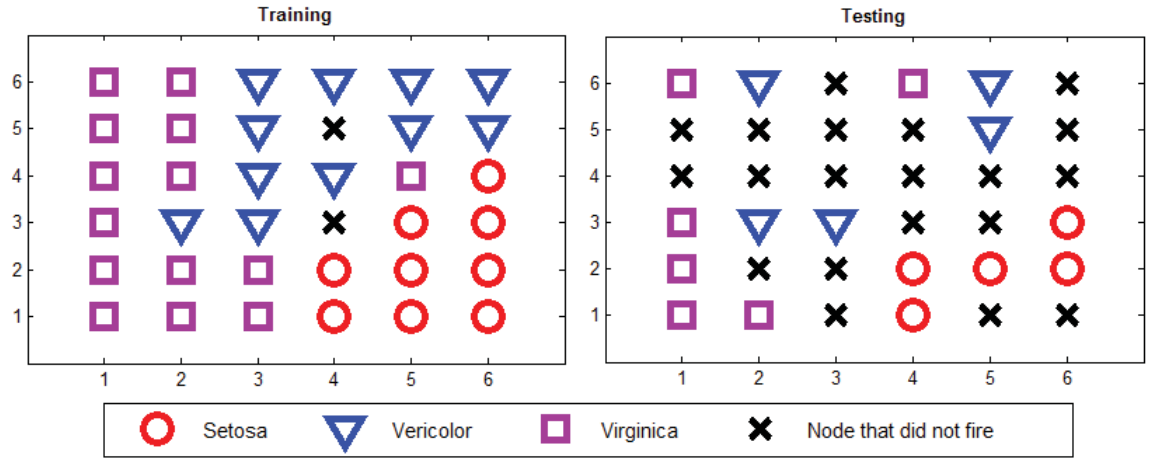


Figure 4.7: The SOM output on Iris data that shows three different clusters: Setosa, Vericolor and Virginica. The example is based on the 1st test set. *Left*: Training data. *Right*: Test data.

Test Sets	Recognition Accuracy		
	Self-Organising Map (on original IRIS data)	**Our Proposed Method** (compression and edit distance)	**Self-Organising Map** (on transformed IRIS data)
1st Set	90%	90%	90%
2nd Set	97%	90%	90%
3rd Set	93%	87%	83%
4th Set	93%	93%	87%
5th Set	97%	93%	90%
Average	**94%**	**91%**	**88%**
Standard Deviation	**3**	**2.5**	**3**

Table 4.2: A comparison of results between the Self-Organising Map (SOM) method and our proposed method on Fisher's iris data.

The results presented in Table 4.2 showed that our algorithm did very well given that it did not know that the input data were four dimensional, which the SOM obviously did. However, we were interested in how much of the error was caused by the quantisation that was required to turn the data into tokens. We therefore re-transformed the tokens back into numbers, by replacing each token by the median of the values range for each token (e.g., token 'a' is transformed into 0.05, token 'b' as 0.15, etc.). The results in the final column of Table 4.2 show the results when the SOM is trained on these quantised data, which leads to a significant reduction in the accuracy of the SOM, and makes the accuracy of our method comparable to the baseline.

The choice of how to quantise the data into tokens is an important one, providing a trade-off between compression rate and accuracy. One can use smaller range values for the transformation to achieve a higher recognition accuracy or using larger range values to achieve a higher compression rate. The choice of 0.1 used in this experiment is empirical and results have shown that it performs acceptably well in Fisher's iris dataset. Note that it is beyond the scope of this study to investigate other quantisation values since the values could be different for different datasets.

Experiment 2.2: Variable-length Data Sequences: The MIT PlaceLab Dataset

One aim of our method is that it should work well where the inputs are of variable length, such as behaviour recognition and speech recognition, where conventional unsupervised learning algorithms do not do well. To demonstrate it on this type of problem, we used a real smart home dataset from the MIT PlaceLab [133], which is described in Section 1.5.

We evaluated the performance of our method by comparing it with the SOM. The data from the sensor stream is presented to the SOM by taking the frequency count of each sensor activation using a window that slides over the data. The size of the window is determined by taking the average number of sensor observations that describe the behaviours, which is 3.

The training of the SOM was in batch mode and the learning rate was set to 0.05. The SOM was trained for 500 epochs and the SOM is a 10×10 map shown in Figure 4.8. Since there are more training and testing examples in the smart home dataset, the map size is determined by following the heuristic rule in [66], which is approximately $5\sqrt{n}$, where n is the number of training examples. Due to the different number of training examples in each partition of training set, the average across all the training set is taken for the value n which is equal to 410. Recognition accuracy in the SOM is calculated by determining the nodes that are the best match according to the behaviours in the map after training i.e., the ratio of the number of activity labels from the test set that match to the labels in the map divided by the total number of activities used for testing.

Note that not all the nodes in the map fired during training. One problem with this is that when these nodes fire when data from the test set is presented, there is no way to determine which behaviours these nodes belong to. Referring to Figure 4.8, during training (left figure) the node at location $(6, 9)$ does not fire. When the node

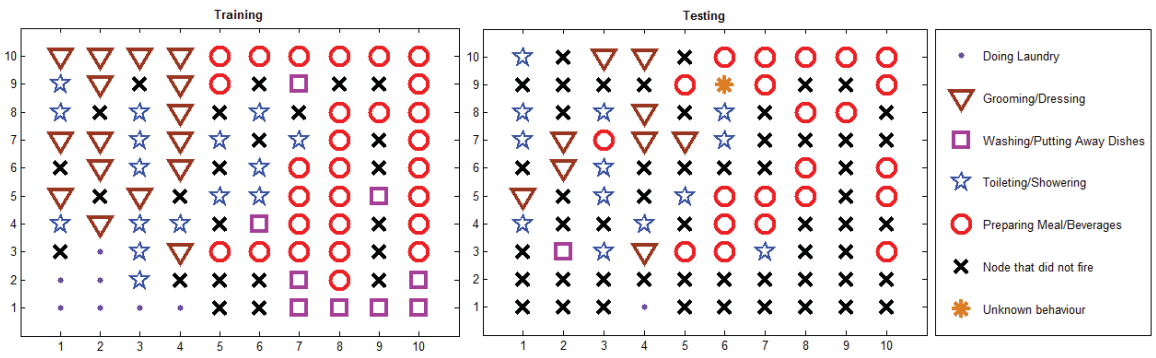


Figure 4.8: The SOM output on smart home data that shows five different behaviours: doing laundry, grooming/dressing, washing/putting away dishes, toileting/showering and preparing meal/beverages. The example is based on the 3rd test set. *Left*: Training data. *Right*: Test data.

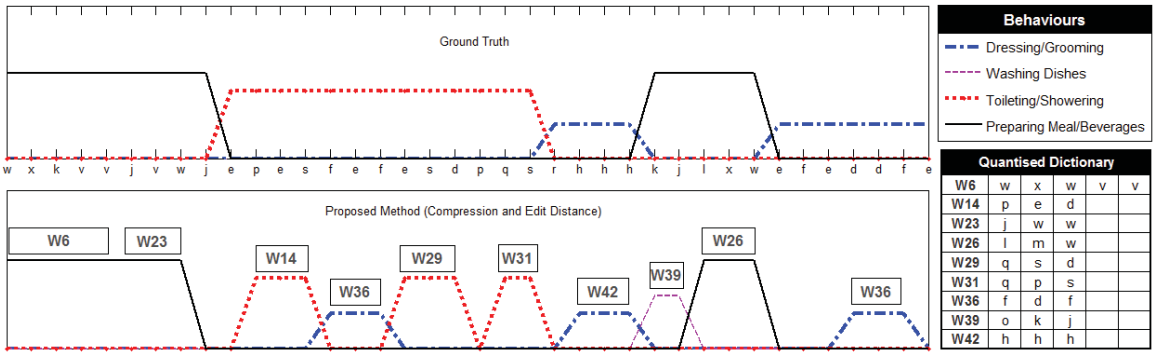


Figure 4.9: Visualisation of the output of the ground truth (top) and proposed method (bottom). The lower case letters on the x-axis show the sensor readings, while the y-axis shows the potential behaviours (which corresponds to the top-right of the figure). The notation 'W6' refers to one of the words in the quantised dictionary (shown on the bottom-right of the figure). The example is based on 5 activity examples on the 3rd test set.

fires during testing (right figure), it is difficult to know which behaviour this node recognises. We have thus labelled this as 'unknown behaviour'.

To demonstrate our compression-based method, we first used the LZW algorithm to build a dictionary of substrings based on the tokens in the training set. Once the prototype words for the dictionary have been identified, we parse the tokens from the test set to recognise dictionary exemplars. A visualisation of the output of the system is shown in Figure 4.9. The tokens which have been segmented into behaviours are then validated against the ground truth annotations on the test set.

Test Sets	No. of Activity Examples	No. of Activities Correctly Identified	Unidentified Activities	Recognition Accuracy
1st Set	31	25	2	81%
2nd Set	54	41	7	76%
3rd Set	20	18	0	90%
4th Set	33	30	0	91%
5th Set	49	41	3	84%
6th Set	34	27	2	79%
7th Set	37	32	2	86%
8th Set	52	41	3	79%
Average				**83%**
Standard Deviation				**5.4**

Table 4.3: Results on unsupervised learning based on compression and edit distance on different training/testing splits. The term 'unidentified activities' refers to the total number of activities that are not identified by the compression-based algorithm.

For our compression-based method, some behaviours were too rare and compression simply could not be achieved, which results in some behaviours (such as 'doing/putting away laundry' and 'washing/putting away dishes') not being identified when building the dictionary. We use the term 'unidentified activities' to refer to the activities in the test set that are not identified by our algorithm, as shown in Table 4.3.

However, it is still instructive to see if there are consistent reasons for these to occur. Instances in these behaviours vary from the usual norm and the behaviour is often interrupted by another event or noise from other sensors. These result in high values in the edit distance and our algorithm is unable to recognise the word. We expect that if there had been more examples of those words, they would have been identified successfully.

The results of the SOM and our compression-based method are presented in Table 4.4. We have conducted a significance test to compare the recognition accuracy of the SOM and our compression-based method. An F-test was first carried out to determine the equivalence of the variances for these two methods. The test statistic is $F = (\frac{S_1^2}{S_2^2})(\frac{\sigma_2^2}{\sigma_1^2}) = \frac{5.4}{8.8} = 0.6136$ with p-value $= 0.27$. Thus, the null hypothesis of equal variances is accepted. A Student's t-test is conducted to test the alternative

Test Sets	Recognition Accuracy	
	Compression and Edit Distance	Self-Organising Map
1st Set	81%	56%
2nd Set	76%	57%
3rd Set	90%	69%
4th Set	91%	56%
5th Set	84%	57%
6th Set	79%	64%
7th Set	86%	75%
8th Set	79%	47%
Average	**83%**	**60%**
Standard Deviation	**5.4**	**8.8**

Table 4.4: A comparison of results between our proposed method based on compression and edit distance, and the self-organising map (SOM) on smart home data.

hypothesis that the average recognition accuracy of the compression-based method is greater than the average recognition accuracy of the SOM. The test statistic is $T = \frac{(\bar{x_1}-\bar{x_2})-(\mu_1-\mu_2)}{S_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} = \frac{(83-60)-(0)}{(\sqrt{53.3})(\sqrt{\frac{1}{8}+\frac{1}{8}})} = \frac{23}{3.65} = 6.3$ with p-value $= 9 \times 10^{-6}$, where S_p is the pooled variance. Thus, the null hypothesis is rejected and we can conclude that the average accuracy of our compression-based method is significantly higher than the average accuracy of the SOM. Our method did well because it can deal with codewords of different lengths, while in the SOM, the size of the input vector is chosen in advance. This means that the number of tokens that are presented to the SOM is preset, here to 3.

4.4.3 Experiment 3: Providing labels to training data for a supervised learning algorithm

The system that we are proposing will essentially report which behaviour is identified at each time. However, this might not always be sufficient, depending upon what the aim of the smart home is. For example, suppose that we wish to add other information, such as context [39], or to detect abnormal behaviour. Neither of these would be easy to do using the current method. An alternative is to use the unsupervised learning approach as a way to provide labels to training data for a supervised algorithm in a bootstrap approach to learning. This supervised algorithm can then be

used to recognise behaviours from that point onwards.

The objective of this experiment is to demonstrate how we use the output of our unsupervised algorithm to train a supervised algorithm. In this experiment, we trained a supervised algorithm (i.e., the hidden Markov model (HMM)) based on the 'words' in the quantised dictionary (see Figure 4.10).

However, before using the unsupervised algorithm to train a supervised learning algorithm, we need to first label the 'words' in the quantised dictionary. We labelled them based on the knowledge gained from the sensor descriptions. Using the example shown in Figure 4.10, the tokens 'a' and 'q' refer to the sensor on the 'medicine cupboard' and tokens 'u' and 's' refer to the sensor on the 'bathroom tap', which correspond to the 'grooming/dressing' behaviour. If the first word (i.e., 'W1') in the quantised dictionary is labelled as the 'grooming/dressing' behaviour, then the HMM that represents this behaviour should train all the tokens that make up 'W1'.

It is also possible that there is more than one word that describes a behaviour. This can be seen from the example shown in Figure 4.10, where words 'W3' and 'W4' correspond to 'preparing meal/snack/beverages' behaviour. In such a situation, we only use the distinct set of tokens from 'W3' and 'W4' to train the 'preparing meal/snack/beverages' HMM.

We applied the method described in Chapter 3 to perform segmentation and behaviour recognition. The observations are the tokens from the sensors and the hidden states are the events that caused the observations. We train a set of HMMs, where each HMM represents one behaviour (see Figure 4.10), using the standard Expectation Maximization (EM) algorithm (described in Section 3.2.3). We present the sensor observations from the test set to the set of trained HMMs by sliding a window of size 10 across the sensor stream and segmentation is performed following the method described in Section 3.3. The results are shown in Table 4.5.

To evaluate the effectiveness of the unsupervised method and using the labels from

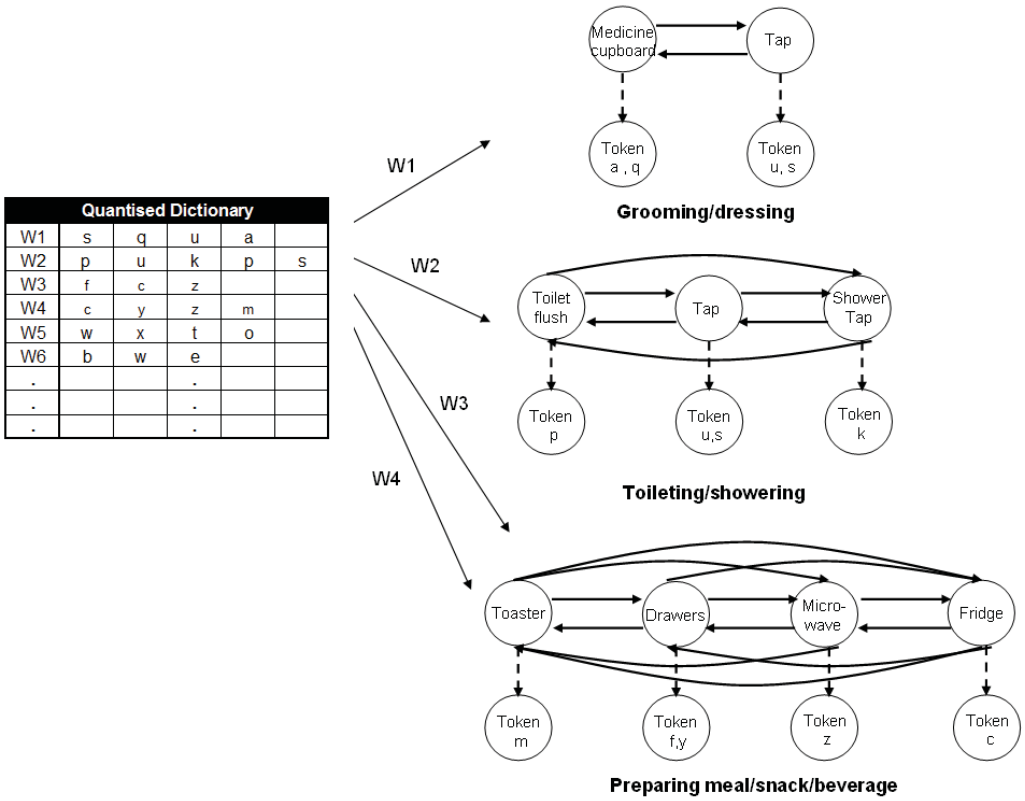


Figure 4.10: An illustration on how we train a set of hidden Markov models based on the 'words' in the quantised dictionary. There are probabilities associated to each edge, but they are omitted for clarity.

the unsupervised method to train a supervised learning method, we train a baseline classifier on the annotated data, where the sensors and activities are known *a priori*. This enables us to obtain a baseline recognition system. Since all HMMs are trained from the examples of the training datasets, there are no unidentified activities (i.e., the number of activities that are not identified by the HMM algorithm) reported. The results of unsupervised learning, using the unsupervised learning to train a supervised learning, and supervised learning are presented in Table 4.5.

