ACTIVITY DETECTION AND CLASSIFICATION ON A SMART FLOOR

by

ANIL KUMAR MULLAPUDI

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree, of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2017

Acknowledgements

I would like to thank Dr. Manfred Huber for the continued support during my thesis design and implementation. I appreciate the time he allocated throughout the last two semesters. I would also thank Dr. Gergely V. Zaruba and Mr. David Levine for being my committee members and providing valuable suggestions.

I thank all friends namely Gaurav, Suraj, Suhas, and Harshal and all the other lab mates for support in discussions and implementation. Lastly, I would like to thank my family for the continuous support during my journey in UTA.

April 20, 2017

Abstract

ACTIVITY DETECTION AND CLASSIFICATION ON A SMART FLOOR

Anil Kumar Mullapudi, MS

The University of Texas at Arlington, 2017

Supervising Professor: Manfred Huber

Detecting and analyzing human activities in the home has the potential to improve monitoring of the inhabitants' health especially for elderly people. There are many approaches to detect and categorize human activities that have been applied to data from several devices such as cameras and tactile sensors. However, use of these sensors is not feasible in many places due to security and privacy concerns or because of users who may not be able to attach sensor to their body. Some of these issues can be addressed using less intrusive sensors such as a smart floor. A smart floor setup allows to detect human temporal behaviors without any external sensors attached to users. However, use of such indirect, environmental sensors also changes the character and quality of the data available for activity recognition. In this thesis, an approach to activity detection and classification aimed at smart floor data is developed and evaluated. The approach developed here is applied to data obtained from a pressure-sensor based smart floor and activities of interest include standing, walking, and a miscellaneous class of movement.

The main aim this thesis is to detect and classify human activities from time series data which is collected from pressure sensors. No assumption is made here that the data has been segmented into activities and thus the algorithm must not only determine the type of activity but also has to identify the corresponding region within the data. The activities standing, walking, and other are identified in data obtained from pressure sensors which are mounted under the floor. Various features extracted from these sensors such as center of pressure, speed, and average pressure are used for the detection and classification. To identify activities, a Hidden Markov Model (HMM) is trained using a modified Baum-Welch algorithm that allows for semi-supervised training using a set of labeled activity data as well as a larger set of unlabeled pressure data in which activities have not been previously identified. The goal of being able to classify these activities is to allow for general behavior monitoring and, paired with anomaly detection approaches, to enhance the ability of the system to detect significant changes in behavior to help identify warning signs for health changes in elderly individuals.

Contents

Acknowledgements iii

Abstract iv

List of Illustrations viii

Chapter 1 Introduction 10

1.1 Motivation Behind the Thesis 10

1.2 Challenges 11

Chapter 2 Related Work 12

2.1 Activity detection based on computer vision system. 12

2.2 Activity detection based on external sensors 13

Chapter 3 Technical Background 14

3.1 Discrete Fourier Transformation (DFT): 14

3.2 Hidden Markov Model (HMM): 14

3.3 Forward algorithm: 16

3.4 Backward Algorithm 16

3.5 Viterbi Algorithm 17

3.6 Baum-Welch Algorithm: 17

3.7 Logistic Regression 19

Chapter 4 Smart Floor Setup 20

4.1 Smart Floor 20

4.2 Calibration of Smart Floor 21

Chapter 5 Experiment Overview 22

Chapter 6 Technical Approach 24

6.1 Feature Generation: 24

6.1.1 Center of pressure 25

6.1.2 Speed and Total Pressure 25

6.1.3 Sliding Window and Frequency Component Generation 25

6.1.5 Feature Dimension Reduction Using Logistic Regression 27

6.2 Manual Labeling: 28

6.2.1. Standing: 28

6.2.2 Walking: 29

6.2.3 class of movement 30

6.4 HMM Classifier 31

6.4.1 HMM Classifier based on Heuristic Approach (Model1) 32

6.4.2 HMM Classifier based on biased expectation Approach1 (Model 2) 34

6.4.3 HMM Classifier based on biased expectation Approach2 (Model 3) 37

Chapter 7 Experiments and Results 40

7.1 Model validation 40

7.2 Model1 experiments and results 41

7.3 Model2 experiments and results 42

7.4 Model3 experiments and results: 44

Chapter 8 Conclusion and Future work 47

References 48

Biographical Information 50

List of Illustrations

Figure 3-1: Hidden Markov Model…………………………………..15

Figure 4-1: Smart Floor Setup [3] 20

Figure 4-2 Tekscan FlexiForce A401 Sensor 20

Figure 5-1 User COP while moving on a smart floor [3] 22

Figure 6-1 Classification Approach 24

Figure 6-2 Sliding window 26

Figure 6-3 Frequency Component Generation 27

Figure 6-4 Plotting center of pressure sliding window of length 25, when user is standing 29

Figure 6-5 COP of walking window 30

Figure 6-6 COP pattern of movement class..31

Figure 6-7 HMM with three states 33

Figure 6-8 HMM with a single Gaussian for each state 33

Figure 6-9 HMM with a biased expectation approach 35

Figure 6-10 HMM Transitions when observations are changed from class1 to class2 36

Figure 6-11 HMM Transitions when observations are changed from class2 to class3 37

List of Tables

Table 6-1 Classification Accuracy Using Logistic Regression ... 28

Table 6-2 State to Label Mapping ... 34

Table 6-3 Forward probability for the observations [1,1,1,1,1] ... 38

Table 6-4 Forward probability for the observations [1,3,3,3,2] ... 39

Table 7-1 Model 1 testing results with random initialization ... 41

Table 7-2 Model1 experiment results with 3states for 30 five cross-fold validations ... 42

Table 7-3 Model 2 testing results with random initialization ... 42

Table 7-4 Model2 experiment results with 3states for 30 five cross-fold validations ... 43

Table 7-5 Model2 experiment results with 6states for 30 five cross-fold validations ... 43

Table 7-6 Model 3 testing results with random initialization ... 44

Table 7-7 Model3 experiment results with 3states for 30 five cross-fold validations ... 44

Table 7-8 Model3 experiment results with 6states for 30 five cross-fold validations ... 45

Table 7-9 Semi-Supervised Learning Results ... 46

Chapter 1
Introduction

Identifying human activities has become increasingly popular in health care monitoring to provide support for assisted living especially for elderly people. A major portion of the elderly population is suffering from age-related conditions such as Parkinson's disease, diabetes, cardiovascular disease, Alzheimer's disease, different chronic diseases and limitations in physical function [1]. To be able to manage these conditions efficiently in home settings, there is a significant need to monitor health continuously, leading to reduced risks and thus to improved quality of life. Smart home technologies are one of the methodologies to provide useful data to identify human activities in everyday settings. The data for this project is collected from a custom designed infrastructure built into an apartment in the context of previous work [2],[3]. IN these papers, A brief overview and setup of this smart floor is provided in the related work section. For more detail, Oluwatosin et al. [3] provide a more detailed description of the hardware design and architecture of the smart floor infrastructure (2016).

1.1 Motivation Behind the Thesis

The motivation behind the thesis was to provide support for the monitoring of people's health by identifying activities. This can be achieved by identifying what they are doing over time. Therefore, we need to analyze time series data and apply machine learning algorithms (generative models) such as Hidden Markov Models or other time series based algorithms to form recognition models that can then be used to improve health monitoring. To perform such monitoring efficiently, it is important that the actual technology does not directly influence the activities of the people. As a result, it is desirable to be able to perform activity recognition based on sensors that can capture common activities continuously and in a transparent and non-intrusive fashion. To address this, the smart floor used here provides a technology that captures data without the need for any particular interaction of the user (as would be the case with person-mounted sensors). Moreover, the smart floor provides a means of monitoring that is largely invisible to the user and thus minimizes the "white coat effect" where the knowledge of the presence of technology or measurements itself changes the way persons act and thus the data itself, potentially invalidating previous models.

Using data from these sensors comes with a number of issues that have to be addressed in the choice of activity detection methodology. In particular, the data only contains information regarding pressure patters and no information regarding direct measurements of body posture or objects a person interacts with. As a result, the approach chosen has to be able to built models which can extract relevant information for each possible activity from such raw data. To do this, this thesis uses supervised, unsupervised, and semi-supervised learning techniques centered around Hidden Markov Models to build a activity detection and recognition framework that can operate on such data.

Chapter 2
Related Work

Activity detection can be performed using several approaches. The most common approaches use either computer vision or attach external sensor to the human body. The data generated in these approaches are generally time series data that has then to be interpreted. The following two sections describe some of the existing activity detection methodologies.

2.1 Activity detection based on computer vision system.

One of the general methodologies used for activity recognition is video surveillance. Activities such as jogging, running, walking and other activities can be detected using 3D Convolutional Neural Networks [4]. Activities can also be detected by identifying the human in video frame sequences based the position and velocity parameters [5]. Besides these, there are a significant number of other approaches which are based on the computer vision methodologies. However, these techniques are difficult to implement in home environment due to privacy reasons. Also, computer vision based approaches often have difficulties with occlusion of parts of the body which is very common in home environments where furniture and other objects often only allow partial views of the person. Similarly, many recognition approaches require correct body segmentation around the legs which is difficult in the context of dresses or nightgowns which obscure the shape of the body and thus make the analysis of gait and walking activities more difficult.

2.2 Activity detection based on external sensors

Activity detection can be performed using external sensors attached to the human body. For example, sensors such as accelerometers, gyroscopes and bend sensors can be used to extract data related to the human body to detect human activities [6]. Maurer et al., (2006) extracted features from sensors using a windowing technique, and applied k-Nearest Neighbor and Naive-Bayes classifiers. As one of the most commonly used sensors, accelerometer sensors can be attached to different places of the body, and from these sensors, activities are detected by extracting features like mean, energy, or frequency-domain entropy [7]. In their paper, Bao et al. used Naive Bayes and Decision Table classifiers to detect activities such as walking, sitting, running and other similar activities. As opposed to these works, the approach in this thesis is built around a sensor system that is embedded in the environment in the form of a smart floor rather than carried with the person, posing somewhat different challenges.

Chapter 3
Technical Background

The input for activity detection and classification in this thesis work is time series data which is collected from a smart floor. This time series data is then converted into the frequency-domain to make the data independent of the floor location and thus allow the same detection and identification model to be used on every part of the floor. After the frequency component generation, Logistic regression and Hidden Markov Models are used to classify activities. The rest of this section provides the background of the methodologies used in the implementation.

3.1 Discrete Fourier Transformation (DFT):

Any time series signals can be represented in terms of sinusoidal waveforms [8], effectively translating the signal into its frequency components. DFT is a tool that efficiently separates the time series signals into these frequency components, providing information regarding the strength (or energy) of each of the components. This, in turn, allows to analyze the same signal independent of the spatial parameters purely in terms of its frequencies, thus representing a complete time series as a set of static parameters. In this thesis, the MATLAB “fftn” function is used to obtain the frequency components of a time series signals.

3.2 Hidden Markov Model (HMM):

HMM's are one of most commonly used models for time series data analysis. The input to an HMM is a sequence of observations ($O_1, O_2, \ldots O_M$). These observations can be collected from any sensor. From this sequence of observations, a HMM can evaluate the likelihood of the observation sequence (and thus its match to the model).

Moreover, it can also predict the most likely sequence of internal (hidden) state parameters and with this allows to infer the context in which the observations were made. In this chapter, the details of the HMM model framework are briefly explained. Rabiner [9] illustrated the capabilities of HMMs with well-defined speech recognition examples. Figure 3-1 shows the basic HMM model structure.

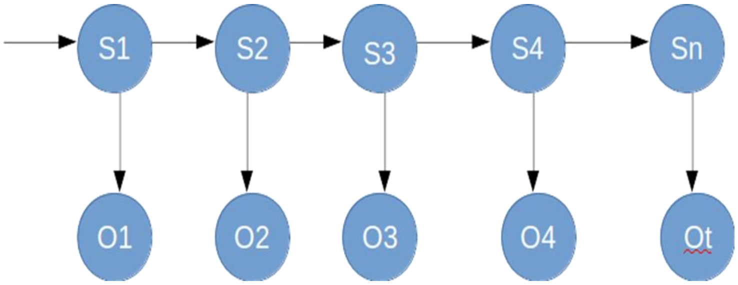


Figure 3-1Hidden Markov Model

In Figure 3-1, S1,.,.Sn represents the hidden states, O1,...Ot represents the observations. Every state is dependent only on its immediate predecessor state and independent of all other previous states. This assumption is called the Markov assumption [9]. The transition probabilities between the states are represented in a transition matrix (A), and the observation probabilities given the states are represented by an observation matrix (B). Every run of the model (i.e. beginning of an observation sequence) starts in a state according to an initial probability which is represented by (Π). The overall model is then represented with $\lambda((A,B,\pi))$. An HMM basically solves three fundamental problems [9]. They are 1) finding the probability of the observation sequence given the model 2) finding the most likely state sequence corresponding with an observation sequence given the and 3) estimation of the best model parameters A, B, Π given a set of observation sequences. The algorithms for these three problems are briefly explained in the next sections.

3.3 Forward algorithm:

The forward algorithm (HMM Problem1 Solution) is used to find the probability of an observation sequence given a model [9]. The steps in this algorithm are shown below [9].

Definition: $\alpha_t(i) = P(O_1\ O_2\ O_{3\ldots}O_T, q_t = S_t \mid \lambda)$

1) Initialization $\quad \alpha_t(i) = \pi_i b_i(\mathrm{O}_1) \quad for\ i \leq N$

2) Induction $\quad \alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\alpha_{ij}\right] b_j(\mathrm{O}_{t+1}) \quad 1 \leq t \leq T-1 \quad 1 \leq j \leq N$

3) Termination $\quad P(\mathrm{O}|\lambda) = \sum_i^N \alpha_T(i)$

Where:

π_i represents the initial probability

$b_j(\mathrm{O})$ represent the observation probability

a_{ij} represents the probability of the transition from state i to state j

λ represents the model

S_t represent State S at time t

3.4 Backward Algorithm

The backward algorithm is also used to find the probability of an observation sequence for a given model but operates in the reverse direction, starting with the observation from time t+1 [9].

Definition: $\beta_t(t) = P(O_{t+1}, O_{t+2}\ldots O_T | q_t = S_i, \lambda)$

1) $\beta_T(i) = 1, 1 \leq i < N$

2) $\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1})\beta_{t+1}(j)\ \ t = T-1, T-2, \ldots.1, 1 \leq i \leq N$

3.5 Viterbi Algorithm

This algorithm is used to find the most probable state sequence for a given observation sequence and is also known as the decoding problem [9]-[10].

Definition: $\delta(i) = max_{q_1,q_2,..,q_{t-1}} P\left[q1\; q2 \ldots q_t = i, O_{1,} O_{2,} \ldots O_t \middle| \lambda\right]$

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

where $\delta(j)$is the maximum probability path at the time t

2) $P^* = max_{1<i<n}[\delta_T(i)]$

The above two steps will find the maximum probability at every time step from all possible paths. After these 2 steps, the results are propagated backwards to trace the optimal state sequence path.

3.6 Baum-Welch Algorithm:

The Baum-Welch algorithm (HMM Problem 3) is one of the most common algorithms used to estimate the model parameters $\lambda = ((A, B, \pi))$ [9]. As the observation values in HMMs can be either discrete or continuous, slight variations in the algorithm exist. If the observations are discrete random variables, then every state has a probability of emitting that observation for all possible values of that discrete random variable. If the observations are continuous, then the observation probability for a given state is derived from the probability density function (pdf). As the data in this thesis is continuous, the description here will focus on this case. Here, the following multivariate Gaussian pdf function is used to represent the observation probability using mean and covariance values of [9].

$$b_j(O) = \sum_{m=1}^{M} C_{\{jm\}}\; \aleph\left[O, \mu_{jm}\,, U_{jm} \quad 1 \le j \le N\right]$$

Here $b_j(O)$ is the observation probability at state j, M is the number of Gaussian mixtures, μ_{jm}, U_{jm} are the mean vector and covariance matrix at state j with a m^th^ Gaussian mixture. In this thesis work, we used only one mixture, i.e. a single Gaussian distribution, at each state. The Baum-Welch algorithm primarily consists of two steps. The first step is the Expectation step and the Second step is the Maximization steps. Both are shown below

Expectation Step [9] :

a) $\alpha_{t+1}(j)$ forward probability for each state

b) $\beta_t(i)$ backward probability for each state

c) $\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\, a_{ij}\, b_j(O_{t+1})\, \beta_{t+1}(j)}$ probability of transitioning from i to j at time t

d) $\gamma_t(i) = \sum_{j=1}^{N}\xi\,(i,j)$ probability of being in state i at time t

Maximization Step [9]:

a) $\grave{\pi}_\imath = \gamma_1(i)$ initial probability

b) $\grave{a}_{\imath\jmath} = \frac{\text{expected number of transitions from state}S_i\text{to state}S_j}{\text{expected number of transitions from stae}S_j}$ Transition probability

$$= \frac{\sum_{t=1}^{T-1}\xi_t\,(i,j)}{\sum_{t=1}^{T}\gamma\,(i)}$$

c) $\mu_j = \frac{\sum_{t=1}^{T}\gamma_t(j)\, o_t}{\sum_{t=1}^{T}\gamma_t(j)}$ Estimated mean

d) $U_j = \frac{\sum_{t=1}^{T}\gamma_t(j)(o_t-\mu_j)(o_t-\mu_j)'}{\sum_{t=1}^{T}\gamma_t(j)}$ Estimated covariance

Here the mean and covariance are used to compute the observation probability of a multi-variate Gaussian.