The unsupervised method achieves an accuracy of 83%, which is not too dissimilar from the supervised method with an accuracy of 91%, considering that the proposed method works on unannotated data streams. This means that the unsupervised method presented in this thesis works effectively to identify behaviours from

Test Sets	Recognition Accuracy		
	Unsupervised Learning	Using labels from the unsupervised learning to train a supervised learning	Supervised Learning
1st Set	81%	84%	90%
2nd Set	76%	87%	91%
3rd Set	90%	95%	95%
4th Set	91%	94%	91%
5th Set	84%	90%	92%
6th Set	79%	88%	91%
7th Set	86%	92%	95%
8th Set	79%	83%	85%
Average	**83%**	**89%**	**91%**
Standard Deviation	**5.4**	**4.4**	**3.2**

Table 4.5: A comparison of results between unsupervised, using the unsupervised learning method to train a supervised learning method and supervised learning methods. The supervised method trains the HMMs directly from the ground truth.

the unlabelled data stream. The output of our method can also be used to train a supervised classifier, achieving an accuracy of 89%, which is comparable to the supervised method. We believe that adding additional contextual information (such as time, spatial or other context information) could improve the accuracy of the system. This could be useful to recognise behaviour that is not typical. We will investigate this further in future work.

4.5 Discussion

Our algorithms did well, producing over 80% accuracy. The encouraging results have shown the applicability of using compression to learn patterns of variable length in a data stream. We used the discovered patterns as the basis to identify the behaviours from the data stream and observed that the predicted 'words' correspond with the inhabitant's activities.

The main advantage of this approach is that it requires no prior knowledge or data labelling of the sensors and yet is able to identify patterns that correspond to the behaviours of the inhabitant. This means that generalisability is not an issue since the recognition system can learn from scratch without any built-in or prior knowledge when the sensors are installed in the home.

The output of our system will essentially report which 'word' is identified at each

time, which could only be meaningful by adding a human label to it to see what the inhabitant of the house is doing at each time. This means that at some point, the caregivers will need to interpret the output. One challenge with unsupervised learning is that there is no guarantee that the identified behaviour will match precisely the labels that would have been assigned by a human. For example, if a person always has a shower before having breakfast, the algorithm may well decide that there is only one behaviour here.

Since this approach is based on compression, it does mean that the activities need to be repeated frequently, otherwise, compression simply cannot be achieved. This will result in some behaviours (typically behaviours which are rare) not being identified (such as mowing the grass that happens every few weeks or washing the car bi-weekly). One potential solution is to use *transfer learning*, where the algorithm can leverage on the acquired knowledge from the available patterns (output from the compression method), and transfer the knowledge between different behaviours that share common tasks.

Although LZW allows discovery of patterns in an incremental fashion, the dictionary quantisation step could only be performed once it 'sees' the entire token sequence before any individual patterns can be identified. The question is then when should the quantisation be carried out. Based on our own experience with the MIT PlaceLab data, for a pattern to be recognised, it needs to be seen at least approximately 1.6 times of the word length. For the algorithm to be able to detect the word 'mouse' as an example, it should appear at least 9 times in the token stream. This means that for the algorithm to effectively recognise daily patterns, the LZW should be trained for at least 10 days before dictionary quantisation is carried out. This was shown in the experiment described in Section 4.4.2 which uses 14 days training data, and can effectively recognise most of the inhabitant's behaviours, producing over 80% accuracy.

Another question that is often raised in connection to this is to decide whether the LZW should stop exploiting the repetition in the sensor stream once the dictionary has been quantised. There are two possible ways to this, which depends on how well the 'words' from the quantised dictionary represent the behaviours of the inhabitant. If it doesn't, then LZW should start to learn from scratch and re-construct a new model, with the hope that it will be offset by a better model later. Otherwise, we can use the 'words' in the dictionary as a base and the algorithm can learn from the existing model.

4.6 Summary

In this chapter, we presented a new approach to automatically cluster the unlabelled data stream. The approach that we took is based on compression and exploits the redundancy in an unlabelled data stream, which we define to be behaviours. In order to allow variations in the behaviours, we extended the Lempel-Ziv-Welch (LZW) method to perform lossy compression using edit distance, which is also used to segment the unlabelled token stream into behaviours that we have identified.

We evaluated the effectiveness of our proposed method on two datasets: one fixed length dataset and another of variable-length data sequences. We compared the labels produced by our approach with the ground truth labels. The results are promising since the method does not need any prior human labelling, and show significant improvement in the recognition accuracy over the Self-Organising Map. We also show how the output of the proposed unsupervised method described in this chapter can be used to train a supervised classifier. We compared the unsupervised method with a baseline supervised method, based on the hidden Markov model, to see how effective the system can be made.

The results of using the unsupervised method are encouraging and we believe that using this approach, a smart home can be built that starts from tabula rasa,

meaning that the recognition system can learn from scratch without any built-in or prior knowledge when the sensors are installed in the home.

Chapter 5

Finding Informative Sensors

This chapter addresses the sensor selection problem for behaviour recognition. A description of the problem is first introduced and then a solution is given, together with experiments showing the effectiveness of the proposed method for identifying informative sensors. The results are validated using methods described in Chapters 3 and 4 on a real smart home dataset.

5.1 Problem Description

The work in Chapters 3 and 4 mainly focussed on finding the mapping between sensor outputs from the home and the behaviours of the inhabitant by using two approaches: one using supervised learning on an annotated sensory stream (Chapter 3) and one using unsupervised learning on unannotated ones (Chapter 4).

The literature reviews from the previous two chapters have shown that there has been a lot of work focussing on recognising the behaviours of the inhabitant from unobtrusive sensors. Most of the work, including ours as described before, assumes that the sensors are already in place in the house. However, this is not a necessary assumption, in particular in cases where the smart home is at the design stage. Then the question of which sensors are useful to effectively recognise the inhabitant's behaviours arises.

Our interest in this problem begins with this question of whether to train a learning

algorithm on: (1) as many sensors as possible or (2) a subset of the sensors by eliminating those sensors that do not contribute to the classification accuracy. Even when the budget is not a constraint to purchase the number of sensors needed in a smart home, it has been shown in many studies that training a learning algorithm on features that are irrelevant and/or redundant not only affects the overall classification performance and the generalisability of the learned model, but also requires a greater amount of training data [6, 67, 73]. For example, the number of training data required to train a naïve Bayes classifier, which assumes independence of features (described in Section 3.2.2), grows exponentially with the number of irrelevant features [73], while the decision tree tends to overfit and the performance of nearest neighbour degrades when trained on irrelevant and redundant features [72].

In fact, many inductive machine learning algorithms (such as neural networks, decision trees, genetic algorithms, etc.) employ the principle of Occam's Razor, which prefers the simpler model that fits the data as the basis for model construction. A learning algorithm that follows this principle trains on a smaller number of features over a larger number in order to generalise beyond the training data and achieve a better recognition accuracy.

This thesis views the sensor selection problem as a task of selecting a set of sensors that are relevant by removing as many irrelevant and/or redundant sensors as possible for behaviour recognition. An irrelevant sensor is a sensor that does not contribute to distinguishing the different classes of behaviours [58]. Consider the example of a sensor attached to the kitchen tap. The kitchen tap might be included in multiple behaviours such as somebody could be in the kitchen washing dishes, making tea (filling up the kettle to boil water) or preparing a meal (filling up the pot to boil soup). If other sensors are able to discriminate between these behaviours then the kitchen tap sensor can be removed without affecting the recognition performance. A redundant sensor, in general, refers to a sensor that is co-present with another sensor,

both being relevant for behaviour recognition, but the removal of one of the sensors will not affect the classification performance [152]. For example, a sensor attached to the laundry door and a sensor attached to the washing machine. Removing either one will not affect the classification of 'doing laundry' behaviour.

Our aim in this chapter is to identify 'informative' sensors, i.e., sensors that provide us with the most information about the inhabitant's behaviours. The idea here is that if we could find one particular sensor that could tell us exactly what the behaviour is then this could not only reduce the number of arbitrary sensors needed, but also help to guide future sensor installation in a new home, where the designer of future smart homes can use the posteriori knowledge to install the sensors that provide the most information about the inhabitants behaviours. This motivates us to look into this problem, which has not received wide attention yet, but which we consider to be important.

Given a set of sensors currently installed in the house, the question then is 'which sensors are useful to recognise the inhabitant's behaviours?' For that, we need to determine the amount of information we can get from each sensor. Our solution to the sensor selection problem is based on an information-theoretic approach, which uses information gain as a measure to identify the informative sensors.

Information gain is commonly used in decision tree induction to select which feature to test at each node while building the tree [95]. As described in Section 3.2.1, a decision tree is a supervised learning method that consists of a set of *nodes*, where the features are tested and the *leaves* represent the classification decision. Most decision tree algorithms employ a greedy approach i.e., beginning at the root, they recursively partitions the training set into smaller subsets by choosing the most informative feature at each step.

Quinlan's ID3 decision tree algorithm [112] is one example that uses information gain as a heuristic for selecting which feature will best partition the training data into

separate classes. It computes the information gain for each feature at each step while growing the tree. The feature with the highest information gain, i.e., most useful for classifying the training examples according to their class labels, is chosen as the test feature for a given training set. Since the effectiveness of a sensor to classify the inhabitant's behaviours can be measured using information gain, we can therefore train a decision tree to determine which sensors are informative, where the sensors that are placed close to the root have higher information gain and hence are more informative.

In many machine learning algorithms, resampling methods (such as cross validation) are often used to evaluate the performance of the learned model or compare the performance of different algorithms [117]. In the cross validation method, the same data is repeatedly split into different training-testing partitions in such a way that each subset of the data is tested on. When repeated rounds of tree building are trained on different partitions of training data, this will produce different decision trees, even when the same information gain method is used. This is also particularly important when the decision tree is trained on different smart home datasets (e.g., one using information gain and another using gain-ratio), which will result in more than one tree. This brings us to the question of how to identify the informative sensors among a set of trees?

There are a number of challenges that make this task difficult. To see this, we use the example shown in Figure 5.1:

(a) a sensor could be very informative in one tree and less informative in another tree, meaning that the sensor could be placed at different levels on different trees (e.g., sensor S_3 in Figure 5.1)

(b) a sensor may not necessarily appear in all the set of trees (e.g., sensor S_6 in Figure 5.1 does not appear in τ_3)

(c) the size of the trees i.e., the number of nodes and leaves, might not necessarily

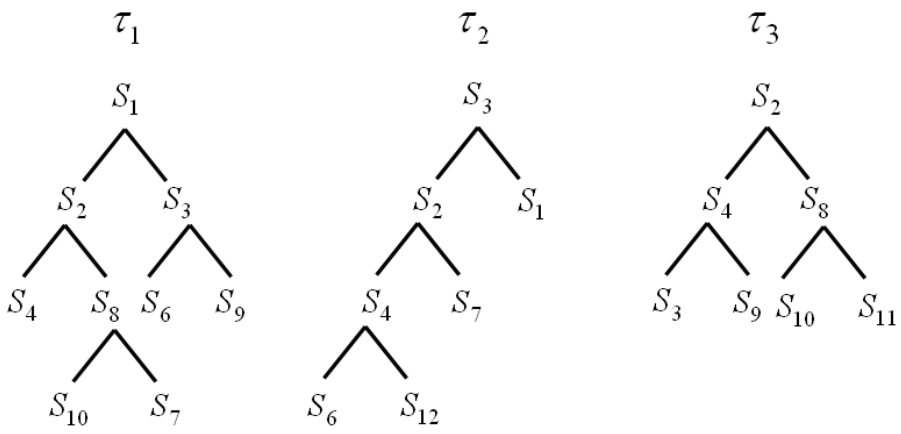


Figure 5.1: An example of 3 different decision trees (τ_1, τ_2 and τ_3) trained on 3 different partitions of training data.

be the same.

For the last decade, many issues concerning tree building such as finding good splits, preprocessing of the data before inducing trees, stopping criteria, constructing optimal trees, generalisation accuracy and addressing missing attribute values have been extensively reported in the literature [9, 31, 112]. However, very few methods gain insights about the significance of the nodes in a set of trees.

This chapter addresses the sensor selection problem for behaviour recognition in a smart home. We first review methods that are used to address the sensor selection problem for behaviour recognition. We then describe the information gain measure to identify informative sensors from a set of sensors currently installed in the home and introduce our methods to gain insights about the significance of the informative sensors identified across the different decision trees. We use a smart home dataset to demonstrate the efficacy of our methods to recognise the inhabitant's behaviours.

5.2 Relevant Literature

In this section, we review methods that address the sensor selection problem for the behaviour recognition task. In fact, the sensor selection problem is closely related to

the feature selection problem in machine learning and statistics, which aims to find useful features in a dataset prior to learning.

There are two well-known approaches: (1) a wrapper-based approach [64], which uses a learning algorithm to score the features according to their predictive performance and (2) a filter-based approach [58], which evaluates characteristics of the features based on some heuristics without involving any learning algorithm.

In the wrapper-based approach, there are two common search strategies to explore the combination of features. The first strategy is called the forward selection, where it starts with no features and successively adds relevant features. This was considered in the work of Maurer et al. [90], where they first disabled all the sensors in the beginning and enabled one sensor at a time. The recognition performance from a learning algorithm is used as a criterion to determine the importance of the sensors.

The second strategy is called backward selection, and starts by training all the features and removing irrelevant ones. In the work of Ravi et al. [116], they trained all the sensors in the beginning and removed one sensor at a time. The central idea of using forward and backward selection strategies is that when a sensor is added or removed, it will affect the recognition performance, which gives an indication on whether the sensor is important or not. Both the work of Maurer et al. [90] and Ravi et al. [116] addressed the sensor selection problem by identifying the performance of the accelerometers worn on different body positions in order to detect human physical activities.

A variation to this is the work of Bao and Intille [4] where a supervised algorithm was trained and tested on the data from only one sensor each time. The sensor that achieves a high recognition accuracy and recognises a majority of the human physical activity is considered important. However, this method does not take into account sensor dependency.

One of the drawbacks with the strategies described above is that they are imprac-

tical when the number of sensors that need to be evaluated is large and thus training a classifier for each feature subset is a time-consuming process with high computational cost.

The filter-based approach, on the other hand, evaluates the characteristics of the data based on some measures, independent of any learning algorithm. One common filter-based method is to use information-theoretic measure such as the information gain to measure the importance of the sensors.

Information gain measures the reduction of uncertainty about the class labelled (i.e., the behaviour of the inhabitant). A higher information gain means that the sensor has a higher chance of getting a pure class and is therefore more informative. In both the works of Chen et al. [15] and Lösch et al. [82], information gain is used to identify the importance of each sensor for behaviour recognition. Chen et al. [15] used information gain to identify informative sensors from objects-used and other contextual information such as temperature and humidity. For each pair of sensor and behaviour, they calculate the information gain and choose the sensors with a high information gain, where these sensors are more highly correlated with the corresponding behaviour. The identified sensors are then used to train a dynamic Bayesian network for behaviour recognition. The work of Lösch et al. [82] also used information gain the same way as in [15], but their work focused on informative features obtained from the camera, which is applied to the field of human-robot interaction.

Although these works use information gain to measure the importance of the sensors by ranking them and selecting sensors with highest information gain, they are not effective as they do not reduce the number of sensors and often rely on prior knowledge in order to define a suitable cut-off point to remove sensors with low information gain.

Our approach to the sensor selection problem draws from the filter-based approach, i.e., using the information gain measure for sensor selection by modelling it

directly in the form of a decision tree. However, as described in the previous section, when a decision tree is trained on each partition of training data from a smart home dataset or data from different smart homes, it will result in more than one decision tree. For that, we need a method that can 'generalise' the identified informative sensors across a set of trees.

The work of Ratanamahatana and Gunopulos [115] identified features from each tree by empirically taking only the features that appear on the top 3 levels of the decision tree. The features on each tree are then unified together and trained on a supervised learning algorithm for classification. One problem with this method is that it may identify some unimportant features as important ones. Consider the following scenarios: (a) a sensor only appears at level 3 in one tree and not in other trees, and (b) another sensor appears at level 4 in all the trees. Following this approach, the sensor in scenario (a) is more likely to be selected as an important sensor since it appears on the top 3 levels in one of the trees, while the sensor in scenario (b) is not selected although it appears in all the trees. This may result in some equally good sensors being missed.

In the next section, we describe the information gain measures for sensor selection and then describe our methods to gain insights about the significance of the sensors identified among a set of trees.

5.3 Information Gain Measures for Sensors Selection

A decision tree is a supervised machine learning method that consists of a set of nodes where features are tested and edges that descend from the nodes represent the feature values. A node without any descending edges is called a leaf, which corresponds to the classification decision.

The decision tree is analogous to the 20 questions game in order to identify a piece

of information.[1] In the context of the smart home, the piece of information that we are interested in is 'what the inhabitant is doing'. Suppose that a system is designed that could only respond to questions that can be answered with either 'yes' or 'no', what question should we ask first? Obviously, we would ask a question that gives us the most information and reduces ambiguity about the inhabitant's behaviour. For example, suppose that you begin asking the system if the inhabitant is in the kitchen. A 'yes' answer to this question certainly eliminates the possibility that the inhabitant is in the bedroom, washroom, or living room. You might then proceed to ask if the inhabitant is cooking, which eliminates making tea/coffee, eating and washing dishes behaviours from the range of possible answers. Obviously, not all behaviours are equally likely to occur. For example, if the inhabitant is making tea in 80% of all the cases and perfoming any other behaviour in the remaining 20%, then it would make sense to ask if the inhabitant is making tea first.

In this 20 questions game, no prior information is provided at the beginning, but after obtaining answers to several questions, we can have a more precise idea of the particular behaviour of the inhabitant. Thus, the ability to ask the correct question is important. For example, in a vague question such as 'is the inhabitant squatting?', a 'no' reply leaves many possibilities and could lead to additional questions. The idea is to minimise the number of questions needed to be asked in order to identify the probable behaviour of the inhabitant.

One common criterion to measure a question's usefulness is the *information gain* measure, which is the expected reduction in uncertainty. To illustrate the information gain, we use the following example:

[1]The 20 question game is a spoken parlour game that enables deductive reasoning [145]. In this game, a person selects a subject, which is not to be revealed to all other players and acts as an answerer. The other players are the questioners and each takes a turn to ask a question. If a player guesses the correct answer then that player wins.

Burner	Kitchen Light	Behaviour
Off	On	Washing dishes
Off	On	Washing dishes
On	Off	Cooking
On	On	Cooking
On	Off	Cooking

The first two columns are the sensors attached to the burner and the kitchen light, which can take on the value on or off, while the last column represents the classes of behaviour, which is either washing dishes or cooking. From the above example, we could discover some relationships which can help to predict the behaviour of the inhabitant. Looking at the example, it clearly shows that the sensor attached to the burner is highly correlated to the cooking behaviour where we could use a simple rule such as 'if burner = on, then behaviour = cooking' to represent the relationship between the burner and the cooking behaviour. This means that the chances of getting the right answer is maximal by focussing on the sensor attached to the burner. The information gained from the sensor on the burner is high since it effectively classifies the cooking behaviour as compared to the kitchen light. Based on this simple illustration, the information gain is the amount of information provided about the behaviour (e.g., cooking), by knowing the value of the sensor (e.g., burner).

The decision tree uses information gain to identify which feature should be tested at each node when growing the tree. The aim is to minimise the expected number of tests (questions) to classify a given set of training examples so that a smaller tree is found. In a decision tree, the goodness of split is characterised by an impurity measure. A split is considered pure if after splitting, all instances belong to the same class, otherwise if a node is not pure, then the instances should be split to decrease the impurity. The Shannon entropy [122] is one common measure in information theory to measure impurity. The entropy is defined as the minimum number of bits required to encode the classification of an arbitrary number of examples in the data [95]. Thus,

the entropy for a set of training examples S, with a target class that can take on m different values, is defined as:

$$Entropy(S) = -\sum_{i=1}^{m} p_i \log_2 p_i \tag{5.1}$$

where p_i is the proportion of S belonging to class i. By defining entropy as a measure of impurity, information gain is thus the expected reduction in entropy resulting from partitioning the training examples according to its feature. So, the information gain $Gain(S, F)$ of a feature F, that corresponds to the set of training examples S, is defined as in [95]:

$$Gain(S, F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{5.2}$$

where the $Values(F)$ are the possible values of feature F. S_v is the subset of S for which feature F has value v, i.e., $S_v = \{s \in S | F(s) = v\}$. The $Entropy(S)$ in Equation 5.2 refers to the entropy of the entire set of the training examples S, while $\sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v)$ is the expected value of the entropy after S is partitioned using feature F. Thus, $Gain(S, F)$ is the expected reduction in entropy by knowing the value of feature F. The value of $Gain(S, F)$ is the average number of questions that we need to ask in order to determine the inhabitant's behaviour.

The ID3 decision tree algorithm [112] is one example that uses information gain to measure the effectiveness of a feature to classify the training examples at each step when building the tree i.e., selecting those features that maximise the information gain (Equation 5.2). Features with high information gain are placed closer to the root, and hence are more informative. It is notable that when information gain is applied to features whose class labels have low frequency in the dataset, then the

features that describe these classes will have a poor information gain.

Just like any supervised learning algorithm, the ID3 decision tree is presented with a set of exemplars of each behaviour that is to be learned, with each exemplar labelled. In the smart home context, the only information that describes the inhabitant's behaviours is the sensor observations, which takes on the values of the sensor state (i.e., opened/closed or on/off). This sensor information represent the set of features that are to be learned. By training the decision tree, we can then determine the informative sensors, since the sensor that effectively classifies the inhabitant's behaviour has a higher information gain and thus is more informative.

However, one problem raised in connection to this is that training a decision tree on different partitioning of training data will result in more than a single decision tree. Thus, we need a method that can gain insights about the significance of the nodes in a set of trees, which is discussed next.

5.4 Description of Our Approaches

The majority of machine learning algorithms use cross validation either to estimate the performance of the learned model or to compare the performance of two or more different algorithms [117]. When repeated rounds of cross validation are performed on different partitions of training data, this will produce different decision trees. In this section, we introduce two approaches to gain insights about the significance of the informative sensors identified among a set of trees.

The general idea of the first approach is to give each sensor a weight and rank them according to their score. The second approach is to find the shortest distance tree, which aims to identify a tree where the distance between this tree and all the other trees is minimal. We introduced two methods to find the shortest distance tree: (1) edge matching and (2) subtree pruning and regraft (SPR). Both methods use a distance metric to measure the similarity between pairs of trees, but they are

conceptually rather different from each other. The former calculates the distance based on the number of unmatched directed edges while the latter is determined by the number of cuts made to each tree.

Since we are only interested in obtaining insights about the significance of the informative sensors, which are the nodes in the decision tree, we exclude the leaves from the trees in the discussion of our methods in the subsequent sections.

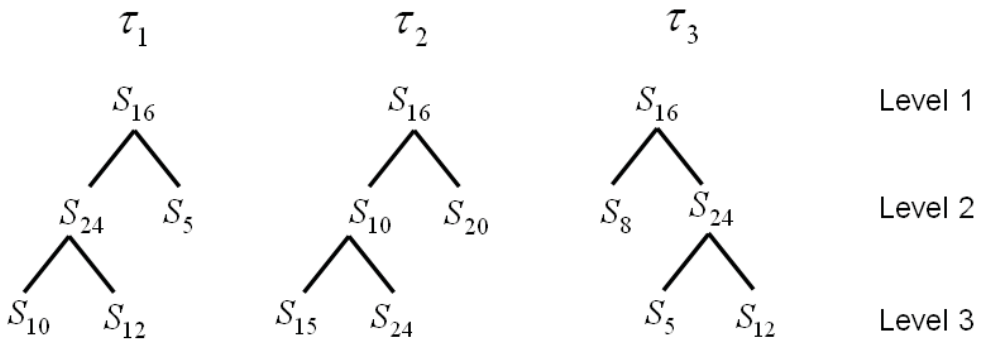
5.4.1 Ranking sensors by weights

The first approach is to assign each sensor a weight and then rank them according to their score. There are many ways to assign the weight to the sensor, depending on the penalty we apply to the sensors in the lower levels of the tree and the sensors that do not appear in all the set of the trees. Since we are taking the average of the weights across a set of trees, we use a naïve approach i.e., assigning the weight w_i by taking the inverse of the level of the sensor on a tree, where $0 \leq w_i \leq 1$. The idea here is that if a sensor does not appear in a tree, then we penalise the sensor by assigning a weight of 0, while the weight for the sensor at the root will always be 1. Thus, the more informative sensors, which are placed close to the root, will have higher weights.

We use the average weight as the ranking criterion, which is calculated by averaging the sensor weights on each tree across all the given trees. The sensors are then ranked according to their average weight. The output of this method is a list of sensors ranked according to their scores.

Figure 5.2 shows an illustration on how sensors' weights are calculated and ranked according to their respective scores. To illustrate, we use S_5 as a running example. As the figure shows, S_5 is at level 2 on τ_1 and level 3 on τ_3, and since it does not appear on τ_2, a weight 0 is assigned for τ_2. Thus, inversing the level of sensor S_5 in each tree will give a weight of $\frac{1}{2}$ for τ_1, 0 for τ_2 and $\frac{1}{3}$ for τ_3. The total weight for

S_5 is therefore the sum of all these weights added together, giving a total of 0.833. Since in this example, there are a total of 3 trees, the average weight is calculated by dividing the total weights by 3. All the sensors are then ranked according to their average weights. In the example shown in Figure 5.2, S_{16} is ranked 1^{st}, since this sensor is most informative.



Sensors	Weights	Total Weights	Average	Rank
S_5	$\frac{1}{2}$, 0, $\frac{1}{3}$	0.833	0.278	3=
S_8	0, 0, $\frac{1}{2}$	0.5	0.166	6=
S_{10}	$\frac{1}{3}$, $\frac{1}{2}$, 0	0.833	0.278	3=
S_{12}	$\frac{1}{3}$, 0, $\frac{1}{3}$	0.666	0.222	5
S_{15}	0, $\frac{1}{3}$, 0	0.333	0.111	8
S_{16}	1, 1, 1	3	1	1
S_{20}	0, $\frac{1}{2}$, 0	0.5	0.166	6=
S_{24}	$\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{2}$	1.333	0.444	2

Figure 5.2: Illustration of how sensors are ranked based on weights. The weights are calculated by inverting the level of the sensor in each tree and the total weights are the summation of all the weights. The average is calculated by dividing the total weights by the number of trees, which in this example is 3. The sensors are ranked in descending order according to their average weights.

This method can be summarised in a three-step procedure:

(1) **Calculate the sensor weights.** For each sensor S_i ; $i = 1, 2, \ldots, n$, calculate

the weights (w_{s_i}) by taking the inverse of the level of the sensor in each tree ($\tau_1, \tau_2, \ldots, \tau_k$). This means that the higher the score, the more informative the sensor is. The total weights for each sensor is therefore equal to $\sum_{\tau=1}^{k} w_{s_i}$.

(2) **Calculate the average weight.** The average weight for each sensor across all the trees is calculated by dividing the total weights by the number of trees k.

(3) **Rank the sensors.** The sensors are then ranked (in descending order) according to the average weights. The sensor with the highest average weight is the most informative sensor.

This approach has the advantage of knowing which are the top n informative sensors that are needed most. This is helpful in situations where budget allocation for sensor installation is limited and based on the ranking of these sensors, we can identify which top n sensors to install. Using the example shown in Figure 5.2, we can identify the top 5 most informative sensors as $\{S_{16}, S_{24}, S_5, S_{10}, S_{12}\}$.

Although this method measures the importance of the sensors by ranking them according to their respective scores and selecting sensors with the highest information gain, one problem with this method (as described in Section 5.2) is the need to pre-specify a suitable cut-off point to remove sensors with low information gain. Furthermore, as described in Section 5.3, if a behaviour is rare (e.g., mowing the grass, which only occurs once a month or every few weeks), then sensors (e.g., sensors on the lawn mower, rake, glove, etc.) will have a poor information gain and possibly be ranked very low (e.g., S_{15} in Figure 5.2). Such good sensors for distinguishing rare behaviours may probably not be selected if they do not fall within the cut-off point.

5.4.2 Shortest distance tree

In this section, we describe two methods to compute the shortest distance tree. One method is based on edge matching, which calculates the distance based on the num-

ber of unmatched edges, while the other method is based on subtree pruning and regrafting (SPR), which calculates the distance based on the number of cuts made to each tree when pruning the tree.

Given a collection of trees $\tau_1, \tau_2, \ldots, \tau_k$, the shortest distance tree approach aims to find a tree where the distance between this tree and all other trees is minimal. Let $dist_{ij}$ be the distance from τ_i to τ_j, the total distance from τ_i to all other trees (d_i) is thus

$$d_i = \sum_{j=1}^{k} dist_{ij}. \tag{5.3}$$

The average distance of τ_i is

$$\frac{d_i}{k-1}. \tag{5.4}$$

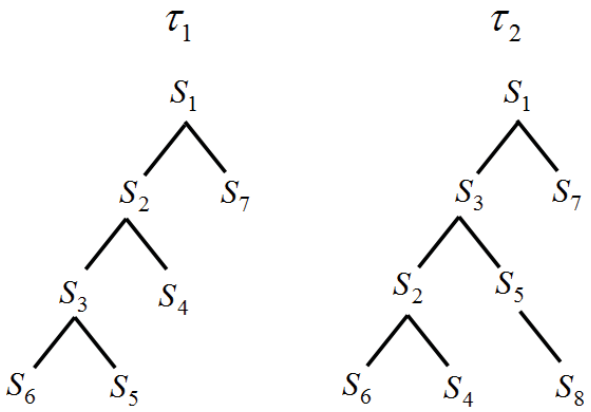
The shortest distance tree is hence the tree with the minimum average distance.

(a) The edge matching method

In this section, we introduce an edge matching method to calculate the shortest distance tree. In this method, we define the distance to transform τ_i to τ_j ($dist_{ij}$) as the number of unmatched directed edges between τ_i and τ_j.

Let $\tau_i = (V_i, E_i)$ where V_i is the set of vertices and E_i is the set of edges for tree τ_i, and similarly $\tau_j = (V_j, E_j)$. For each $(A, B) \in E_i$, where A and B are nodes in τ_i, and A is adjacent to B, the algorithm traverses tree τ_j to find a match for this pair of adjacent nodes. If a match is not found, then the algorithm increments the distance value by 1. An array is used to keep track of the unmatched directed edges of nodes A and B.

To illustrate the edge matching method, we use the example shown in Figure 5.3.



Trees	Unmatched Directed Edge	Distance
$dist_{12}$	$(S_1, S_2), (S_2, S_3), (S_3, S_6)$	3
$dist_{21}$	$(S_1, S_3), (S_3, S_2), (S_2, S_6), (S_5, S_8)$	4

Figure 5.3: Illustration on the edge matching method. The distance is calculated based on the number of unmatched directed edges to transform one tree into the other, i.e., τ_1 to τ_2 ($dist_{12}$) and from τ_2 to τ_1 ($dist_{21}$).

To calculate the distance for $dist_{12}$, the algorithm begins at the root of τ_1, taking the first edge, i.e, (S_1, S_2) and traverses through τ_2 to find a match. Since there is no match for the edge (S_1, S_2) in τ_2, the distance is increased by 1 and this edge is written into an array. The algorithm repeats the same process by taking the next edge from τ_1, i.e., (S_1, S_7) and traverses through τ_2 to find a match. The distance value remains unchanged since a match is found. This process is repeated for all the edges in τ_1. As can be seen in the table shown in Figure 5.3, there are 3 unmatched directed edges and thus the total distance for $dist_{12}$ is 3.

The algorithm repeats the same process for $dist_{21}$. This time, the algorithm begins at the root of τ_2, taking the first edge i.e., (S_1, S_3) and traverses through τ_1 to find a match. This process is repeated for all the edges in τ_2. Thus, the total distance for $dist_{21}$ based on the example shown in Figure 5.3 is 4.

As illustrated in Figure 5.3, the transformation from one tree into the other is

based on the number of unmatched edges on the tree that is to be transformed, hence the distances for $dist_{12}$ and $dist_{21}$ are not necessarily the same. Thus, for every pair of trees, the edge matching method will be carried out twice (i.e., once for $dist_{ij}$ and again for $dist_{ji}$). For each tree τ_i ; $i = 1, 2, \ldots, k$, the total distance for τ_i to other trees (d_i) and the average distance are calculated by following Equations 5.3 and 5.4. The shortest distance tree is selected based on the tree with the minimum average distance.

(b) The Subtree Pruning and Regraft (SPR) method

The second method to find the shortest distance tree is the subtree pruning and regraft (SPR). This method is motivated by phylogenetics, which aims to reconstruct the evolutionary history of species. The evolutionary histories of biological sequences is usually represented by a phylogenetic tree (i.e., a leaf-labelled tree), which is a directed graph with only one vertex that has in-degree zero and vertices with out-degree zero represent living species [7].

However, due to the occurrence of reticulate evolutionary events such as hybridisation, horizontal gene transfer and recombination, this will result in different phylogenetic trees for the same species. Thus, reconstructing the correct evolutionary tree for a set of species is one of the problems in evolutionary biology. Many tree transformation operations [2, 11, 41, 148] have been proposed in the literature to compare the different evolution trees for the same set of species. One common operation is the Subtree Pruning and Regraft (SPR) [7, 45]. An SPR operation to transform τ_1 to τ_2, is by cutting a subtree from τ_1 and then reattaching it to a different branch within τ_1 to match τ_2. Thus, the SPR distance is the minimum number of SPR operations needed to transform one phylogenetic tree into another. In the example shown in Figure 5.4, τ_1 can be transformed to τ_2 by cutting the subtree containing leaves 2 and 3, and reattaching it to the edge of leaf 4.

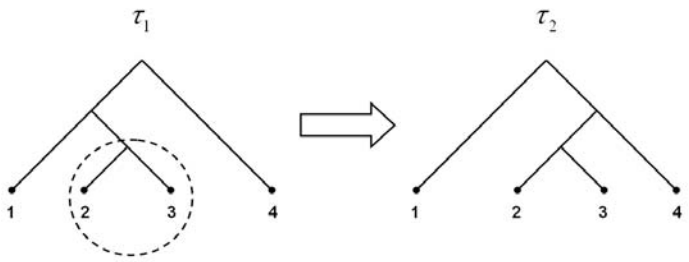


Figure 5.4: An illustration showing how Subtree Pruning and Regraft (SPR) operation on tree. τ_1 is transformed to τ_2 by cutting the subtree that contains the leaves 2 and 3 (shown in dashed line) and regrafting it to the edge of leaf 4.