3.7 Logistic Regression

Logistic regression is a linear binary classifier used to solve classification problems. If there are K classes, then K-binary logistic classifiers are required for training, where the kth class is trained with positive examples and all the remaining classes are trained as negative examples [11]. This is also known as one vs all classification approach. In this thesis work, scikit-learn logistic regression library is used to obtain the probability of the class given the observation data[12].

Chapter 4

Smart Floor Setup

4.1 Smart Floor

When the user is on the on a smart floor, the raw pressure sensor data is collected at 25Hz. In this thesis work, the data is collected from the already built in the smart floor [3]. The smart floor used to collect the data is shown in Figure 4-1.

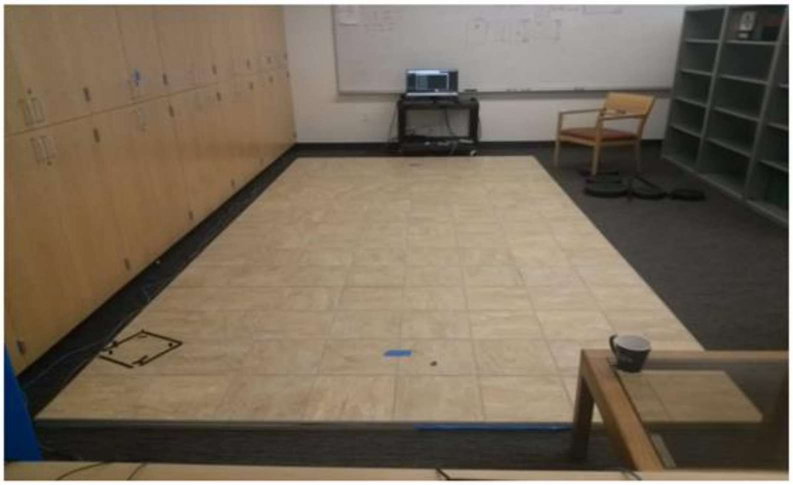


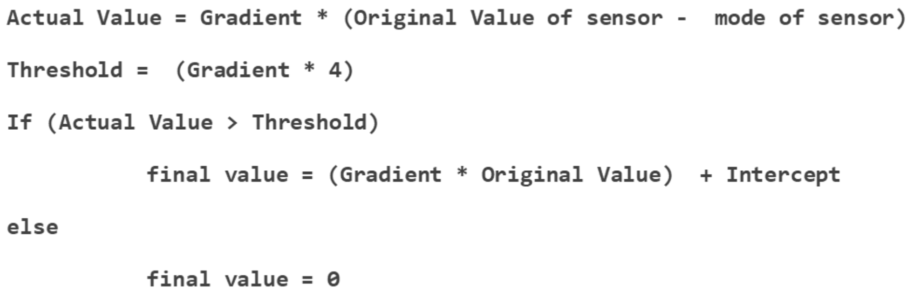
Figure 4-1: Smart Floor Setup [3]

Each tile has an underlying pressure sensor as shown in Figure 4-2. For parameter extraction, the entire floor is divided is into four panels and while collecting the data, all four-panels' readings are transmitted every 40 milliseconds. The detailed description of hardware used to setup the smart floor is explained in [3].

Figure 4-2 Tekscan FlexiForce A401 Sensor

4.2 Calibration of Smart Floor

Even if there is no person walking on the smart floor, a sensor will emit readings due to noise in the sensors. Therefore, to eliminate the noise in the unloaded sensors, Oluwatosin et al., represented each sensor in a slope-intercept form. The slope and intercepts are calculated by placing a set of standard weights on the floor [3]. The actual, calibrated value for the sensor can then be derived using these parameters and further filtered to remove noise-induced readings on remote portions of the floor to eliminate outliers when extracting such location-related features as the center of pressure. The algorithms used for reconstruction and filtering used here is shown below:



```
Actual Value = Gradient * (Original Value of sensor -  mode of sensor)
Threshold =  (Gradient * 4)
If (Actual Value > Threshold)
          final value = (Gradient * Original Value)  + Intercept
else
          final value = 0
```

Chapter 5
Experiment Overview

As discussed above, the smart floor data is collected from the hardware and experiments built and conducted in previous work where the users are asked to perform a series of repetitive activities [3]. All these activities are recorded with a Kinect camera to provide ground truth data and to aid in manual segmentation of the data (as humans are better at interpreting visual data as compared to pressure value arrays. The activities performed in these experiments include standing, walking, slow walking, turning, opening a door, closing a door, holding a cup, placing a cup on a table and other actives. Figure 5-1 shows the center of pressure (COP) of a subject while he is moving on the smart floor during the experiment.

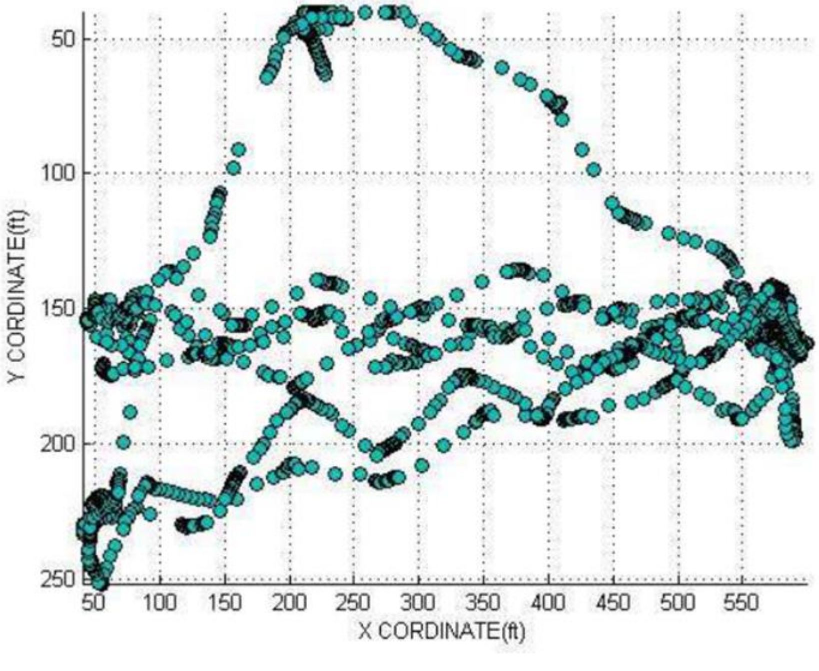


Figure 5-1 User COP While Moving on a Smart Floor [3]

In this thesis work, the COP and average pressure are used to detect and classify the three activities standing, walking and other class of movements. In Figure 5-1, the COP pattern is not a straight line, because the COP is switching from left leg to the right leg. The technical approach to classifying these activities is explained in the next section.

Chapter 6
Technical Approach

Figure 6-1 shows the complete overview of the activity detection and classification approach developed in this thesis. The data collection and calibration are explained in the previous sections and the remaining sections of Figure 6-1 are used to detect and classify the activities using HMM models.