Our SPR-based method borrows the idea of SPR operation in the application of phylogenetics. There are two main differences between the SPR operation in phylogenetics and our application to smart homes. First, the similarity between two trees in phylogenetic is by comparing the leaf sets, while in a smart home, we are interested to find the similarity of the sensors, which are the nodes in the tree. Thus the similarity between two decision trees is by comparing the nodes in the tree. Second, the majority of the SPR operations in phylogenetic assume that trees have the same set of taxa (i.e., leaves), while in the smart home, the number of sensors (i.e., nodes) in the trees could well not be the same (e.g., τ_1 and τ_2 in Figure 5.1) and a sensor may not necessarily appear in all the trees (e.g., sensor S_8 in Figure 5.1 does not appear in τ_2).

As with the edge matching method, the SPR method also uses a distance measure to determine how similar the two trees are. For our SPR-based method, we define the distance as the total number of cuts made on both trees. In general, given two trees τ_i and τ_j, the algorithm traverses both trees and makes a cut to the edge of the node that does not match. The algorithm keeps track on the number of cuts made to an edge, i.e., whenever a cut is made, the distance is increased by 1. The input to the SPR method is a pair of trees, τ_i and τ_j and the output is a distance value between the two trees.

The algorithm has two main steps. The first step is to traverse both trees τ_i and

τ_j, beginning at the root and check if the two nodes match. If the root in both trees matches, the algorithm continues the path to its child node and checks if the child node matches. If it does not, then two cuts are made, one on τ_i and another on τ_j and the unmatched subtrees are written into an array. Since there are two cuts made, the distance is increased by 2.

The second step of the algorithm is to traverse the subtrees whose edges have been cut from the main tree, which are stored in the array. For each subtree x in the array, the algorithm searches through all the remaining subtrees y. When all the nodes in subtree x match part of subtree y, a cut is made on subtree y, which increases the distance by 1 (see Figure 5.5(b)). However, when part of subtree x matches part of subtree y, two cuts are made, one on subtree x and another on y (see Figure 5.6(a)). Since two cuts are made, the distance is increased by 2 and the matched subtree x is then added to the array. If no match is found for the remaining subtrees y, then a cut is made on all the edges of subtree x where the distance is increased by the number of edges in subtree x (see Figure 5.6(b)) and the algorithm iterates for the next subtree.

To illustrate the SPR method, we use the example shown in Figure 5.5. Given two trees τ_1 and τ_2 in Figure 5.5(a), the algorithm begins at the root and checks if the node in both trees matches. As the figure shows, the first node in both trees matches (i.e., S_1) and this process is repeated by traversing down to its child node. Since S_8 in τ_1 does not match with S_7 in τ_2, then a cut is made at the edge of S_8 in τ_1 and S_7 in τ_2 (shown as dashed line in Figure 5.5(b)), and so the distance is 2. Continuing in this manner, there are two subsequent cuts made to the edge of S_9 in τ_1 and S_{12} in τ_2, which updates the distance to 4.

Once the algorithm traverses all the remaining nodes in τ_1 and τ_2, the next step is to traverse the subtrees, whose edges have been cut from the τ_1 and τ_2. Taking the first subtree, i.e., S_8, S_{10}, S_{11}, the algorithm searches for a match in other subtrees. A

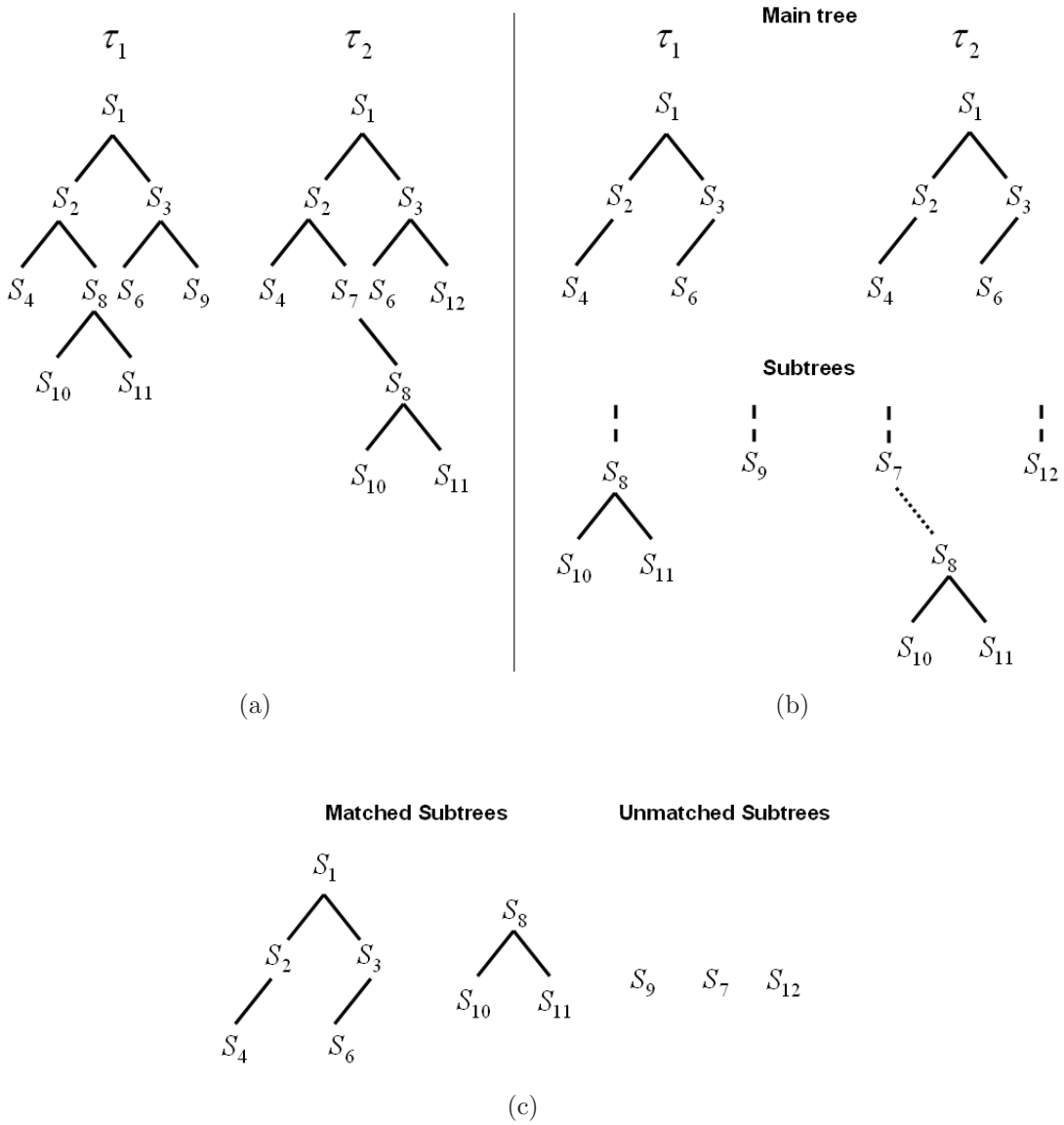


Figure 5.5: Illustration on the Subtree Pruning and Regraft (SPR) method on (a) two trees, τ_1 and τ_2. (b) [i] *Iteration on main tree*: the algorithm traverses τ_1 and τ_2 and finds a node that matches. A cut is made on the edge of the node that does not match (shown in dashed line). At this point, 4 cuts are made on both trees and thus, the distance is 4. [ii] *Iteration on subtree*: The algorithm then traverses the subtrees of S_8, S_{10}, S_{11} and find a match in the remaining subtrees. Since there is a match for this subtree, a cut is made (shown in dotted line) from the branch of S_7. This brings the total number of cuts made to 5. This process continues for the remaining subtrees S_9,S_7 and S_{12}. The total distance in this example is 5. (c) The results of SPR operations on τ_1 and τ_2 are two matching subtrees and three unmatched subtrees. For details of the method, see text.

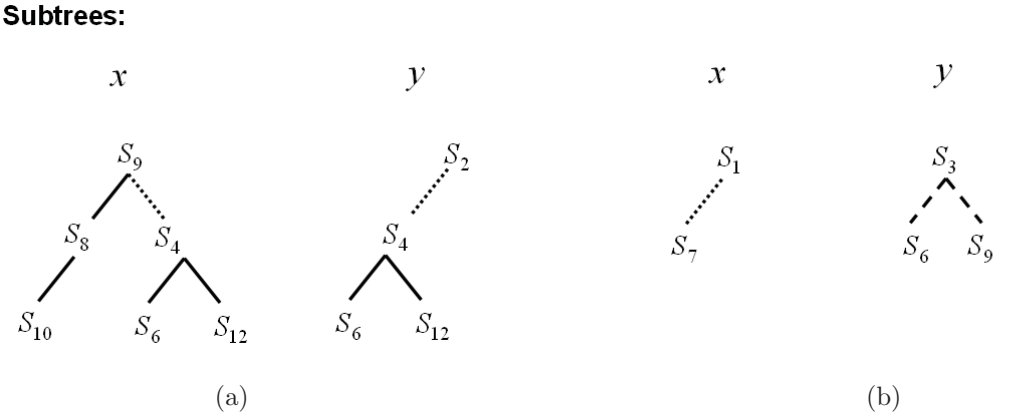


Figure 5.6: Illustration on the iteration on subtrees (i.e., x and y) of the SPR algorithm: (a) part of subtree x matches part of subtree y, i.e., subtree of S_4, S_6, S_{12}. Since two cuts are made (shown in dotted line) on these subtrees x and y, the distance in this example is 2. (b) If none of the remaining subtrees y match subtree x, a cut is made (shown in dotted line) from the branch of S_1 of subtree x, where the distance is 1. Since subtree y in this example is the only remaining subtree, two cuts are made from the branch of S_3 of subtree y (shown in dashed line), which increases the total distance in this example to 3.

match is found from the subtree of S_7 (shown as a dotted line in Figure 5.5(b)), thus a cut is made, which increases the distance by 1, giving a total distance of 5. The process is then repeated for the remaining subtrees S_9, S_7 and S_{12}. The results of SPR operation on τ_1 and τ_2 are two matching subtrees and three unmatched subtrees shown in Figure 5.5(c) with a total distance of 5.

Unlike the edge matching method, which transforms one tree into the other, the SPR method prunes the trees. Thus, the SPR method on a pair of trees only need to be carried out once. The total and average distance for each tree τ_i are calculated by following Equations 5.3 and 5.4. A tree with the minimum distance across all other trees is selected as the shortest distance tree. The benefit of using the shortest distance tree approach is the fact that the shortest distance tree can be reused for classification without the need for additional training once the decision tree that recognises the behaviours have been created. Furthermore, this approach eliminates the need to empirically define a cut-off point to remove sensors with low information gain (as can be seen with the weight ranking method (Section 5.4.1)).

SPR Algorithm

The SPR algorithm to compute the distance between two trees is as follows:

(1) *Initialisation*

- two trees, τ_i and τ_j
- distance = 0

(2) *Iteration on main tree*

- For each node in τ_i and τ_j (beginning at the root), check if node matches.
  - If node matches and has child nodes
    - ▷ continue the path to its child node by repeating step (2).
  - else
    - ▷ distance = distance + 2
    - ▷ write unmatched subtrees to array

(3) *Iteration on subtrees*

- For each subtree x in array
  - For each remaining subtree y in array
    - if part of subtree x matches part of subtree y (see Figure 5.6(a))
      - ▷ distance = distance + 2
      - ▷ write matched subtree x to array
    - if all the nodes in subtree x match part of subtree y or vice versa (see subtrees S_8, S_{10}, S_{11} in Figure 5.5(b))
      - ▷ distance = distance + 1
  - If no match for each remaining subtree y (see Figure 5.6(b))
    - ▷ distance = distance + number of edges in subtree x

5.5 Experimental Results

To test the efficacy of our methods, we conducted an experiment that consists of 3 steps. In the first step, we train a decision tree based on each partition of training data, which results in a set of decision trees. Using the set of induced decision trees, the second step investigates the weight ranking, edge matching, and subtree pruning and regraft methods (described in Section 5.4) to gain insights about the significance of the informative sensors across the different trained decision trees. The third step validates the results by training two supervised machine learning methods (i.e., the hidden Markov model (described in Chapter 3)) and decision tree on the informative sensors identified from the second step. We also compare the accuracy with the unsupervised compression-based method (described in Chapter 4).

We demonstrated our method using the MIT PlaceLab dataset, which is described in Section 1.5. The main metric that we used is *recognition accuracy*, which is the ratio of the total number of behaviours correctly identified by the algorithm over the total number of activities used for testing.

5.5.1 Step 1: Tree building

Using the MIT PlaceLab dataset, we used a leave-two-out cross validation method to evaluate the accuracy performance. From the total of 16 days data, we used 14 days as training data and the remaining 2 days as test set. Based on the 8 partitions of training-test sets, we train a decision tree on the training data in each set, resulting in a total of 8 decision trees.

Since the sensor stream is unsegmented, the data from the sensor stream is presented to the decision tree algorithm by using a window that slides over the data. A value '1' is assigned to sensor states which are opened/on, while '0' for sensor states which are closed/off. The size of the window is determined by taking the average number of sensor observations that describe the behaviours. On average, 3 sensor

observations gave rise to a behaviour, and thus the window size is set to 3.

Given a sequence of sensor observations, we are also interested to determine the dependency between two sensors (S_i and S_j), since some sensors taken together could be more informative than taken individually. For example, switching on the bathroom heater is often followed by turning on the shower tap. In this step, we extend the data by including the temporal ordering of the sensors by considering the qualitative temporal relationship, which is *after*. If sensor S_j is triggered after sensor S_i then we have one temporal relationship for '$S_i < S_j$'. This temporal ordering of the sensors is determined when a window is slid over the sensory data.

To avoid the risk of overfitting, each tree is pruned by replacing a node in a subtree with a leaf labelled that have the most examples of that subtree, i.e., whenever a node has a leaf with less than a defined n% examples. In this step, we chose to prune the subtree when any of the leaves beneath the node has less than 10% examples and 20% examples. Figure 5.7 illustrates an example on tree pruning based on a leaf that has less than 20% examples. The subtree S_9 has a leaf with 12% (i.e., 4/34) examples. This subtree is pruned and replaced with the class that consists of the most examples in this subtree, which is 'toileting'. Since our aim is to build trees, which can then be used to gain insights about the significance of the nodes among a set of trees, existing pruning methods [30, 94] are not applied.

Two separate steps were conducted for tree pruning. The first is to prune the subtree when any one of the leaf beneath the node consists of less than 10% examples and the second prunes the subtree that has less than 20% examples. The results of the decision trees, before and after pruning, along with the number of nodes and unique sensors on each tree are shown in Table 5.1. The results show that some trees achieved better accuracy when they are pruned while some do not. Since there is not much difference in the accuracy, we used the decision trees based on 20% pruning as the input to our next experiment in order to keep the computational costs low.

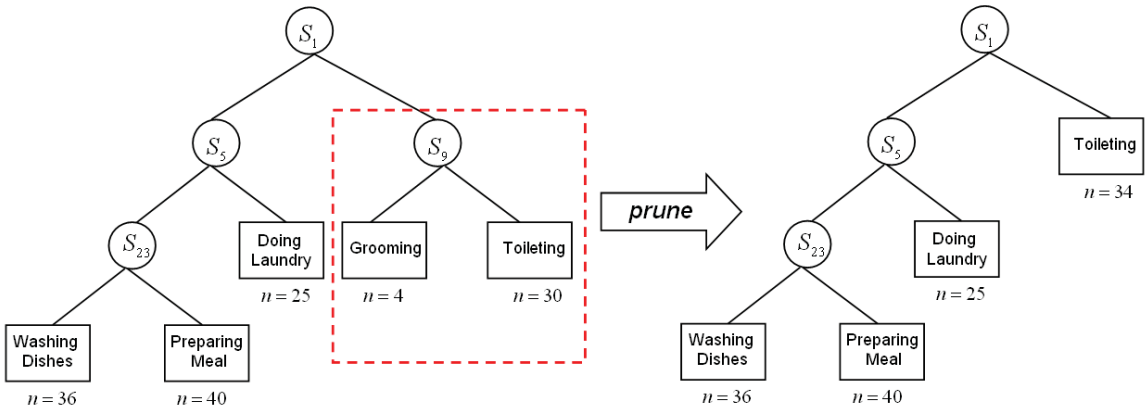


Figure 5.7: Pruning the decision tree. When any of the leaves beneath a node is less than 20% examples (shown in dashed lines), the node is replaced with a leaf labelled with the most examples of that subtree. In this example, sensor S_9 is pruned and labelled as 'toileting'.

Test Sets	Trees	Recognition Accuracy		No. of Nodes	No. of Unique Sensors
		Training	Testing		
1st Set	Original	100%	93%	28	19
	Pruning - 10%	95%	90%	16	12
	Pruning - 20%	92%	87%	14	10
2nd Set	Original	100%	96%	24	18
	Pruning - 10%	98%	94%	17	16
	Pruning - 20%	97%	91%	15	14
3rd Set	Original	100%	85%	25	19
	Pruning - 10%	98%	85%	19	16
	Pruning - 20%	94%	90%	12	9
4th Set	Original	100%	82%	26	16
	Pruning - 10%	99%	82%	24	16
	Pruning - 20%	97%	82%	19	15
5th Set	Original	100%	90%	26	20
	Pruning - 10%	99%	90%	23	20
	Pruning - 20%	98%	90%	19	17
6th Set	Original	100%	91%	26	17
	Pruning - 10%	99%	91%	21	16
	Pruning - 20%	96%	91%	16	11
7th Set	Original	100%	97%	26	20
	Pruning - 10%	99%	97%	24	20
	Pruning - 20%	97%	95%	17	15
8th Set	Original	100%	89%	24	18
	Pruning - 10%	98%	89%	18	16
	Pruning - 20%	95%	91%	12	12

Table 5.1: Results on (i) original tree before pruning, and (ii) pruned tree (when one of the leaf beneath the node consists of less than 10% examples and 20% examples) on different training/testing splits.

5.5.2 Step 2: Methods evaluation

The trees $(\tau_1, \tau_2, \ldots, \tau_8)$ that have been pruned on subtrees with less than 20% examples from Step 1 (Section 5.5.1), were used as inputs to this part of the experiment.

In this experiment, we explore the weight ranking and the two shortest distance tree methods, which are described in Section 5.4.

(a) The weight ranking method

This step ranks the sensors based on the average weight across the 8 trees. For each sensor, the algorithm calculates the sensor weights (which is the inverse level of the sensor on each tree). Based on these weights, the average weight for each sensor is calculated by dividing the total weights by the number of trees i.e., 8. The sensors are then ranked according to their average weight.

The results are shown in Table 5.2. Looking at the results, the top 3 sensors are very informative and significant for behaviour recognition. However, relying on just three sensors is not sufficient to recognise the behaviours of the inhabitant. Thus, we have chosen a cut-off point for average weight more than 0.1, which results in a total of 13 distinct sensors.

(b) The edge matching method

Using the edge matching method described in Section 5.4.2(a), we ran the algorithm on the set of 8 trees produced from Step 1 (Section 5.5.1). For each run, the input to the edge matching method is a pair of trees τ_i, τ_j; $i, j = 1, 2, \ldots, 8$. Since this method transforms one tree into the other based on the number of unmatched edges on the tree to be transformed, the distances for $dist_{ij}$ and $dist_{ji}$ are not necessarily the same, we ran the algorithm twice i.e., one for $dist_{ij}$ and another on $dist_{ji}$. For each tree τ_i, the total and average distances are calculated using Equations 5.3 and 5.4. The results are shown in Table 5.3 where τ_3 has the minimum average distance across all the 8 trees and hence is identified as the shortest distance tree with this method.

Figure 5.8 shows τ_3 decision tree. Looking at the figure, there are 9 unique sensors that are important to recognise the five different behaviours of the inhabitant. The feature values of the sensor represent the sensor state. The edges that branch toward

Sensor Details		Total Weight	Average Weight	Rank
No.	**Object Type**			
S_{16}	Toilet flush	8	1	$1st$
S_{15}	Dishwasher	3.9167	0.4896	$2nd$
S_{24}	Freezer	3.8333	0.4792	$3rd$
S_1	Laundry door	2	0.25	$4th$
S_{10}	Kitchen drawer	1.4206	0.1776	$5th$
S_{17}	Shower tap	1.3	0.1625	$6th$
$S_{23} < S_{24}$	Fridge<freezer	1.2	0.15	$7th$
S_{13}	Kitchen cabinet	1.1576	0.1447	$8th$
S_4	Medicine cabinet	0.9714	0.1214	$9th$
S_8	Bathroom drawer	0.954	0.1192	$10th$
S_6	Bathroom tap (cold)	0.8552	0.1069	$11th$
S_{12}	Kitchen cabinet	0.8468	0.1059	$12th$
S_3	Laundry dryer	0.8333	0.1042	$13th$
$S_{10} < S_{15}$	Kitchen drawer<dishwasher	0.8095	0.1012	$14th$
S_5	Bathroom tap (hot)	0.5556	0.0694	$15th$
S_{18}	Bathroom exhaust fan	0.4242	0.053	$16th$
$S_6 < S_5$	Bathroom tap (cold)<bathroom tap (hot)	0.2909	0.0364	$17th$
$S_{13} < S_{10}$	Kitchen cabinet<kitchen drawer	0.2679	0.0335	$18th$
$S_{10} < S_{13}$	Kitchen drawer<kitchen cabinet	0.254	0.0317	$19th$
S_{22}	Microwave	0.25	0.0313	$20th$
S_{23}	Fridge	0.25	0.0313	$20th$
S_2	Washing machine	0.225	0.0281	$21st$
$S_{10} < S_{11}$	Kitchen drawer<kitchen cabinet	0.1944	0.0243	$22nd$
S_{19}	Bathroom cabinet	0.1	0.0125	$23rd$
S_{21}	Oven	0.1	0.0125	$23rd$
S_{20}	Toaster	0.0909	0.0114	$24th$
S_9	Bathroom drawer	0.0769	0.0096	$25th$

Table 5.2: Results based on weight ranking method.

the right refer to sensor state that is opened/on, while the edges that branch toward the left refers to sensor state which is closed/off.

(c) The Subtree Pruning and Regraft (SPR) method

This step investigates the SPR method described in Section 5.4.2(b). We ran the SPR algorithm on the set of 8 trees output from Step 1 (Section 5.5.1). For each run, the input to the SPR method is a pair of trees τ_i , τ_j ; $i, j = 1, 2, \ldots, 8$. The distance is calculated by the number of cuts made (see the second and third steps of the SPR algorithm in Section 5.4.2(b)). Since the distance from $dist_{ij}$ and $dist_{ji}$ is the same, we ran the algorithm once for each pair of trees.

For each tree τ_i ; $i = 1, 2, \ldots, 8$, the total and average distances are calculated using Equations 5.3 and 5.4. The results are shown in Table 5.4. Since τ_3 has the minimum average distance across all the 8 trees, it is again identified as the shortest

Trees	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	Distance across 8 trees	
									Total Distance	Average Distance
τ_1	–	10	6	8	7	4	8	10	53	7.57
τ_2	11	–	12	12	10	11	12	10	78	11.14
τ_3	4	9	–	5	5	2	5	6	36	**5.14**
τ_4	13	16	12	–	11	12	15	15	94	13.43
τ_5	12	14	12	11	–	11	13	13	86	12.29
τ_6	6	12	6	9	8	–	7	8	56	8
τ_7	10	14	9	12	10	7	–	13	75	10.71
τ_8	8	7	6	8	6	4	8	–	47	6.71

Table 5.3: Results based on edge matching method

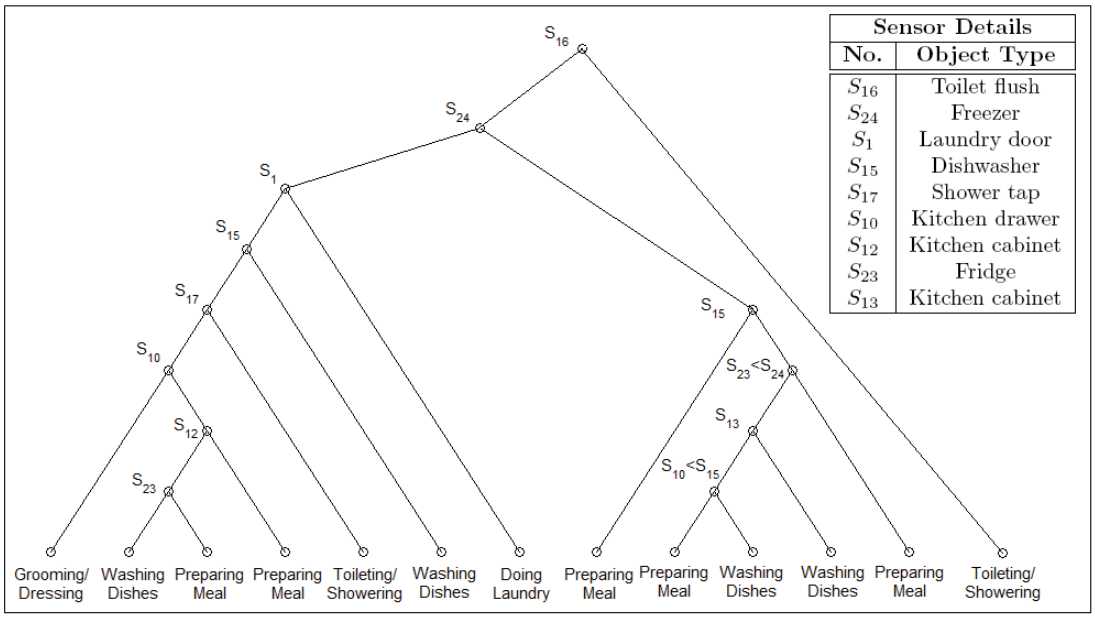


Figure 5.8: The τ_3 decision tree clearly shows that there are 9 unique informative sensors that classify 5 behaviours: doing laundry, grooming/dressing, washing/putting away dishes, toileting/showering and preparing meal/snack/beverages. Edges that branch towards the right from the node refer to sensor state that is opened/on, while edges that branch towards the left refer to sensor state that is closed/off.

Trees	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7	τ_8	**Distance across 8 trees**	
									Total Distance	**Average Distance**
τ_1	–	21	10	17	19	10	19	18	114	16.29
τ_2	21	–	19	26	22	23	24	15	150	21.43
τ_3	10	19	–	17	17	8	15	12	98	**14**
τ_4	17	26	17	–	22	19	28	23	152	21.71
τ_5	19	22	17	22	–	19	24	19	142	20.29
τ_6	10	23	8	19	19	–	15	12	106	15.14
τ_7	19	24	15	28	24	15	–	21	146	20.86
τ_8	18	15	12	23	19	12	21	–	120	17.14

Table 5.4: Results based on subtree pruning and regraft (SPR) method.

Shortest Distance Tree	
Edge Matching Method	**Subtree Pruning and Regraft Method**
τ_3	τ_3
τ_8	τ_6
τ_1	τ_1
τ_6	τ_8
τ_7	τ_5
τ_2	τ_7
τ_5	τ_2
τ_4	τ_4

Table 5.5: The order of trees (in ascending order) according to their average distance between the edge matching method (Table 5.3), and subtree and pruning regraft (SPR) method (Table 5.4).

distance tree with the SPR method.

A comparison between the order of trees based on the results of edge matching and SPR methods is shown in Table 5.5. As the table shows, both edge matching method and SPR method identified τ_3 as the shortest distance tree.

When comparing the 9 informative sensors in τ_3 identified using the shortest distance tree approach, with the top 9 sensors identified using weight ranking method in Table 5.2, we found that 8 informative sensors from the weight ranking method appear in τ_3. This equivalent results showed that all three methods identically identify the top 8 most informative sensors for behaviour recognition.

5.5.3 Step 3: Validation through classification

The aim of this step is to use those informative sensors identified in the previous step (Section 5.5.2) to recognise the inhabitant's behaviours. This is important to determine how many and which sensors are needed in a home in order to recognise the inhabitant's behaviours.

To test the efficacy of the methods, we trained 2 supervised learning algorithms (i.e., decision tree and HMMs) and an unsupervised compression-based method on the informative sensors identified from each method – 13 informative sensors identified using weight ranking method (Section 5.5.2(a)) and 9 informative sensors identified using edge matching (Section 5.5.2(b)) and SPR (Section 5.5.2(c)) methods. We used the original set of 24 sensors as a baseline to test how effective our methods are.

In order to train the supervised and unsupervised learning algorithms, a subset of the data was required. We partitioned the data into a training set consisting of the first few days, followed by a test set consisting of the remainder. It is obviously important that both the training and testing periods show examples of all the behaviours that should be recognised. We use the same MIT PlaceLab dataset. However, the evaluation was done on the dataset where the less informative sensors were removed. For example, when evaluating the 9 identified informative sensors, we removed a total of 15 less informative sensors from the dataset. By removing these sensors from the dataset, we can then determine how effective the 9 informative sensors used to recognise the inhabitant's behaviours. After removing the less informative sensors from the dataset, we re-partitioned the data into 8 days of training and 8 days of testing. The number of activity examples and sensor tokens for each method is presented in Table 5.6.

No. of Sensors	Datasets	No. of Days	No. of Activity Examples	No. of Sensor Tokens
24	Training	8	135	827
	Testing	8	165	841

(a)

No. of Sensors	Datasets	No. of Days	No. of Activity Examples	No. of Sensor Tokens
13	Training	8	130	542
	Testing	8	158	535

(b)

No. of Sensors	Datasets	No. of Days	No. of Activity Examples	No. of Sensor Tokens
9	Training	8	101	414
	Testing	8	122	339

(c)

Table 5.6: Partitioning of the MIT PlaceLab dataset into 8 days training and 8 days testing based on (a) original set of 24 sensors (baseline), and based on results produced by (b) weight ranking method, and (c) edge matching and SPR methods.

(a) Decision tree

In this step, we trained a supervised learning method (i.e., decision tree) on the informative sensors identified in Step 2 (Section 5.5.2) for behaviour recognition. Since the results of edge matching (Section 5.5.2(b)) and the SPR methods (Section 5.5.2(c)) have identified tree τ_3 as the shortest distance tree, we reuse the decision tree τ_3 (shown in Figure 5.8) for classification. Thus, in this experiment we only need to train the decision tree on 13 sensors (from the results of weight ranking method (Section 5.5.2(a))) and on the original set of 24 baseline sensors. The decision trees were each trained on the relevant labelled data in the training set using the ID3 decision tree algorithm (shown in Table 5.6). The results of the decision tree trained on 13 sensors and 24 baseline sensors are shown in Figure 5.9 and Figure 5.10.

The data from the test set and the temporal relationship 'after' (i.e., $S_i < S_j$) are presented to the trained decision tree by sliding a window of length 3 (where the window size is determined by taking the average number of sensor observations that describe the behaviours) over the sensory data. Using the example of the trained decision tree shown in Figure 5.9, the classification for the following sensors, which have been turned on/opened $\langle S_4, S_8, S_4 < S_8 \rangle$ is 'grooming/dressing'.

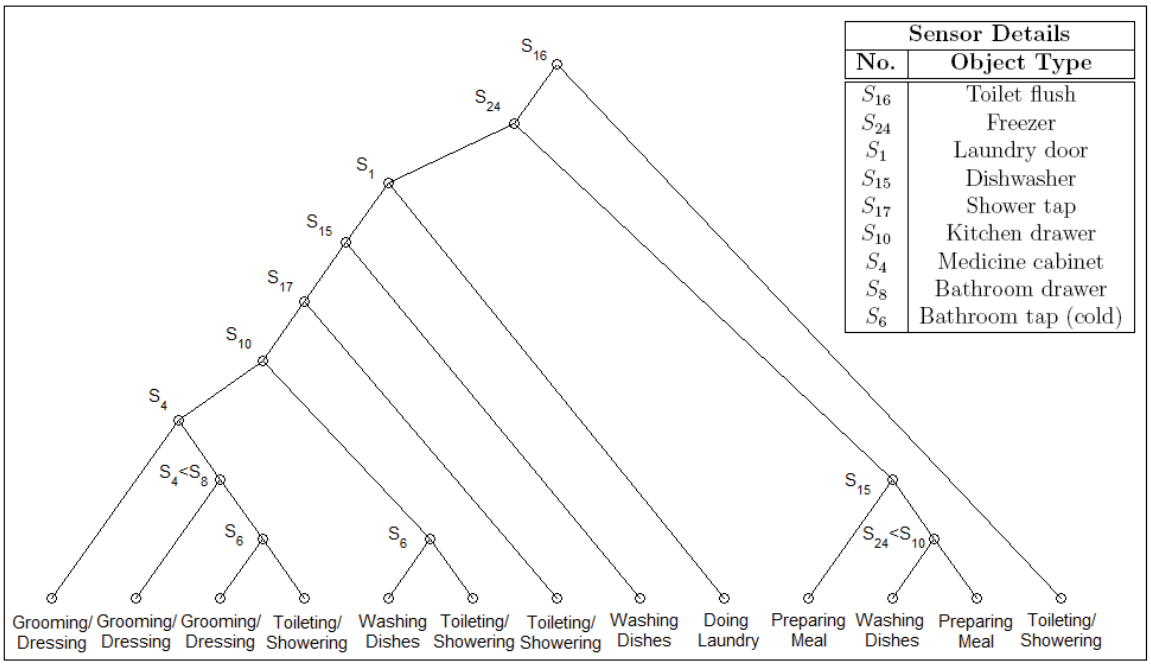


Figure 5.9: Decision tree based on weight ranking method. Edges that branch toward the right from the node refer to sensor state that is opened/on, while edges that branch toward the left refers to sensor state that is closed/off.

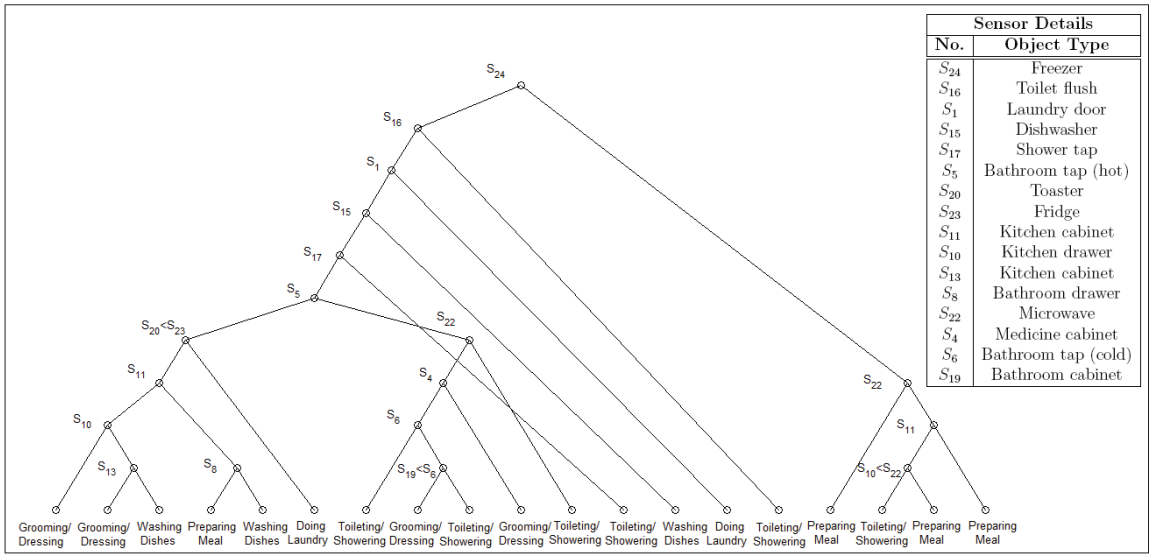


Figure 5.10: Decision tree trained on original set of baseline sensors. Edges that branch toward the right from the node refer to sensor state that is opened/on, while edges that branch toward the left refer to sensor state that is closed/off.