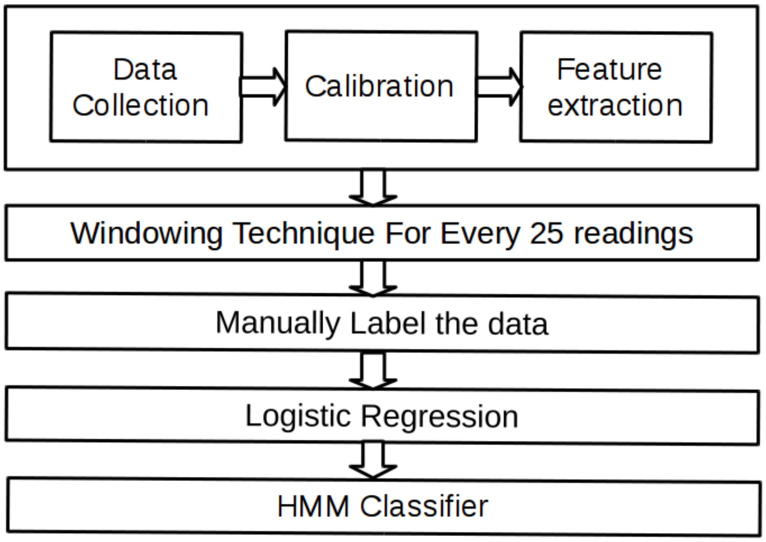


Figure 6-1 Classification Approach

6.1 Feature Generation:

In this work, the three main features used for activity detection and classification are center of pressure, total pressure, and COP speed. The center of pressure and total pressure components are converted into frequency components to make them location independent so they can be used in the HMM without the need for state information in the model. To reduce the dimensions of these generated frequency components to an observation space that is tractable in the context of HMM learning and that makes the multi-variate Gaussian assumption for the observation probability more realistic, logistic regression is used to generate a new observation space in terms of the classification probabilities of the different activity types. The rest of the sections explained the details of each feature and formulation of the frequency components.

6.1.1 Center of pressure

The center of pressure (COP) is calculated to extract the presence of the user on the smart floor and to integrate the pressure distribution of the floor sensors. The COP is a coordinate of user location at time t. The following equation is used to compute the COP [3].

$$\mathrm{COP(x)} = \frac{\sum_i x_i \,.\, F_i}{\sum_i F_i}$$

$$\mathrm{COP(y)} = \frac{\sum_i y_i \,.\, F_i}{\sum_i F_i}$$

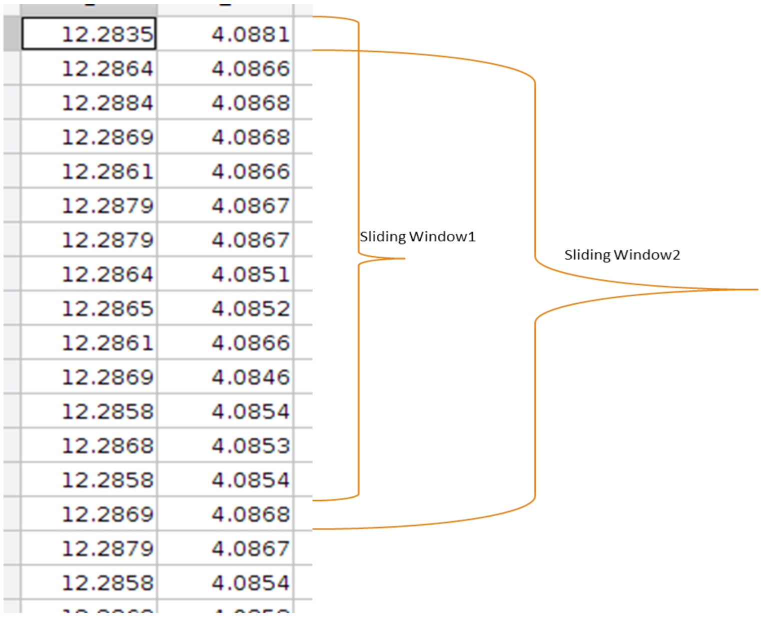
Here, F_i is the pressure output and x_i is the location of the pressure sensor from the reference point in the x-direction.

6.1.2 Speed and Total Pressure

The speed is computed from the COP's (x,y) coordinates. The area on the floor surrounding the user will show high-pressure values and all the other sensors which are far from the user will be zero due to the noise filtering of the calibration. The total pressure is calculated by summing all the non-zero pressure values at time t.

6.1.3 Sliding Window and Frequency Component Generation

The extracted features COP and average pressure are windowed for a length of 25-time steps as shown in Figure 6-2. The reason to choose a window size of 25 is that this corresponds to roughly one second of data and thus is sufficient to capture COP points for at least one step.



12.2835	4.0881
12.2864	4.0866
12.2884	4.0868
12.2869	4.0868
12.2861	4.0866
12.2879	4.0867
12.2879	4.0867
12.2864	4.0851
12.2865	4.0852
12.2861	4.0866
12.2869	4.0846
12.2858	4.0854
12.2868	4.0853
12.2858	4.0854
12.2869	4.0868
12.2879	4.0867
12.2858	4.0854

Figure 6-2 Sliding Window

Once the data is windowed, the feature data corresponding to a window is translated into frequency components using a multidimensional fast Fourier transform(FFT) algorithm. MATLAB “fftn” function is used to get the FFT frequency components. In each window, along with the COP (x,y) coordinates, frequency components are also calculated for the total pressure feature. Thus, with the three-dimensional features for a length of 25 steps as the input data matrix of size (25x3), the corresponding FFT component matrix is of also of size (25x3) (as only the amplitude parameters are used and the phase is ignored). This (25x3) frequency component matrix is flattened into a single row with a length of 75. After this, the speed is added to these features. Here, the speed is the difference between the (x,y) coordinates of the start and end positions of each window. Therefore, every sliding window of 25x3 location-specific input features is translated into a single data point in the form of a 1x76 row vector which is position independent. Also, while the original feature window is a time series, each of the generated 1x76 row vectors are a static parameter representation of the entire window. The brief overview of frequency component generation is shown in Figure 6-3.

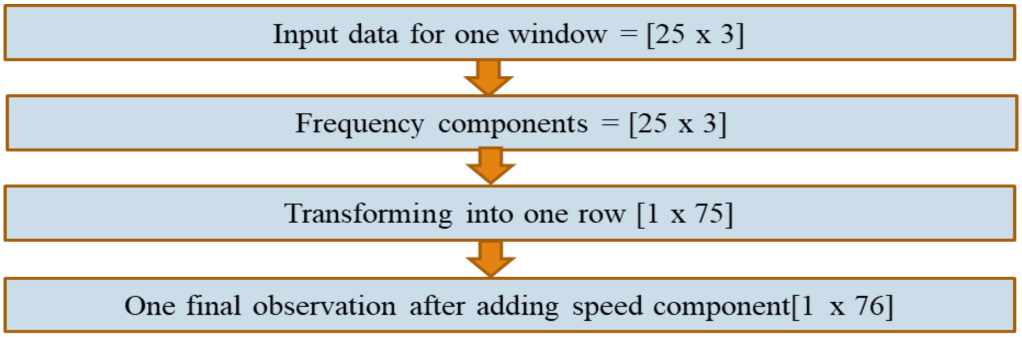

Input data for one window = [25 x 3]
Frequency components = [25 x 3]
Transforming into one row [1 x 75]
One final observation after adding speed component[1 x 76]

Figure 6-3 Frequency Component Generation

6.1.5 Feature Dimension Reduction Using Logistic Regression

Since the generated feature components in the above section are of 76 dimensions, it is difficult to build a HMM model, because the state space is high in HMM for one observation with 76 features, and the sizes of mean and covariance matrices are high. Moreover, it is unlikely that in this 76 dimensional observation space the assumption that the observation probabilities follow a Gaussian is realistic. Therefore, the 76 dimensions are reduced to three dimensions with the help of Logistic Regression. In this work, a one vs all logistic classifier is used to compute the probability of each class for the input observation of length 76. Since, we have three classes, 3 probability values for each class are generated using the logistic regression model and these three probability values are normalized to maintain the sum of all class probabilities as one. Along with the dimensionality reduction, the logistic regression model is validated for all class labels to verify whether the data is distinguishable or not. The accuracy results of logistic regression for the five subjects are shown below table Table 6-1.

Users	Classification Accuracy Using Logistic Regression
Subject1	80.23
Subject2	91.11
Subject3	83.96
Subject4	89.52
Subject5	89.53

Table 6-1 Classification Accuracy Using Logistic Regression

It is important to note here that the Logistic Regression is based only on one second of data and can thus not be expected to be able to identify and classify complex behaviors with high precision, in particular in the presence of activities which share parts. As a consequence, the regression results derived on a sliding window are used here as the observation sequence for a HMM that can capture longer-term patterns needed to distinguish activities.

6.2 Manual Labeling:

To train the regression model and evaluate its performance, labeled data is needed. To obtain this, the labels are derived manually by visualizing each window of length 25. For each of the activities of standing, walking, and other class of movements in he experimental data described previously, manual labels are assigned based on the observed COP trajectory and the available visual data.

6.2.1. Standing:

Figure 6-4 shows a representative pattern of the COP when a subject is standing. In this figure, all COP points can be viewed as a single group, approximately in the same location. It means, there is no significant difference in change of COP when the user is standing on the smart floor. Therefore, this kind of window is labeled as Standing.