No. of Sensors	Supervised Learning		Unsupervised
	Decision Tree	HMMs	Compression-based Method
Original set of 24 Sensors (*baseline*)	89%	92%	79%
13 Sensors (*weight ranking method*)	92%	90%	76%
9 Sensors (*edge matching and SPR methods*)	91%	89%	70%

Table 5.7: Comparison between the supervised learning methods (decision tree and HMMs) and the unsupervised compression-based method trained on different number of sensors output from weight ranking, edge matching and SPR methods.

The results in Table 5.7 show that decision tree trained on 13 sensors (identified using the baseline weight ranking method) achieve an accuracy of 92% and an accuracy of 91% when trained on 9 sensors (identified using edge matching and SPR methods). These results are comparable to the original set of 24 baseline sensors with an accuracy of 89%.

(b) Hidden Markov Model (HMM)

In this step, we trained another supervised learning method, which is based on HMM on the identified informative sensors for behaviour recognition. We trained a set of HMMs that each recognises different behaviours following the method described in Chapter 3. The HMMs were each trained on the relevant labelled data in the training set (shown in Table 5.6) using the standard Expectation-Maximization (EM) algorithm. The sensor observations are presented to the set of trained HMMs by sliding a window of size 10 across the sensor stream (test set) and segmentation is performed following the method described in Section 3.3. The results are presented in Table 5.7.

As can be seen from the results shown in Table 5.7, when the HMMs trained on the set of informative sensors identified using our methods, it achieves an accuracy of 90% and 89%, which is quite close compared to the HMMs trained on the original set of 24 sensors with an accuracy of 92%. This shows that the HMMs trained on the set

of informative sensors achieve an acceptable accuracy compared to the original set of baseline sensors.

Note that our aim in step 3 of this experiment is to test how well the identified informative sensors generalise to different learning algorithms, and not to compare among the supervised methods and determine which is the best. We also tested our methods on the unsupervised learning algorithm, which is described next.

(c) The unsupervised compression-based method

Using the unsupervised compression-based method described in Chapter 4, we ran the LZW algorithm on the 8 days training data and then quantised the dictionary using edit distance. We then segmented the sensor stream into behaviours by parsing the tokens from the test set into the set of quantised words in the dictionary. The tokens which have been segmented into behaviours are validated against the ground truth annotations. The results presented in Table 5.7 shows that the unsupervised compression-based method able to identify patterns even on limited training data.

In comparison to the results in Tables 3.2 and 4.3, the low recognition performance observed in this experiment is mainly due to the limited training data used, which is based on 8 days training, while the results in Tables 3.2 and 4.3 are based on 14 days training data. We believe that with more training examples, the recognition accuracy could be improved since the compression works well when the behaviours are repeatedly seen in the data.

From this experiment, it shows that given a set of informative sensors (13 and 9 sensors), we can still achieve an accuracy comparable to the original set of 24 sensors. This shows that the information gain can be used to first identify the informative sensors, and then taking only those sensors identified by the tree, we can use the weight ranking or the shortest distance tree approaches (i.e., edge matching and SPR methods) to gain insights about the significance of the informative sensors identified

across a set of decision trees.

5.6 Summary

In this chapter, we have presented a method which identifies informative sensors from a set of sensors and uses them to recognise an inhabitant's behaviours. This method is based on information gain measure, which is modelled in the form of a decision tree.

However, there may be more than one tree produced when trained with different partitions of training data. We present two approaches to gain insights about the significance of the informative sensors across the different decision trees. The first approach is to rank the sensors according to their relative weight and the second approach is to identify the shortest distance tree. The central idea of shortest distance tree is to identify a tree where the distance between this tree and all given trees is minimal. Two methods were demonstrated, one is based on edge matching and another is based on subtree pruning and regraft.

To test the efficacy of our methods, we used a real smart home dataset. Using each partition of training data, we trained a decision tree. We evaluated our methods on the set of trees produced. Based on the identified informative sensors, we trained two supervised learning methods based on decision tree and HMMs (described in Chapter 3), and the unsupervised compression-based method (described in Chapter 4) for behaviour recognition. We compared the supervised and unsupervised methods with the original set of baseline sensors.

The results are promising, achieving about 90% recognition accuracy when trained on the set of informative sensors with a supervised learning method and more than 70% accuracy with an unsupervised learning method. The encouraging results have shown that our methods work effectively to identify the set of informative sensors and yet able to achieve an accuracy comparable to the original set of baseline sensors.

Our methods not only reduce the number of irrelevant and/or redundant sensors but the output can also be used to guide the future sensor installation in a new home.

Chapter 6

Conclusions

This final chapter provides an overview of the research and highlights the significant findings of this thesis to the application of assisted living. This chapter ends with a discussion on opportunities for continued research.

6.1 Discussion and Significant Findings of the Thesis

This thesis has considered the task of recognising behaviours from a sequence of sensor readings output from the house: one using a supervised learning approach on a labelled sensor stream and another using an unsupervised learning approach on unannotated ones, and the task of selecting sensors that are relevant for behaviour recognition.

Chapter 1 of this thesis began by first describing what behaviour recognition is, outlining the behaviour recognition problem, and why it deserves further study. The challenges that make the behaviour recognition task difficult were presented and the dataset used in this thesis was introduced. The aims of the research were introduced along with a set of objectives that this thesis should accomplish.

The literature review conducted in Chapter 2 covered the recent work in behaviour recognition. The reviews have shown that behaviour recognition has a wide range

of applications, and is not only useful for assisted living. It was also shown that the types of sensor that could be fitted into the house depend on the particular types of behaviour that are to be recognised. It was demonstrated that the existing methods described in the literature do not satisfy the criteria that are considered in this thesis, in particular segmentation of the sensor stream into behaviours and learning behaviours of variable length, and identifying the set of informative sensors to effectively recognise the inhabitant's behaviours.

Chapter 3 presented a supervised learning method to recognise behaviours and perform segmentation on a labelled sensor stream. This method was based on a set of trained hidden Markov models (HMMs) where each HMM represents one behaviour. The data from the sensor stream is presented to the set of trained HMMs using a variable window that moves over the sequence of observations. The central idea is to slide a window of default size over the sensor sequence and presenting all the sensor observations in the window to the set of trained HMMs for competition. A winning behaviour is selected based on the HMM that maximises the likelihood of the observations in the window. Since it is unlikely that all of the sequences in the window belong to one behaviour, a re-segmentation is performed by using the forward algorithm to calculate the likelihood of each observation in the window according to the winning HMM. This ensures that the majority of the behaviours in the window are recognised.

Experiments with this method were reported in Section 3.5. The first experiment in Chapter 3 has demonstrated the idea of competition between HMMs and re-segmentation of the sensor stream. It was demonstrated that this method can effectively recognise the behaviours and perform segmentation on the sensor stream, achieving an average accuracy of 91%. The second experiment in this chapter demonstrated that the HMMs with variable window length outperformed the HMMs with fixed window length. The problems of using the fixed window length are due to the

fact that (1) only one winning behaviour could be identified at each time, resulting in other behaviours in the window not being identified and (2) the problem of identifying the correct window size; if the window size is too large, classification is biased towards the behaviour that was described with the most sensors and if the window size is too small, classification is biased towards the behaviour that was described with fewer sensors. The effects of different default window sizes on computation and recognition performances were demonstrated in Section 3.5.3, and it was observed that the algorithm takes longer to run and the recognition performance decreases with an increase in the window size. Thus, a shorter window size is preferred to reduce the computational cost. The experiment demonstrated in Section 3.5.4 analysed the amount of training data needed to train the HMMs. Results showed that the method does not need a large set of training data.

Other supervised learning methods that could be used for behaviour recognition and perform segmentation on a sensor stream were also discussed. The problems with the supervised learning method were described; it generally relies on labelled data, where a person has to manually label the behaviours at each time. This is particularly a time consuming process, resulting in relatively small datasets, and is also prone to significant errors as people do not pinpoint the end of one behaviour and commencement of the next correctly.

Chapter 4 introduced an unsupervised learning solution to recognise human behaviours and perform segmentation on the unlabelled sensor stream. The literature review of unsupervised learning method has shown that the standard unsupervised learning methods are not suitable for behaviour recognition where the data sequences are of variable length. A solution based on compression and edit distance was proposed. Our method was shown to be able to automatically cluster the unlabelled data and segment the sensor stream into behaviours. It exploits the fact that a behaviour is a pattern that is a repeated set of tokens, and can thus be identified using a compres-

sion algorithm that exploits redundancy in the input stream. The Lempel-Ziv-Welch (LZW) algorithm was first applied to the unlabelled data stream, which creates a dictionary of substring. In order to allow variations in the behaviours, the dictionary is quantised using edit distance, which is also used to segment the unlabelled sensor stream into behaviours.

Experiments with the unsupervised learning method were reported in Section 4.4. The effect of edit distance threshold values for dictionary quantisation was demonstrated in Section 4.4.1. It was observed that the most effective way to quantise the dictionary is to use a threshold value of 1; increasing the edit distance threshold value results in a greater reduction on the dictionary size and higher chance of losing good representative examples. It was demonstrated in the second experiment (Section 4.4.2) that the unsupervised compression-based method outperformed the SOM on two experiments: one on Fisher's iris dataset where the inputs are of fixed dimensionality and another on MIT PlaceLab smart home dataset where the data sequences are of variable length. The proposed unsupervised compression-based approach was able to identify patterns that correspond to the human behaviours and addressed the generalisability problem, since the recognition system can now learn from scratch to recognise behaviours in different home settings without any prior human labelling or knowledge about the sensor stream. The final experiment in Chapter 4 was designed to demonstrate the fact that the output of the unsupervised compression-based method can be used as a way to provide labels to training data for a supervised algorithm. The HMM method presented in Chapter 3 was applied to perform segmentation and behaviour recognition, which was also used as a baseline to compare the unsupervised compression-based approach. It has been shown that by adding a supervised method, the system could potentially be used to identify abnormal behaviours.

Both Chapters 3 and 4 presented a solution to recognise behaviours from two

different directions: one using a supervised learning method (Chapter 3) and another using an unsupervised learning method (Chapter 4). Chapter 5 presented a solution to address the sensor selection problem. A literature review was conducted and highlighted that most work were based on identifying which particular body locations are best to place the accelerometers in order to recognise human physical activities. These methods were not effective when applied to state-change sensors as the number of sensors involved in a home setting can be much larger and span across different room locations (unlike the accelerometers attached to the body).

A solution based on information gain was proposed to address the sensor selection problem in a home setting in Chapter 5. Information gain was used to measure the importance of the sensors and was modelled in the form of a decision tree. The central idea was to identify a set of 'informative' sensors that could provide the most information about the inhabitant's behaviours by removing irrelevant and/or redundant sensors that do not contribute to behaviour recognition. This chapter also introduced three solutions to gain insights about the significance of the informative sensors identified across the different decision trees, which were presented in Section 5.4. The first approach was based on weight ranking. The main idea is to assign each sensor a weight and rank them according to their score. The second approach was based on edge matching, while the third approach was based on subtree pruning and regraft (SPR). Both edge matching and SPR methods aims to identify a shortest distance tree where the distance between this tree and all other trees is minimal.

An experiment was conducted to demonstrate the efficacy of using information gain to identify the set of informative sensors and the efficacy of the weight ranking, edge matching and SPR methods to gain insights about the significance of the informative sensors identified across different decision trees. This experiment was conducted in three steps. The first step of the experiment demonstrated that the informative sensors that were useful for behaviour recognition can be identified ef-

fectively using information gain (Section 5.5.1). A total of 8 decision trees were produced and used as an input to the next step, which evaluated the weight ranking, edge matching and SPR methods (Section 5.5.2). It was demonstrated that all these methods identically identified the informative sensors, although these methods were conceptually different. The identified informative sensors were validated using three methods, which were presented in this thesis, i.e., HMM (Chapter 3) and decision tree (Chapter 5), as well as the unsupervised compression-based method (Chapter 4). It was demonstrated that the reduced set of sensors effectively recognise the inhabitant behaviours and that they could be used to guide future sensor installation in a new home (Section 5.5.3).

It was also demonstrated that the decision tree outperformed HMM (although not significantly) when trained on limited sensors. This might not always be adequate depending upon what the aim of the smart home is. For example, suppose that we wish to recognise variations in behaviours, which could highlight potential abnormality. These would not be easy using the decision tree. The HMM method presented in Chapter 3 could address this problem.

All the three objectives described in Section 1.4.2 were fulfilled in this thesis. The first objective, which is to develop a supervised learning method was fulfilled in Chapter 3. The algorithm has the ability to perform behaviour recognition and segmentation simultaneously on the sensor stream (Section 3.3). The algorithm is able to recognise the majority of the behaviours in the window (Section 3.3). This includes the number of variation in the behaviours and the tokens that are shared by multiple behaviours. The algorithm is scalable when trained on limited training data (Section 3.5.4).

The second objective, which is to develop an unsupervised learning algorithm was fulfilled in Chapter 4. This includes learning of patterns of variable length (Section 4.3.1) and segmentation of the sensor stream (Section 4.3.2). The algorithm

has the ability to generalise to different homes since it can learn from scratch and identify patterns that correspond to the inhabitant's behaviours (Section 4.4.2). The algorithm is able to provide labels to training data for a supervised algorithm (Section 4.4.3).

The third objective was fulfilled in Chapter 5. This includes using information gain to identify the set of informative sensors (Section 5.5.1) and solutions to gain insights about the significance of the informative sensors across the different decision trees (Section 5.5.2). Both supervised and unsupervised methods were trained on the identified informative sensors and they performed well to recognise the inhabitant's behaviours (Section 5.5.3).

6.1.1 Thesis Contributions

This thesis explores the field of behaviour recognition in the smart home context. The research makes contributions in terms of developing machine learning algorithms, which are centered around two main research questions:

1. How can we recognise the inhabitant's behaviours?
2. Which sensors are useful to effectively recognise the inhabitant's behaviours?

The following list our contributions to the behaviour recognition problem:

- We proposed a *supervised learning* method based on hidden Markov model and variable window length, which can do the following:
  - simultaneously perform segmentation and behaviour recognition on the sensor stream
  - learn behaviours from labelled sensor stream
  - recognise variations in behaviours

- We introduced an *unsupervised learning* approach based on compression and edit distance between words to learn the mapping between sensor outputs and behaviours. The algorithm developed as part of the research can:

    - learn the sensory data in an unsupervised way, i.e., without any prior human labelling
    - perform segmentation of the sensor stream into suitable patterns
    - identify patterns that correspond to human behaviours
    - recognise variations in behaviours

The contributions of the research to the sensor selection problem is two-fold, which are listed below:

- We used an information-theoretic approach based on information gain, embedded within a decision tree learning algorithm to identify the set of informative sensors for behaviour recognition.

- We proposed three methods to gain insights about the significance of the informative sensors identified among a set of trees. These methods are:

    i. ranking sensors by weights
    ii. edge matching
    iii. subtree pruning and regraft

6.2 Prospects for Future Research and Open Questions

The methods proposed in this thesis have been shown to be promising to address the behaviour recognition and sensor selection problems. However, there are other

interesting properties that have not been explored, which we believe merit further study. In this section, we outline the directions for future research.

The first two suggestions are on how to include additional information that could help to recognise abnormal behaviours or potentially dangerous behaviours. The third suggests a way to extend the current HMM method to recognise interleaving behaviours, while the fourth looks at recognising behaviours from multiple inhabitants in the home. The final two suggestions are on how the smart home can continuously learn to adapt to inhabitant's ongoing changing behaviours.

6.2.1 Adding spatio-temporal and context reasoning

The supervised and unsupervised learning methods presented in Chapters 3 and 4 of this thesis posit that a typical behaviour is a sequence of activities that occur close to one another in time. It does not consider activities such as laundry, which may well be separated in time (while waiting for the washer to finish) and in space (for example, if clothes are hung outside rather than using a dryer).

For a smart home to react intelligently to its inhabitant's needs, the system needs to recognise their behaviour and to use spatio-temporal information, such as where (in which room?) and when (at what time?) a particular event occurred? Additionally, and possibly more important is the contextual information (how was the current situation reached, what else is happening and what is the state of the environment). Representing all of this information in the smart environment is a significant challenge and worthy further research.

6.2.2 Exploring other type of sensor and smart home datasets for behaviour recognition

This thesis focuses on state-change sensors, which are attached to household objects in the home. We are also interested in exploring other types of sensors for behaviour

recognition such as motion sensors and accelerometers.

One of the problems found when using state-change sensors for behaviour recognition is that it is rather difficult to identify the 'cleaning' activity since it involves the same set of sensors that other activities would use. For example, both cleaning and showering behaviours will trigger the same sensors. We believe that the motion sensor (described in Section 2.3.3) could provide additional information to help discriminate between these two behaviours. In future, we plan to apply our methods on the dataset that consists of both objects and motion sensors, such as the CASAS dataset [21], and investigate on how far this can improve the recognition performance.

We are also interested in applying our methods on accelerometers, which could be helpful to detect potentially dangerous behaviours such as when a person has fallen or vital body signs indicate deterioration of health conditions. In order to apply the unsupervised compression-based method, we need to discretize the acceleration signals into a sequence of tokens. One of the methods that could achieve this is to use the symbolic aggregation approximation (SAX) [79]. The central idea of SAX is to transform signals into equal segments, where each segment is then converted into symbols. We plan to further our work in this area, and ultimately perform a comparison of our methods with other methods.

One option that we have not explored in this thesis, but will in the future, is to test the generalisation of our methods on other smart home datasets [21, 89, 136]. Since the number of sensors used and the activities to be recognised are different and that the data are collected in different number of days, a new experimental setup needs to be designed before testing our methods on these datasets.

6.2.3 Recognising interleaving behaviours

In this thesis, we only considered disrete-time stochastic processes, i.e., every time a new sensor observation arrives, the time t is increased by one to $t + 1$. However,

this is not always the case in the real world environment, as behaviours are normally interleaved. For example, while cooking lunch, the person may require a trip to the toilet or the cat may demand to be fed, which could be done while the washing machine is running.

The strict independence assumption between observations in HMM (described in Section 3.2.3) makes it difficult to model multiple interacting behaviours. The conditional random field (CRF) [132], which relaxes this assumption, has shown some promise to model interleaving behaviours. The CRF is an undirected graph, which encodes the independence relationship between the labels and not the observations. However, learning in CRF is computationally expensive since it needs to perform inference repeatedly during training when the model has complex multiple interleaving behaviours or when there is a large number of training examples. Addressing these issues could be a promising area for new research.

6.2.4 Multiple inhabitants

This thesis assumed that the house consists of only a single inhabitant. In a real world environment, it is possible that there could be more than one inhabitant living in the home or check-up visits by the caregiver, or visits made by families/friends. In either way, the recognition system should have the ability to recognise behaviours in the presence of multiple inhabitants.

One of the challenges of recognising behaviours from multiple inhabitants is that the sensors from different locations could be activated at the same time since different inhabitants could perform activities separately (e.g., one person is showering in the bathroom, while another is doing laundry in the laundry area). Another challenge is the fact that both the inhabitants could perform the same activity, e.g., cooking, which could result many overlapping events for the same activity performed by different inhabitants.

One way to address this is to enable each inhabitant to wear a tag so that when the sensors are activated, the identification information from the tag can help to make a distinction between the inhabitants performing the activity. However, as described in previous section, behaviours may be interleaved. Thus, recognising interleaving behaviours between multiple inhabitants is another challenging and interesting area of study.

6.2.5 Abnormality detection

In this thesis, we mainly focus on recognising the inhabitant's behaviours. However, for a smart home to support the inhabitant, the system should not only recognise behaviours but also monitor potential abnormality. A system could learn whether a novel input presented to it is a completely new behaviour, an additional information to describe the current learned behaviours (which may be due to newly added sensors) or an abnormal behaviour. The challenges in this task are that abnormal behaviours are rarely seen in the data, and since people have variations in their behaviours and may react differently to different situations, it is quite difficult to have a training set that consists of all the example of every behaviour.

The majority of the works [29, 56, 119] reported in the literature address abnormalities by learning on a set of normal behavioural patterns and any deviation from that trained habitual patterns is considered as abnormal. In situations where an abnormal behaviour could be a completely new behaviour (e.g., sitting around the fireplace during winter), cannot be tracked and learned by the present models since the previously learned models do not accommodate new behaviours.

One of the methods that could address this is the novelty detection, which has the ability to handle inputs that are not seen during training. The central idea of novelty detection is to train on a set of normal behaviours and then using the learned 'normal' behavioural models to identify inputs that do not fit into the pattern of the

training set [88]. For the supervised method described in Chapter 3, when the system detects a novel feature that belongs to an existing behaviour, the system can add this new piece of information to the previously learned HMM, or otherwise initialise a completely new HMM and train it. Recognising potential abnormality in the smart environment is another challenging and promising area of research.

6.2.6 Life-long learning

For a smart home to react intelligently to its inhabitant's needs, the recognition system should not only recognise the inhabitant's behaviours but continuously learn and adapt to the inhabitant's ongoing changing behaviours.

The majority of the work reported in this thesis consider learning in a fixed environment. This might not be the case in the real smart home, since both behaviours and environment can change over time. This means that the recognition system needs to adapt to changes in behavioural patterns over time without forgetting previously learned behaviours.

In fact, many of the learning systems are facing the 'plasticity-stability dilemma' i.e., how to continuously learn new things (plasticity) while preserving the previously learned knowledge (stability) [12, 124]. Life-long learning goes beyond on-line learning by preserving old knowledge while new knowledge is added, and using the currently learned knowledge to guide future learning. For the supervised and unsupervised methods described in Chapters 3 and 4, we can add HMMs to learn new behaviours and remove old ones. The problem is how to retrain them. Thus, addressing this is one promising area of research.

Appendix A

Declaration of Previous Work

Parts of this thesis are based on previously published materials as follows:

- Sook-Ling Chua, Stephen Marsland, and Hans W. Guesgen. Unsupervised Learning of Human Behaviours. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI)*, pages 319–324, 2011.

- Sook-Ling Chua, Stephen Marsland, and Hans W. Guesgen. Unsupervised Learning of Patterns in Data Streams using Compression and Edit Distance. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1231–1236, 2011.

- Sook-Ling Chua, Stephen Marsland, and Hans W. Guesgen. Behaviour recognition in smart homes. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2788–2789, 2011.

- Sook-Ling Chua, Stephen Marsland, and Hans W. Guesgen. Towards behaviour recognition with unlabelled sensor data: as much as necessary, as little as possible. IGI Global, 2010. *To appear.*

- Joe Steinhauer, Sook-Ling Chua, Hans W. Guesgen, Stephen Marsland. Utilising Temporal Information in Behaviour Recognition. In *AAAI Spring Symposium Special Session: It's all in the Timing*, pages 54–59, 2010.

- Sook-Ling Chua, Stephen Marsland, and Hans W. Guesgen. Behaviour recognition from Sensory Streams in Smart Environments. In *Australasian Conference on Artificial Intelligence (AI09)*, pages 666–675, 2009.

- Sook-Ling Chua, Stephen Marsland, and Hans W. Guesgen. Spatio-Temporal and Context Reasoning in Smart Homes. In *Proceedings of the International Conference on Spatial Information Theory (COSIT) Workshop on Spatial and Temporal Reasoning for Ambient Intelligence Systems*, pages 9–20, 2009.

Bibliography

[1] Tarik Al-ani, Quynh Trang Le Ba, and Eric Monacelli. On-line automatic detection of human activity in home using wavelet and hidden Markov models Scilab toolkits. In *IEEE International Conference on Control Applications (CCA 2007)*, pages 485–490, October 2007.

[2] Benjamin L. Allen and Mike Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5:1–15, 2001.

[3] Juan Carlos Augusto and Chris D. Nugent. The use of temporal reasoning and management of complex events in smart homes. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI), including Prestigious Applications of Intelligent Systems (PAIS)*, pages 778–782. IOS Press, 2004.

[4] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Pervasive*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 2004.

[5] Timothy Bell, John G. Cleary, and Ian H. Witten. *Text compression*. Prentice Hall, Englewood Cliffs, NJ, 1990.

[6] Avrim L. Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, December 1997.

[7] Magnus Bordewich and Charles Semple. On the computational complexity of the rooted subtree prune and regraft distance. *Annals of Combinatorics*, 8:409–423, 2004.

[8] François Brémond, Monique Thonnat, and Marcos Zúñiga. Video-understanding framework for automatic behavior recognition. *Behavior Research Methods*, 38:416–426, 2006.

[9] Leonard A. Breslow and David W. Aha. Simplifying decision trees: A survey. *Knowledge Engineering Review*, 12(1):1–40, January 1997.

[10] Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293, 2000.

[11] David Bryant. A classification of consensus methods for phylogenetics. In *Bioconsensus*, Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, pages 163–183. DIMACS-AMS, 2001.

[12] Gail A. Carpenter and Stephen Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988.

[13] Jianfeng Chen, Jianmin Zhang, Alvin H. Kam, and Louis Shue. An automatic acoustic bathroom monitoring system. In *IEEE International Symposium on Circuits and Systems (ISCAS 2005)*, volume 2, pages 1750–1753, May 2005.

[14] Liming Chen, Chris Nugent, Maurice Mulvenna, Dewar Finlay, Xin Hong, and Micheal Poland. A logical framework for behaviour reasoning and assistance in a smart home. *International Journal o Assistive Robotics and Mechatronics (ARM)*, 9(4):20–34, December 2008.

[15] Yi-Han Chen, Ching-Hu Lu, Kuo-Chung Hsu, Li-Chen Fu, Yu-Jung Yeh, and Lun-Chia Kuo. Preference model assisted activity recognition learning in a smart home environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, pages 4657–4662, October 2009.

[16] Pau-Choo Chung and Chin-De Liu. A daily behavior enabled hidden Markov model for human behavior understanding. *Pattern Recognition*, 41:1589–1597, May 2008.

[17] Rudi Cilibrasi and Paul M. B. Vitányi. Clustering by compression. *IEEE Transactions on Information Theory*, 51(4):1523–1545, 2005.

[18] John G. Cleary and William J. Teahan. Unbounded length contexts for PPM. In *The Computer Journal*, volume 40, pages 67–75, 1997.

[19] John G. Cleary and Ian H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32:396–402, 1984.

[20] Charles J. Cohen, Katherine A. Scott, Marcus J. Huber, Steven C. Rowe, and Frank Morelli. Behavior recognition architecture for surveillance applications. In *37th IEEE Applied Imagery Pattern Recognition Workshop (AIPR'08)*, pages 1–8, October 2008.

[21] Diane Cook, M. Schmitter-Edgecombe, Aaron Crandall, Chad Sanders, and Brian Thomas. Collecting and disseminating smart home sensor data in the CASAS project. In *Proceedings of CHI'09 Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*, 2009.

[22] Diane J. Cook, Michael Youngblood, Edwin O. III Heierman, Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja. MavHome: an agent-

based smart home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 521–524, March 2003.

[23] Aaron S. Crandall and Diane J. Cook. Using a hidden Markov model for resident identification. In *Proceedings of the 2010 Sixth International Conference on Intelligent Environments*, IE'10, pages 74–79, Washington, DC, USA, 2010. IEEE Computer Society.

[24] Mohamed Dahmane and Jean Meunier. Real-time video surveillance with self-organizing maps. In *Proceedings of the 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, pages 136–143, Victoria, BC, Canada, 2005. IEEE Computer Society.

[25] Sajal K. Das, Dianne J. Cook, Amiya Battacharya, Edwin Heierman, and Tze-Yun Lin. The role of prediction algorithms in the Mavhome smart home architecture. *Wireless Communications, IEEE*, 9(6):77–84, 2002.

[26] Glen Debard, Peter Karsmakers, Mieke Deschodt, Ellen Vlaeyen, Jonas Van den Bergh, Eddy Dejaeger, Koen Milisen, Toon Goedemé, Tinne Tuytelaars, and Bart Vanrumste. Camera based fall detection using multiple features validated with real life video. In *Workshop Proceedings of the 7th International Conference on Intelligent Environments (IE 2011)*, pages 441–450. IOS Press, 2011.

[27] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[28] Kim Dunstan and Nicholas Thomson. Demographic aspects of New Zealand's ageing population, 2006. *http://www.stats.govt.nz/products-and-services/papers/demographic-aspects-nz-ageing-population.htm.*

[29] Thi V. Duong, Hung H. Bui, Dinh Q. Phung, and Svetha Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-Markov model. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 838–845. IEEE Computer Society, 2005.

[30] Floriana Esposito, Donato Malerba, and Semeraro Giovanni. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:476–491, May 1997.

[31] Usama M. Fayyad and Keki B. Irani. What should be minimized in a decision tree? In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI) - Volume 2*, pages 749–754. AAAI Press, 1990.

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

[33] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[34] A. Frank and A. Asuncion. UCI machine learning repository. http://archive.ics.uci.edu/ml, 2010.

[35] Jiang Gao, Alexander G. Hauptmann, Ashok Bharucha, and Howard D. Wactlar. Dining activity analysis using a hidden Markov model. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR2004)*, volume 2, pages 915–918, August 2004.

[36] Dinesh Govindaraju and Manuela Veloso. Learning and recognizing activities in streams of video. In *Proceedings of the AAAI Workshop on Learning in Computer Vision*, 2005.

[37] Tao Gu, Shaxun Chen, Xianping Tao, and Jian Lu. An unsupervised approach to activity recognition and segmentation based on object-use fingerprints. *Data & Knowledge Engineering*, 69(6):533–544, June 2010.

[38] Donghai Guan, Weiwei Yuan, Young-Koo Lee, Andrey Gavrilov, and Sungyoung Lee. Activity recognition based on semi-supervised learning. In *Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'07)*, pages 469–475, Washington, DC, USA, 2007. IEEE Computer Society.

[39] Hans W. Guesgen and Stephen Marsland. *Spatio-Temporal Reasoning and Context Awareness*, pages 609–634. Springer, Berlin, Germany, 2010.

[40] Hani Hagras, Victor Callaghan, Martin Colley, Graham Clarke, Anthony Pounds-Cornish, and Hakan Duman. Creating an ambient-intelligence environment using embedded agents. *Intelligent Systems, IEEE*, 19(6):12–20, 2004.

[41] Michael Hallett and Catherine McCartin. A faster FPT algorithm for the maximum agreement forest problem. *Theory of Computing Systems*, 41:539–550, October 2007.

[42] Jungong Han, Peter H.N. de With, Ashley Merien, and Guid Oei. Intelligent trainee behavior assessment system for medical training employing video analysis. *Pattern Recognition Letters*, 33(4):453–461, 2011.

[43] Lykele Hazelhoff, Jungong Han, and Peter H. With. Video-based fall detection in the home using principal component analysis. In *Proceedings of the 10th International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS'08)*, pages 298–309, Berlin, Heidelberg, 2008. Springer-Verlag.

[44] Albert Hein and Thomas Kirste. Towards recognizing abstract activities: An unsupervised approach. In *Behaviour Monitoring and Interpretation*, pages 102–114, 2008.

[45] Jotun Hein, Tao Jiang, Lusheng Wang, and Kaizhong Zhang. On the complexity of comparing evolutionary trees. *Discrete Applied Mathematics*, 71:153–169, December 1996.

[46] 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.

[47] Thomas Hofmann. Probabilistic latent semantic analysis. In *UAI '99: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden*, pages 289–296. Morgan Kaufmann, 1999.

[48] 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.

[49] Derek Hao Hu and Qiang Yang. CIGAR: concurrent and interleaving goal and activity recognition. In *AAAI'08: Proceedings of the 23rd Conference on Artificial Intelligence*, pages 1363–1368, 2008.

[50] Derek Hao Hu and Qiang Yang. Transfer learning for activity recognition via sensor mapping. In *Proceedings of the 22nd International Joint Conference on Artifical Intelligence*, pages 1962–1967, 2011.

[51] Derek Hao Hu, Vincent Wenchen Zheng, and Qiang Yang. Cross-domain activity recognition via transfer learning. *Pervasive Mobile Computing*, 7(3):344–358, June 2011.

[52] Weiyao Huang, Jun Zhang, and Zhijing Liu. Activity recognition based on hidden Markov models. In *Proceedings of the 2nd International Conference on Knowledge Science, Engineering and Management*, KSEM'07, pages 532–537, Berlin, Heidelberg, 2007. Springer-Verlag.

[53] Tâm Huỳnh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp'08)*, pages 10–19, New York, NY, USA, 2008. ACM.

[54] Tâm Huynh and Bernt Schiele. Analyzing features for activity recognition. In *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient Intelligence*, pages 159–163, New York, USA, 2005. ACM.

[55] Tomohito Inomata, Futoshi Naya, Noriaki Kuwahara, Fumio Hattori, and Kiyoshi Kogure. Activity recognition from interactions with objects using dynamic Bayesian network. In *Proceedings of the 3rd ACM International Workshop on Context-Awareness for Self-Managing Systems*, pages 39–42, New York, USA, 2009. ACM.

[56] Vikramaditya R. Jakkula and Diane J. Cook. Anomaly detection using temporal data mining in a smart home environment. *Methods of information in medicine*, 47(1):70–75, 2008.

[57] A. Jehad Sarkar, Sungyoung. Lee, and Young-Koo. Lee. A smoothed naïve Bayes-based classifier for activity recognition. *The Institution of Electronics and Telecommunication Engineers (IETE) Technical Review*, 27(2):107–119, 2010.

[58] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129. Morgan Kaufmann, 1994.

[59] G. David Forney Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[60] Biing-Hwang Juang and Lawrence R. Rabiner. Hidden Markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[61] Vili Kellokumpu, Matti Pietikäinen, and Janne Heikkilä. Human activity recognition using sequences of postures. In *Machine Vision Applications*, pages 570–573, 2005.