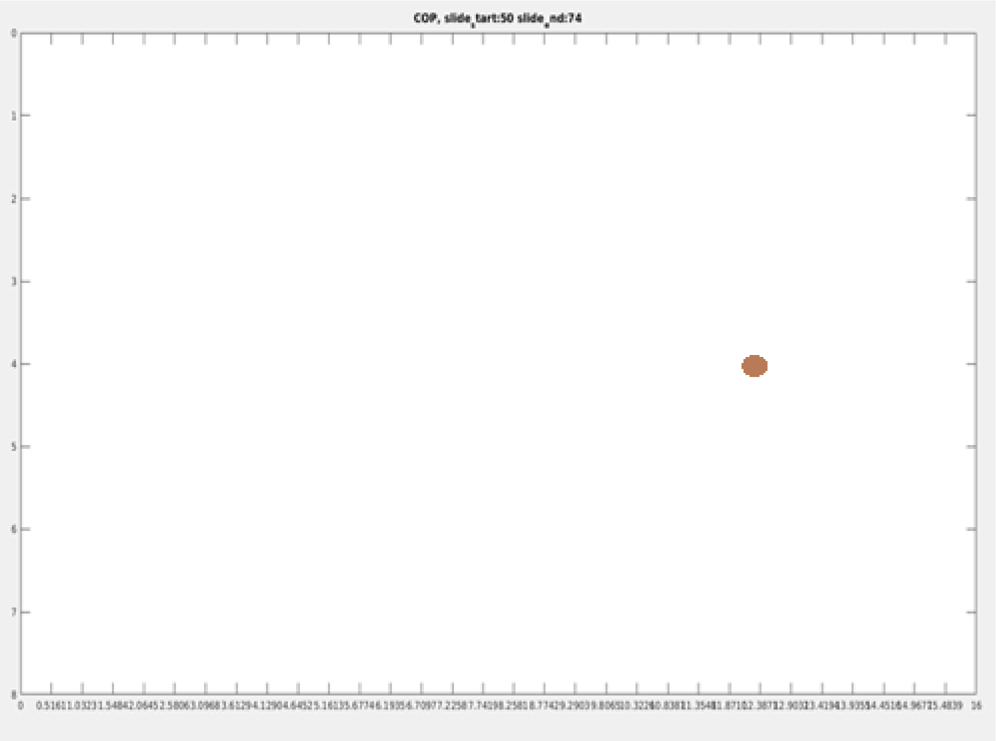


Figure 6-4 Center of Pressure Window of Length 25, When User is Standing

6.2.2 Walking:

Figure 6-5 shows a representative pattern of the COP when a subject is walking on the smart floor. In Figure 6-5, all COP points are spread for a length correspond to a gait cycle. There is a significant difference in the location of the COP when the user is on the smart floor. Hence, the window is labeled as walking.

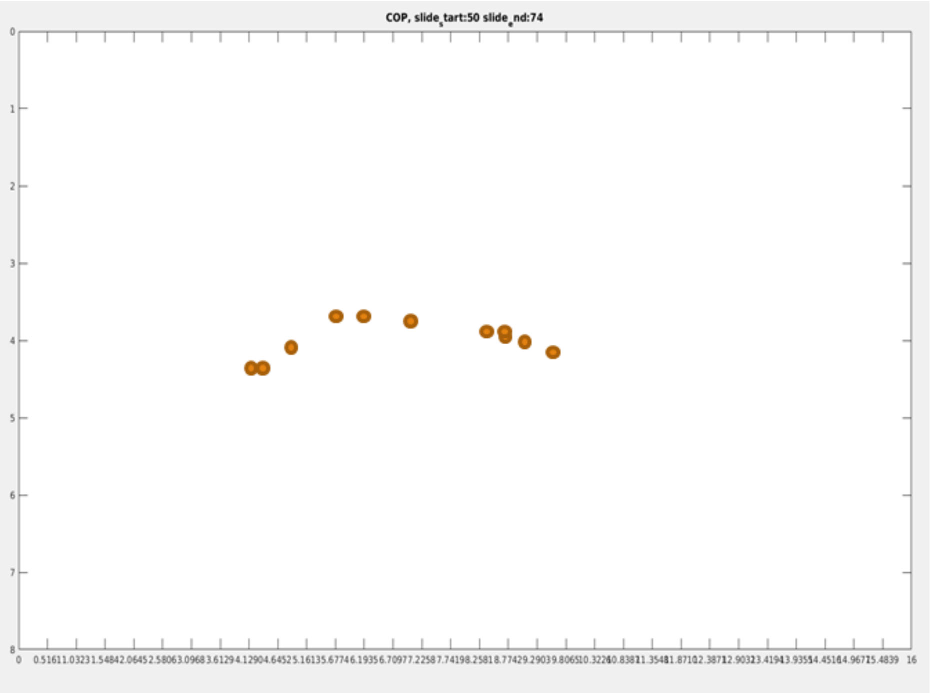


Figure 6-5 COP of Walking Window

6.2.3 Miscellaneous class of movement

Figure 6-6 shows a representative pattern of the COP when the subject is performing some movements other than standing and walking. In Figure 6-6, all COP points are spread in a limited area of the floor and not arching along a walking trajectory. This corresponds to situations such as when the user is turning or doing small feet movements. This pattern of window is labeled as other movement class.

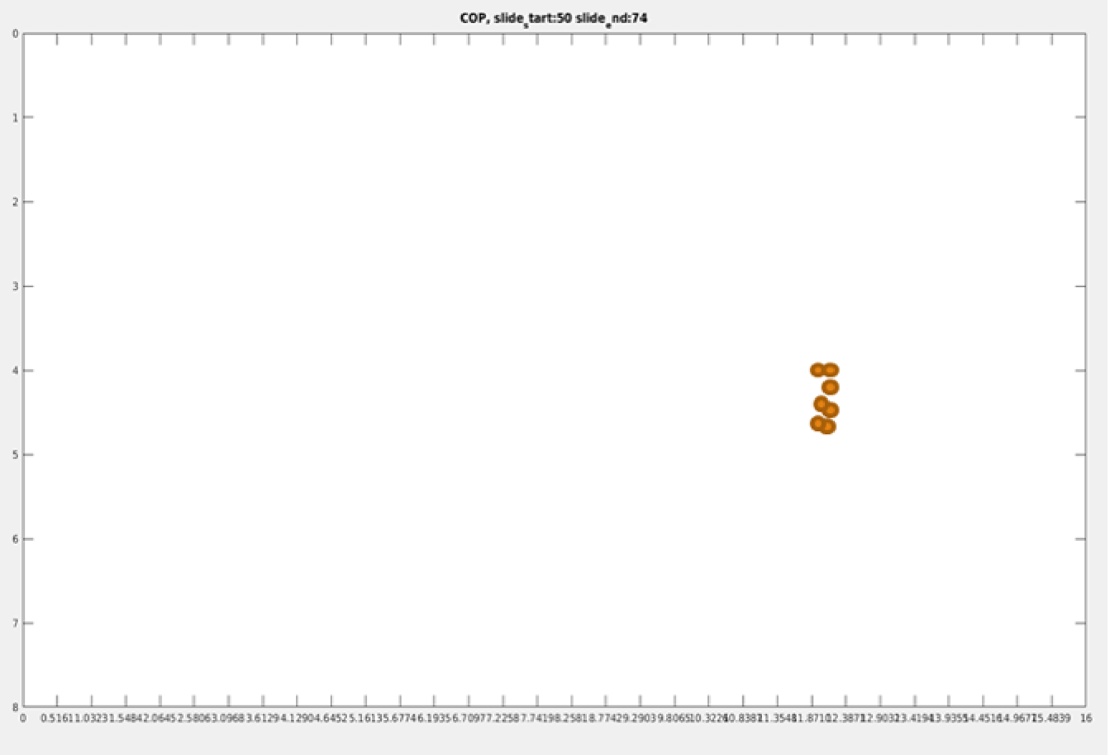


Figure 6-6 COP pattern of movement class

6.4 HMM Classifier

The regression discussed above provides the observations for the HMM model used here. The goal of this model is to capture longer-term relations in activities and to more precisely capture the points where transitions between activities occur. To build the HMM we present variations of the Baum Welch algorithm for partially and semi-supervised applications.

If we train the HMM with the regular Baum-Welch algorithm, we cannot encode the knowledge of class labels in the hidden states. Since, however, the proposed HMM in this thesis work is to provide the class label specific knowledge, it has to be embedded into the the hidden states. Therefore, every state has a meaning and based on the most likely state sequence it is possible to detect and classify activities. The following three HMM-based approaches were developed to classify the activities:

1. HMM Classifier based on Heuristic Approach (Model1)
2. HMM Classifier based on biased expectation Approach1 (Model2)
3. HMM Classifier based on biased expectation Approach2 (Model3)

6.4.1 HMM Classifier based on Heuristic Approach (Model1)

In this approach, the data is trained with a regular Baum-Welch algorithm without any modification to it. Since this does result in a model that does not affiliate activity labels with states, a heuristic has been developed for the label prediction,. To illustrate this heuristic approach, a simple example with three activities and a model with three hidden states is shown in Figure 6-7 and Figure 6-8. In Figure 6-7, each hidden state is assumed to represent one class of activity (indicated in the state) but the label itself is not known during HMM construction. Based on observations sequences from the smart floor, the states will transition from one state to another state with a certain probability. To learn the model parameters for the HMM applied to the data, which is shown in Figure 6-7, the Baum-Welch algorithm is used. This Baum-Welch algorithm learns the model parameters in two steps such as expectation and maximization steps as explained in the technical background, these steps will be repeated for several iterations until the maximum likelihood of the model reaches to convergence.

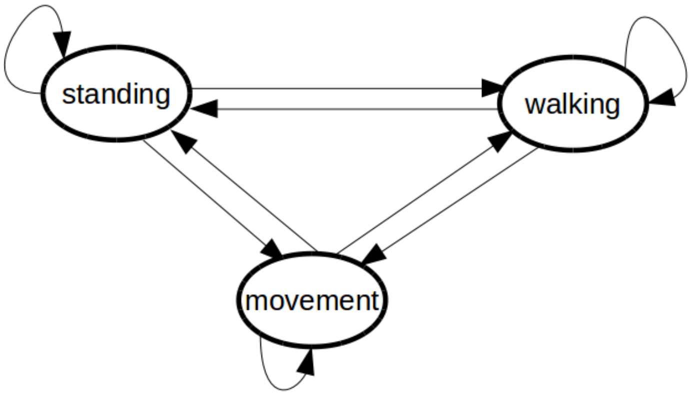


Figure 6-7 HMM with three states

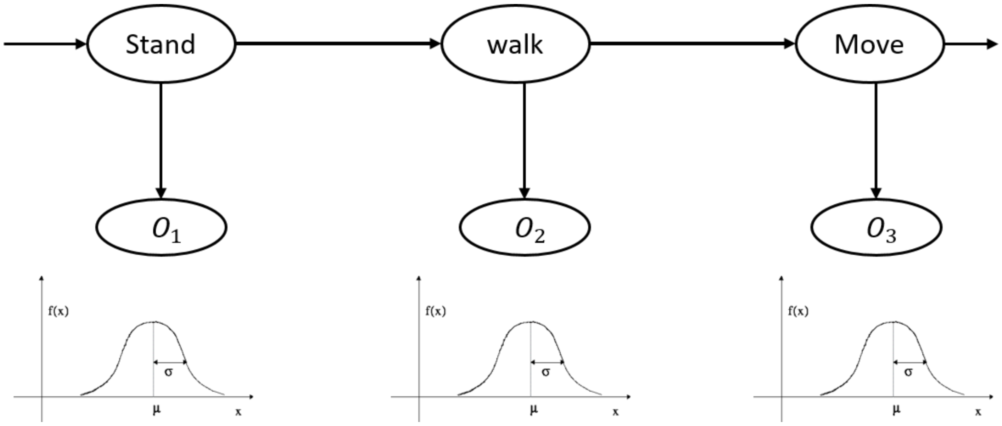


Figure 6-8 HMM with a single Gaussian for each state

In the heuristic approach, after every iteration of Baum-Welch training, the accuracy of classification is tested by using the Viterbi decoding algorithm for every possible combination of state to label mappings. For example, after the first iteration, the output of Viterbi decoding generates a state sequence S_1, S_2,S_3, The accuracy of classification is derived by comparing the state sequence with all possible labels. The following combinations of state to label mappings are possible for the state sequence S_1, S_2, S_3.

Standing Label	Walking Label	Movement Label
S_1	S_2	S_3
S_1	S_3	S_2
S_2	S_1	S_3
S_2	S_3	S_1
S_3	S_1	S_2
S_3	S_2	S_1

Table 6-2 State to Label Mapping

From Table 6-2, the state to label mapping that results in the highest accuracy is chosen as the best representation of hidden state labels. Although this highest accuracy combination will change initially every few iterations, it is found that the state to label mapping combination tends to converge after a certain number of Baum-Welch iterations. Therefore, it is possible using this heuristic to classify activities based on this highest accuracy state to label mapping combination.

While this is a simple extension that often work, the Model 1 approach does not guarantee that it will converge to a constant state to label mapping all the time. Moreover, it is also not feasible to implement this approach if the number of states is increasing because the number of state to label mapping combinations will increase exponentially with the increase in the number of states. These problems are addressed in the Model2 and Model 3 approaches which are explained in the next sections.

6.4.2 HMM Classifier based on biased expectation Approach1 (Model 2)

In the Model2 approach, the model is trained with a modified Baum-welch algorithm. The main idea in the Model2 approach is to annotate the hidden states with the class labels before the training and then forcing the distributions of supervised observation sequences into the annotated hidden states during training. This process will augment the expectation of hidden states during the expectation phase of the Baum-Welch algorithm with the match to the corresponding class label in the observation sequences, effectively reducing the expectation of the state to 0 if it does not match the correct label. This biased expectation approach is illustrated in a simple example shown in Figure 6-9 that shows the hidden state transitions and the observation sequences for the three class labels. In the Model2 approach, each class can be represented with one hidden state or multiple hidden states. In Figure 6-9, Class1 is represented with the hidden states $S_{1,}S_{2,\dots,}S_5$; Class2 is represented with the states $S_{6,}S_{7,\dots,}S_{10}$; Class3 is represented with $S_{11,}S_{12,\dots,}S_{15}$. In this example, if the input observation belongs to class1 during time t, then the modified Baum-Welch algorithm will make the expectation of reaching the hidden states $S_{6,}\ S_{7,\dots,}\ S_{15}$ zero at time t. Since states $S_{6,}\ S_{7,\dots,}\ S_{15}$ do not belong to class 1, Model2 is biasing those observations to reach only class1 hidden states $S_{1,}\ S_{2,\dots,}\ S_5$.

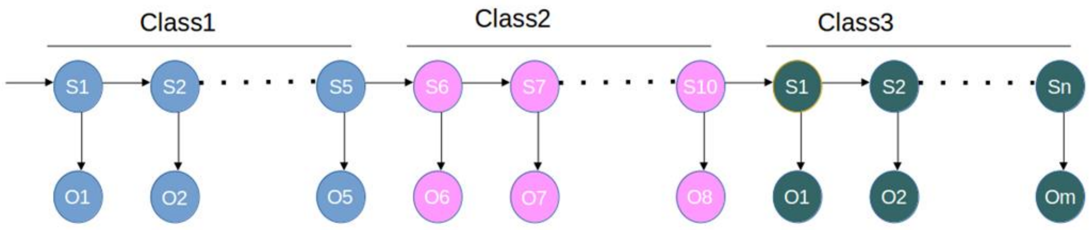
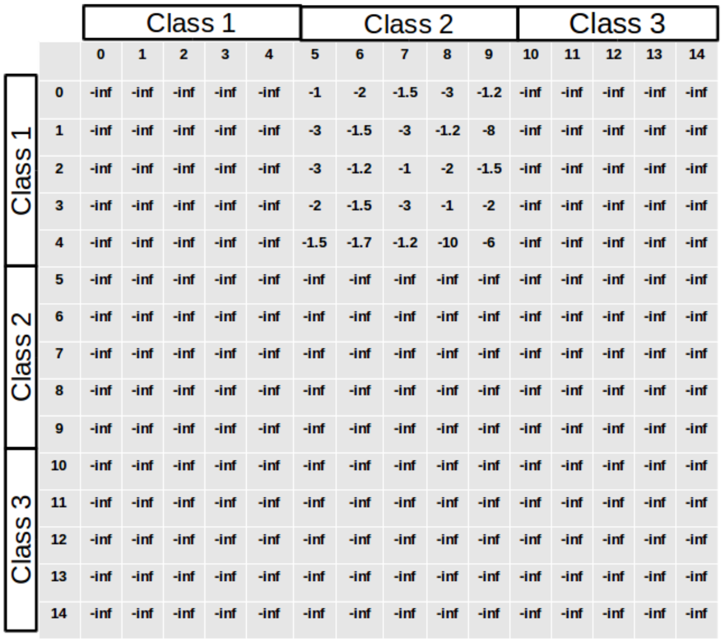