[62] 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 *CoBuild '99: Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*, pages 191–198. Springer-Verlag, 1999.

[63] Daehwan Kim, Jinyoung Song, and Daijin Kim. Simultaneous gesture segmentation and recognition based on forward spotting accumulative HMMs. *Pattern Recognition*, 40(11):3012–3026, 2007.

[64] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, December 1997.

[65] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[66] Teuvo Kohonen, Jussi Hynninen, Jari Kangas, and Jorma Laaksonen. SOM_PAK: The self-organizing map program package. Technical report, Helsinki University of Technology, Laboratory of Computer, 1996.

[67] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *International Conference on Machine Learning*, pages 284–292, 1996.

[68] Heli Koskimäki, Ville Huikari, Pekka Siirtola, Perttu Laurinen, and Juha Röning. Activity recognition using a wrist-worn inertial measurement unit: A case study for industrial assembly lines. In *17th Mediterranean Conference on Control and Automation (MED'09)*, pages 401–405, June 2009.

[69] Andreas Krause, Daniel P. Siewiorek, Asim Smailagic, and Jonny Farringdon. Unsupervised, dynamic identification of physiological and activity context in wearable computing. In *ISWC'03: Proceedings of the 7th IEEE International Symposium on Wearable Computers*, pages 88–97, Washington, DC, USA, 2003. IEEE Computer Society.

[70] Kristof Van Laerhoven. Combining the self-organizing map and k-means clustering for on-line classification of sensor data. In *ICANN'01: Proceedings of the International Conference on Artificial Neural Networks*, pages 464–469, London, UK, 2001. Springer-Verlag.

[71] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01)*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[72] Patrick Langley and Stephane Sage. Oblivious decision trees and abstract cases. In *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 113–117, Menlo Park, CA, 1994. AAAI Press.

[73] Patrick Langley and Stephane Sage. *Scaling to domains with irrelevant features*, pages 51–63. MIT Press, Cambridge, MA, USA, 1997.

[74] Patrick Langley and Stephanie Sage. Induction of selective Bayesian classifiers. In *UAI '94: Proceedings of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 399–406, Seattle, Washington, 1994. Morgan Kaufmann.

[75] Grégoire Lefebvre and Christophe Garcia. A probabilistic self-organizing map for facial recognition. In *19th International Conference on Pattern Recognition (ICPR 2008)*, pages 1–4. IEEE, 2008.

[76] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *In Third Annual Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.

[77] Qiang Li, John A. Stankovic, Mark A. Hanson, Adam T. Barth, John Lach, and Gang Zhou. Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information. In *Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, BSN '09, pages 138–143, Washington, DC, USA, 2009. IEEE Computer Society.

[78] Lin Liao, Dieter Fox, and Henry A. Kautz. Extracting places and activities from GPS traces using hierarchical conditional random fields. *International Journal of Robotic Research*, 26(1):119–134, 2007.

[79] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. In *Proceedings of the Second Workshop on Temporal Data Mining*, Edmonton, Alberta, Canada, July 2002.

[80] Beth Logan, Jennifer Healey, Matthai Philipose, Emmanuel Munguia 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'07, pages 483–500, Berlin, Heidelberg, 2007. Springer-Verlag.

[81] Xi Long, Bin Yin, and Ronald M. Aarts. Single-accelerometer-based daily physical activity classification. In *31st Annual International Conference of the Engineering in Medicine and Biology Society (IEEE-EMBS 2009)*, pages 6107–6110, September 2009.

[82] M. Lösch, S. Schmidt-Rohr, S. Knoop, S. Vacek, and R. Dillmann. Feature set selection and optimal classifier for human activity recognition. In *The 16th IEEE International Symposium on Robot and Human interactive Communication (RO-MAN 2007)*, pages 1022–1027, August 2007.

[83] Sebastian Lühr, Hung H. Bui, Svetha Venkatesh, and George A.W. West. Recognition of human activity through hierarchical stochastic learning. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom2003)*, pages 416–422, March 2003.

[84] Paul Lukowicz, Andreas Timm-Giel, Michael Lawo, and Otthein Herzog. Wearit work: Toward real-world industrial wearable computing. *Pervasive Computing, IEEE*, 6(4):8–13, October–December 2007.

[85] Mitja Luštrek and Boštjan Kaluža. Fall detection and activity recognition with machine learning. *Informatica*, 33(1):205–212, 2009.

[86] Einat Marhasev, Meirav Hadad, and Gal A. Kaminka. Non-stationary hidden semi-Markov models in activity recognition. In *AAAI'06: Proceedings of the 21st conference on Artificial intelligence, Workshop on Modeling Others from Observations*, pages 53–60. AAAI press, 2006.

[87] Stephen Marsland. *Machine Learning: An Algorithmic Introduction.* CRC Press, New Jersey, USA, 2009.

[88] Stephen Marsland, Ulrich Nehmzow, and Jonathan Shapiro. Environment-specific novelty detection. In *From Animals to Animats, the 7th International Conference on Simulation of Adaptive Behaviour*, pages 36–45, 2002.

[89] Massey University Smart Environments (MUSE). The MUSE manifesto. http://muse.massey.ac.nz/, 2011.

[90] 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*, pages 113–116, Washington, DC, USA, 2006. IEEE Computer Society.

[91] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *In AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press, 1998.

[92] Thomas O. Meservy, Matthew L. Jensen, John Kruse, Judee K. Burgoon, and Jay F. Nunamaker. Automatic extraction of deceptive behavioral cues from video. In *Integrated Series in Information Systems*, pages 198–208, 2005.

[93] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naïve Bayes – which naïve Bayes? In *Third Conference on Email and Anti-Spam (CEAS)*, 2006.

[94] John Mingers. An empirical comparison of pruning methods for decision tree induction. *Machine Learning*, 4:227–243, November 1989.

[95] Tom Mitchell. *Machine Learning*. McGraw-Hill, New Jersey, USA, 1997.

[96] Alistair Moffat. Implementing the PPM data compression scheme. *IEEE Transactions on Communications*, 38(11):1917–1921, November 1990.

[97] Simon Moncrieff, Svetha Venkatesh, Geoff West, and Stewart Greenhill. Incorporating contextual audio for an actively anxious smart home. In *Proceedings of the 2005 International Conference on Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005*, pages 373–378, December 2005.

[98] Michael C. Mozer. Lessons from an adaptive home. In *Smart Environments: Technology, Protocols, and Applications*, pages 273–298. Wiley, 2005.

[99] Kevin P. Murphy and Mark A. Paskin. Linear time inference in hierarchical HMMs. In *Proceedings of Neural Information Processing Systems*, 2001.

[100] Pradeep Natarajan and Ramakant Nevatia. Coupled hidden semi Markov models for activity recognition. In *Proceedings of the IEEE Workshop on Motion and Video Computing*, pages 10–17, Washington, DC, USA, 2007. IEEE Computer Society.

[101] United Nations. World population prospects: The 2006 revision. *http://www.un.org/esa/population/publications/wpp2006/*.

[102] Anthony Nguyen, Darren Moore, and Iain McCowan. Unsupervised clustering of free-living human activities using ambulatory accelerometry. In *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4895–4898, 2007.

[103] Nam T. Nguyen, Dinh Q. Phung, Svetha Venkatesh, and Hung Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005 (CVPR 2005)*, 2:955–960, 2005.

[104] Feng Niu and Mohamed Abdel-Mottaleb. HMM-based segmentation and recognition of human activities from video sequences. *IEEE International Conference on Multimedia and Expo, 2005 (ICME 2005)*, pages 804–807, 2005.

[105] Paulito Palmes, Hung Keng Pung, Tao Gu, Wenwei Xue, and Shaxun Chen. Object relevance weight pattern mining for activity recognition and segmentation. *Pervasive and Mobile Computing*, 6:43–57, February 2010.

[106] Juha Pärkkä, Luc Cluitmans, and Miikka Ermes. Personalization algorithm for real-time activity recognition using PDA, wireless motion bands, and binary decision tree. *IEEE Transactions on Information Technology in Biomedicine*, 14(5):1211–1215, September 2010.

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

[108] Alex (Sandy) Pentland. Automatic mapping and modeling of human networks. *Physica A: Statistical Mechanics and its Applications*, 378(1):59–67, May 2007.

[109] 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 *AAAI'06: Proceedings of the 21st National Conference on Artificial Intelligence*, pages 906–912, Boston, Massachusetts, 2006. AAAI Press.

[110] Mike Perkowitz, Matthai Philipose, Kenneth Fishkin, and Donald J. Patterson. Mining models of human activities from the web. In *WWW '04: Proceedings of the 13th international Conference on World Wide Web*, pages 573–582, New York, NY, USA, 2004. ACM.

[111] M. Philipose, K.P. Fishkin, M. Perkowitz, D.J. Patterson, D. Fox, H. Kautz, and D. Hahnel. Inferring activities from interactions with objects. *Pervasive Computing, IEEE*, 3(4):50 – 57, oct.-dec. 2004.

[112] Ross J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, March 1986.

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

[114] Parisa Rashidi and Diane J. Cook. Activity recognition based on home to home transfer learning. In *AAAI'10: Proceedings of the 24th conference on Artificial intelligence*, pages 45–52. AAAI press, 2010.

[115] Chotirat (Ann) Ratanamahatana and Dimitrios Gunopulos. Feature selection for the naïve Bayesian classifier using decision trees. *Applied Artificial Intelligence*, 17(5-6):475–487, 2003.

[116] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th conference on Innovative applications of artificial intelligence - Volume 3*, pages 1541–1546. AAAI Press, 2005.

[117] Payam Refaeilzadeh, Lei Tang, and Huan Liu. Cross-validation. In *Encyclopedia of Database Systems*, pages 532–538. Springer US, 2009.

[118] Jason D. M. Rennie. Improving multi-class text classification with naïve Bayes. Master's thesis, Massachusetts Institute of Technology, 2001.

[119] Patrice Roy, Bruno Bouchard, Abdenour Bouzouane, and Sylvain Giroux. A hybrid plan recognition model for Alzheimer's patients: Interleaved-erroneous dilemma. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*, pages 131–137, Washington, DC, USA, 2007. IEEE Computer Society.

[120] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk E-mail. In *Learning for Text Categorization: Papers from the 1998 Workshop*, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[121] Azaruddin Sayyed and Suneeta Agarwal. PPM revisited with new idea on escape probability estimation. In *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA 2007) - Volume 04*, pages 152–156, Washington, DC, USA, 2007. IEEE Computer Society.

[122] Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 625–656, 1948.

[123] Delsey M Sherrill, Marilyn L Moy, John J Reilly, and Paolo Bonato. Using hierarchical clustering methods to classify motor activities of COPD patients from wearable sensor data. *Journal of NeuroEngineering and Rehabilitation*, 2:16–29, 2005.

[124] Daniel L. Silver and Ryan Poirier. Requirements for machine lifelong learning. In *Second International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC 2007)*, volume 4527 of *Lecture Notes in Computer Science*, pages 313–319. Springer, 2007.

[125] Steven Simoens, Mike Villeneuve, and Jeremy Hurst. Tackling nurse shortages in OECD countries. Technical report, OECD Health Working Papers No. 19, 2005.

[126] Geetika Singla, Diane J. Cook, and Maureen Schmitter-Edgecombe. Incorporating temporal reasoning into activity recognition for smart home residents. In

Proceedings of the AAAI Workshop on Spatial and Temporal Reasoning, pages 53–61, 2008.

[127] Joshua R. Smith, Kenneth P. Fishkin, Bing Jiang, Alexander Mamishev, Matthai Philipose, Adam D. Rea, Sumit Roy, and Kishore Sundara-Rajan. RFID-based techniques for human-activity detection. *Communications of the ACM*, 48:39–44, September 2005.

[128] Dina Sokol, Gary Benson, and Justin Tojeira. Tandem repeats over the edit distance. *Bioinformatics*, 23(2):e30–e35, 2006.

[129] Thomas Stiefmeier, Daniel Roggen, Gerhard Tröster, Georg Ogris, and Paul Lukowicz. Wearable activity tracking in car manufacturing. *IEEE Pervasive Computing*, 7(2):42–50, April-June 2008.

[130] Maja Stikic, Tâm Huỳnh, Kristof Van Laerhoven, and Bernt Schiele. ADL recognition based on the combination of RFID and accelerometer sensing. *Second International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth 2008).*, pages 258–263, 2008.

[131] Maja Stikic, Kristof Van Laerhoven, and Bernt Schiele. Exploring semi-supervised and active learning for activity recognition. In *12th IEEE International Symposium on Wearable Computers (ISWC 2008)*, pages 81–88, 2008.

[132] Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *Journal of Machine Learning Research*, 8:693–723, May 2007.

[133] Emmanuel M. Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive*, pages 158–175, 2004.

[134] Moritz Tenorth, Jan Bandouch, and Michael Beetz. The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1089–1096, September–October 2009.

[135] Tim van Kasteren, Gwenn Englebienne, and Ben Kröse. Recognizing activities in multiple contexts using transfer learning. In *AAAI Fall 2008 Symposium: AI in Eldercare*, 2008.

[136] Tim van Kasteren, Gwenn Englebienne, and Ben Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 165–186. Atlantis Press, 2011.

[137] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *UbiComp'08: Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 1–9. ACM, 2008.

[138] Vinay Vishwakarma, Chittaranjan Mandal, and Shamik Sural. Automatic detection of human fall in video. In *Proceedings of the 2nd International Conference on Pattern Recognition and Machine Intelligence (PReMI'07)*, pages 616–623, Berlin, Heidelberg, 2007. Springer-Verlag.

[139] Levenshtein Vladimir. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[140] Shuangquan Wang, Jie Yang, Ningjiang Chen, Xin Chen, and Qinfeng Zhang. Human activity recognition with user-free accelerometers in the sensor networks. In *International Conference on Neural Networks and Brain (ICNNB'05)*, volume 2, pages 1212–1217, October 2005.

[141] Jamie A. Ward, Paul Lukowicz, Gerhard Troster, and Thad E. Starner. Activity recognition of assembly tasks using body-worn microphones and accelerometers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10):1553–1567, Oct. 2006.

[142] Mark Weiser. The computer for the 21st century. *SIGMOBILE Mobile Computing Communications Review*, 3:3–11, July 1999.

[143] Lloyd R. Welch. Hidden Markov models and the Baum-Welch algorithm. *IEEE Information Theory Society Newsletter*, 53(4), December 2003.

[144] Terry A. Welch. A technique for high-performance data compression. *Computer*, 17(6):8–19, 1984.

[145] Wikipedia. *Twenty Questions*. http://en.wikipedia.org/wiki/Twenty_Questions/, 2011.

[146] Jian Kang Wu, Liang Dong, and Wendong Xiao. Real-time physical activity classification and tracking using wearble sensors. In *6th International Conference on Information, Communications Signal Processing*, pages 1–6, December 2007.

[147] Jianxin Wu, Adebola Osuntogun, Tanzeem Choudhury, Matthai Philipose, and James M. Rehg. A scalable approach to activity recognition based on object use. In *IEEE 11th International Conference on Computer Vision (ICCV 2007)*, pages 1–8, 2007.

[148] Yufeng Wu. A practical method for exact computation of subtree prune and regraft distance. *Bioinformatics*, 25:190–196, November 2008.

[149] Danny Wyatt, Matthai Philipose, and Tanzeem Choudhury. Unsupervised activity recognition using automatically mined common sense. In *AAAI'05: Pro-*

ceedings of the 20th National Conference on Artificial Intelligence, pages 21–27. AAAI Press, 2005.

[150] Jaeyoung Yang, Joonwhan Lee, and Joongmin Choi. Activity recognition based on RFID object usage for smart mobile devices. *Journal Computer Science Technology*, 26(2):239–246, March 2011.

[151] G. Michael Youngblood and Diane J. Cook. Data mining for hierarchical model creation. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 37(4):561–572, 2007.

[152] Lei Yu and Huan Liu. Efficient feature selection via analysis of relevance and redundancy. *Journal of Machine Learning Research*, 5:1205–1224, December 2004.

[153] Piero Zappi, Thomas Stiefmeier, Elisabetta Farella, Daniel Roggen, Luca Benini, and Gerhard Tröster. Activity recognition from on-body sensors by classifier fusion: sensor scalability and robustness. In *Proceedings of the 3rd International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP'07)*, pages 281–286, 2007.

[154] Huiru Zheng, Haiying Wang, and Norman D. Black. Human activity detection in smart home environment with self-adaptive neural networks. In *Proceedings of the IEEE International Conference on Networking, Sensing and Control*, pages 1505–1510, Hainan, China, 2008. IEEE.

[155] Vincent Wenchen Zheng, Derek Hao Hu, and Qiang Yang. Cross-domain activity recognition. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, Ubicomp'09, pages 61–70, New York, NY, USA, 2009. ACM.

[156] Gang Zhou and Youfu Wu. Anomalous event detection based on Self-Organizing Map for supermarket monitoring. In *International Conference on Information*

Engineering and Computer Science, 2009 (ICIECS 2009), pages 1–4, December 2009.

[157] Hankz Hankui Zhuo, Qiang Yang, Rong Pan, and Lei Li. Cross-domain action-model acquisition for planning via web search. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS 2011)*, pages 298–305. AAAI, 2011.

[158] Manuel Zini, Marco Fabbri, Massimo Moneglia, and Alessandro Panunzi. Plagiarism detection through multilevel text comparison. In *AXMEDIS '06: Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, pages 181–185, 2006.

[159] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.