Figure 6-9 HMM with a biased expectation approach

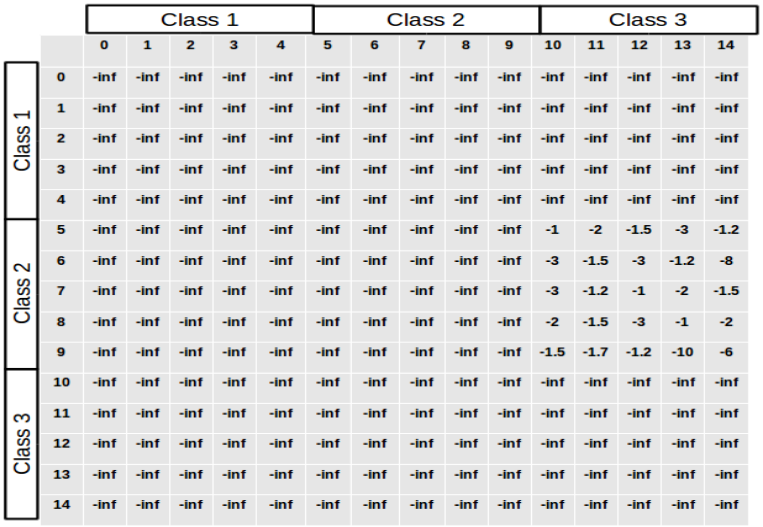
The following explains how the biased expectation is achieved through the Baum-Welch algorithm in more detail. As discussed in the technical background, in the Baum-Welch algorithm exception step, the third step is to calculate the probability of transitioning from state i to j during time t. This is represented with $\xi_t(i,j)$, and this step is modified in the Model2 approach. In the Model2 approach, if the i^{th} state belongs to class1 and the j^{th} state belongs to class 2, which is consistent with the labels on the corresponding observations in the observation sequence, then the remaining class probabilities are set to zero as shown in Figure 6-10. Here, the zero probabilities are represented with “-inf" since all the computations in Model2 are implemented in terms of log likelihoods.



		Class 1					Class 2					Class 3				
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Class 1	0	-inf	-inf	-inf	-inf	-inf	-1	-2	-1.5	-3	-1.2	-inf	-inf	-inf	-inf	-inf
	1	-inf	-inf	-inf	-inf	-inf	-3	-1.5	-3	-1.2	-8	-inf	-inf	-inf	-inf	-inf
	2	-inf	-inf	-inf	-inf	-inf	-3	-1.2	-1	-2	-1.5	-inf	-inf	-inf	-inf	-inf
	3	-inf	-inf	-inf	-inf	-inf	-2	-1.5	-3	-1	-2	-inf	-inf	-inf	-inf	-inf
	4	-inf	-inf	-inf	-inf	-inf	-1.5	-1.7	-1.2	-10	-6	-inf	-inf	-inf	-inf	-inf
Class 2	5	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	6	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	7	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	8	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	9	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
Class 3	10	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	11	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	12	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	13	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	14	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf

Figure 6-10 HMM Transitions When Observations are Changed from Class1 to Class2

Similarly, if the i^{th} observation belongs to class2 and the j^{th} observation belongs to class3, the biased expectation from state i to state j are high and all other probabilities are set to zero as shown in Figure 6-11.



		Class 1					Class 2					Class 3				
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Class 1	0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	1	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	2	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	3	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	4	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
Class 2	5	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-1	-2	-1.5	-3	-1.2
	6	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-3	-1.5	-3	-1.2	-8
	7	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-3	-1.2	-1	-2	-1.5
	8	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-2	-1.5	-3	-1	-2
	9	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-1.5	-1.7	-1.2	-10	-6
Class 3	10	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	11	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	12	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	13	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
	14	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf

Figure 6-11 HMM Transitions When Observations are Changed From Class2 to Class3

After the above-mentioned biased expectation, the $\xi_t(i,j)$ transition probabilities have to be normalized with the sum of all log likelihoods. The experimental results for Model2 are shown in Chapter 7.

6.4.3 HMM Classifier based on biased expectation Approach2 (Model 3)

The Model 3 approach's goal is the same as the for the Model2 approach, however, the modification of the Baum-Welch algorithm is done in the forward and backward algorithms, instead of changing $\xi_t(i,j)$. This is a more stringent label enforcement scheme which effectively zeros out entire trajectories rather than single states. The main idea is to set the probability values to zero of any state along the state trajectory has an inconsistent label. The general forward probability equation is shown below.

$$\alpha_{t+1}(j) \;=\; \left[\sum_{i=1}^{N} \alpha_t\,(i) a_{ij}\right] b_j(O_{t+1})$$

In the above equation, $b_j(O_{t+1})$ is the observation probability at state j while emitting the observation at t+1 at j^{th} state. Here, if j belongs to class 1 and the future observation O_{t+1} belongs to the other class, then the $b_j(O_{t+1})$ is set to zero in the update because the state at j is not consistent with future observation. To illustrate the effect of this approach, consider the HMM with three hidden states S1, S2, and S3 corresponding to the three activities and the observation sequence (1-standing, 2-walking,3-movement) with the following labels [1,1,1,1,1] for the time from t=1 to t=5. Here class1 is represented with S1, class2 is represented with S2 and class3 is represented by S3. In this example, since all the observation sequence labels belong to class1, the state S2 and S3 forward probabilities will become zero due to the Model3 biased expectation modification. This effect of the Model3 approach on the forward probabilities is shown in Table 6-3.

Observation Sequence Labels	1	1	1	1	1
Time steps	t=0	t=1	t=2	t=3	t=4
State S1	-1.83	-19.4	-31.6	-41.64	-47.45
State S2	-123.6	-inf	-inf	-inf	-inf
State S3	-917.9	-inf	-inf	-inf	-inf

Table 6-3 Forward Probability For The Observations [1,1,1,1,1]

Similarly, the forward probabilities for the observation sequence [1, 3, 3, 3, 2] are shown in Table 6-4

Observation Sequence Labels	1	3	3	3	2
Time steps	t=0	t=1	t=2	t=3	t=4
State S1	-4.488	-inf	-inf	-inf	-inf
State S2	-177.3	-inf	-inf	-inf	-78.6
State S3	-953.2	-21.6	-34.5	-41.86	-inf

Table 6-4 Forward Probability For The Observations [1,3,3,3,2]

Like the modifications in the above forward algorithm approach, the backward algorithm also temporarily sets observation probabilities to zero if the future observation is not consistent, i.e. belongs to a different class label than the existing hidden state class label. In the Model3 approach, since we modified the forward and backward algorithms, the Baum-Welch maximization step must be normalized, so that no transition probability is changed to zero since that would be unalterable later on. The normalization of transition probabilities, mean and covariance are shown in the equations below:

a) $\grave{a}_{ij} = \frac{[\sum_{t=1}^{T-1} \xi_t(i,j)]+1}{[\sum_{t=1}^{T} \gamma(i)]+N}$ Transition probability

b) $\mu_j = \frac{\sum_{t=1}^{T}\left[\gamma_t(j)+\frac{N}{T}\right] o_t}{\sum_{t=1}^{T}[\gamma_t(j)]+N}$ Estimated mean

c) $U_j = \frac{\sum_{t=1}^{T}\left[\gamma_t(j)+\frac{N}{T}\right](o_t-\mu_j)(o_t-\mu_j)'}{\sum_{t=1}^{T}[\gamma_t(j)]+N}$ Estimated covariance

Chapter 7
Experiments and Results

7.1 Model Validation on Synthetic Data

To verify the models, simple test data generated with a separate HMM with known model parameters is used to train the HMM models. The test data is generated for a length of 1000 samples using a HMM with three hidden states. The initialization of the model parameters is shown below.

$$\pi = [0.6 \quad 0.3 \quad 0.1]$$

$$A = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.1 & 0.8 & 0.1 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \qquad means = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 3.0 & -3.0 & 3.0 \\ 5.0 & 10.0 & 7.0 \end{bmatrix} \qquad covars = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

where π and A are the initial and transition probabilities, respectively.

Using the above model initialization parameters, three test sequences are generated. Among those three sequences, two sequences are used for training and the third sequence is used for testing. Model1 is trained with two sequences with a random initialization and generated the following mean and transition probabilities after training.

$$\text{Model-1 generated means} = \begin{matrix} 4.98 & 10.02 & 6.99 \\ 3.05 & -3.01 & 3.01 \\ -0.02 & 0.03 & -0.01 \end{matrix}$$

$$\text{Model-1 generated transition probabilities} = \begin{matrix} 0.77 & 0.09 & 0.12 \\ 0.10 & 0.81 & 0.08 \\ 0.08 & 0.08 & 0.83 \end{matrix}$$

The Model-1 generated transition probabilities and means are approximately the same as the test data initialization parameters, indicating its ability to reconstruct the model. It can be noticed here that due to the use of the standard Baum-Welch algorithm, the generated means and covariances for state1 and state2 are swapped within the model as the algorithm during training dynamically choose the hidden states depending on the observation and transition probabilities. This is one of reason Model1 approach is using the heuristic to identify the highest accuracy for all the combinations of state to label mappings. The trained model is used to test the third input sequence and achieved 99.90% of classification accuracy. That is, the Model1 generated state sequence with the corresponding state to label mapping is an almost perfect match to the test data state sequence.

7.2 Model1 experiments and results

The data collected for five subjects is used here and manually labeled. Each of the subject's data includes all three activities [3]. Four subjects are used to train the model using the Model1 approach which is then tested on all four training users and one the new subject. The results of the experiment are shown in Table 7-1.

Number of states = 3	
Subjects used for training	Model2 Classification Accuracy
User 1	78.92
User 2	92.38
User 3	83.80
User 4	91.92
Average accuracy = 86.75	
Subjects used for Testing	Model2 Classification Accuracy
User 5	92.30

Table 7-1 Model 1 Testing Results With Random Initialization

In in the next experiment, five cross-fold validation is performed 30 times, that is overall 150 times the model is trained with different random initializations and mean and standard deviation of the performance results are calculated. These results are shown in Table 7-2

30 Cross-fold validations	Mean	Standard Deviation
Training	82.18	17.85
Testing	88.40	18.89

Table 7-2 Model1 Experiment Results With 3 States for 30 Five-fold Cross Validations

7.3 Model2 experiments and results

As in the previous experiment, four subjects are used to train the HMM with the Model2 approach and tested against all four trained subjects and a new subject. The results of the experiment with the Model-2 approach is shown in Table 7-3.

Number of states = 3	
Subjects used for training	Model2 Classification Accuracy
User 1	91.76
User 2	84.73
User 3	92.53
User 4	91.57
Average accuracy = 90.15	
Subjects used for Testing	Model2 Classification Accuracy
User 5	79.80

Table 7-3 Model 2 Testing Results With Random Initialization

In Table 7-3, the testing accuracy is higher than the training average accuracy, because these results are based on a single random initialization of the Model2 approach. For a better accuracy, 30 five cross-fold validations are performed on all the five users. In the five cross-fold validation, the test and training users are swapped for all possible combinations. The mean and standard deviation of all these 30 five cross-fold validations with 3 hidden states and 6 hidden states are shown in Table 7-4 and Table 7-5.

	Mean	Standard Deviation
Training	88.25	1.18
Testing	88.27	5.22

Table 7-4 Model2 Experiment Results With 3 States for 30 Five-fold Cross Validations

30 Cross-fold validations	Mean	Standard Deviation
Training	88.09	1.73
Testing	87.85	6.66

Table 7-5 Model2 Experiment Results With 6 States for 30 Five-fold Cross Validations

Although the means of training and testing are similar, the standard deviation of the test data is very high when compared to training. Therefore, the training accuracy is more consistent than testing accuracy (which is to be expected). The results of three states are better than six states. This might indicate that the HMM is a local optimal solution algorithm and the results vary based on initialization of the model.

7.4 Model3 experiments and results:

The same five subjects used in Model2 are used for testing Model3. The test results with one initial random initializations are shown below Table 7-6.

Number of states = 3	
Subjects used for training	Model3 Classification Accuracy
User 1	79.42
User 2	92.34
User 3	92.57
User 4	91.92
Average accuracy = 89.06	
Subjects used for Testing	Model2 Classification Accuracy
User 5	84.23

Table 7-6 Model3 Testing Results With Random Initialization

The Model3 approach is also tested with three states and six states against five subjects and the results are shown in Table 7-7.

	Mean	Standard Deviation
Training	88.03	1.96
Testing	88.17	5.40

Table 7-7 Model3 Experiment Results With 3 States for 30 Five-fold Cross Validations

	Mean	Standard Deviation
Training	87.91	2.32
Testing	87.01	7.19

Table 7-8 Model3 Experiment Results With 6 States for 30 Five-fold Cross Validations

7.5 Model2 Semi-Supervised Learning

In this experiment, supervised HMM models are used to label the unlabeled users' data. Then both labeled and unlabeled users' data is used to train a new generalized HMM model. For this experiment, the labeled data is collected for 5 subjects and unlabeled data is extracted for 20 subjects. Four HMM models are constructed from labeled data with different random initializations and each HMM model is used to label all 20 subjects. After labeling, all 25 subjects (combined data) are used to build a new HMM model and performed a five cross-fold validation. The results of this Semi-Supervised approach are shown in the table below. Thes steps of this approach are shown below:

1. A HMM Models is generated from 5 labeled subjects
2. Label the unlabeled data for 20 subjects using the HMM model from step1.
3. Combine all the labeled and unlabeled data that is combined data contains overall 25 subjects.
4. Building a new HMM models using combined data.
5. The new trained HMM model from step4 is used for testing 5 labeled users in a five-fold cross validation manner.

To test this approach, five-fold cross validation is again applied and in each fold Step5 is repeated 30 times to calculate the testing and training accuracies. The results are shown REF _Ref48149
9769 \h

Table 7-9.

	Mean	Standard Deviation
Training	95.42	0.37
Testing	93.43	10.46

Table 7-9 Semi-Supervised Learning Results

The results show that the generalized models with a larger number of subjects have higher accuracy.

Chapter 8
Conclusion and Future work

Activity detection and recognition is an important function in the context of health monitoring. In this thesis, an approach using regression and Hidden Markov Models is presented and evaluated on smart floor data. Three modified versions of Baum-Welch are used and extended into a semi-supervised training approach. The proposed three models are shown to be able to detect and classify the activities such as standing, walking, and other class of movement with high accuracy on the experimental test data obtained form a real smart floor. The heuristic method introduced in Model1 does not scale well with the increase in the number of states. Model2 and Model3, on the other hand, have no such limitations and achieved approximate similar and even better results than Model1. In all cases, some manual labeling of data has to be performed. TO reduce this burden, a semi-supervised approach is finally tested and evaluated that uses both, a small set of labeled data and a larger amount of unlabeled data. The preliminary experiments performed here show that this approach benefits from the unlabeled data and achieves higher accuracy than using only the labeled data.

References

[1] Q. Ni, A. B. G. Hernando, and I. P. de la Cruz, The elderly???s independent living in smart homes: A characterization of activities and sensing infrastructure survey to facilitate services development, vol. 15, no. 5. 2015.

[2] N. B. Burns, P. Sassaman, K. Daniel, M. Huber, and G. Záruba, “PESTO : Data Integration for Visualization and Device Control in the SmartCare Project,” 2016.

[3] O. Oluwatosin, “Gait analysis on a smart floor for health monitoring,” no. May, 2015.

[4] S. Ji, W. Xu, M. Yang, and K. Yu, “3D Convolutional Neural Networks for Human Action Recognition,” vol. 35, no. 1, pp. 221–231, 2013.

[5] R. Bodor, “Vision-Based Human Tracking and Activity Recognition,” Proc. 11th Mediterr. Conf. Control Autom., vol. Vol. 1, 2003.

[6] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, “Activity Recognition and Monitoring Using Multiple Sensors on Different Body Positions,” pp. 4–7, 2006.

[7] L. Bao and S. S. Intille, “Activity Recognition from User-Annotated Acceleration Data,” pp. 1–17, 2004.

[8] D. Sundararajan, “The discrete Fourier transform: theory, algorithms and applications.” World Scientific, 2001.

[9] L. R. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” Proc. IEEE, vol. 77, no. 2, pp. 257–286, 1989.

[10] A. Viterbi, “Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm,” pp. 260–269, 1967.

[11] M. Aly, “Survey on Multiclass Classification Methods Extensible algorithms,” no. November, pp. 1–9, 2005.

[12]http://scikitlearn.org/stable/modules/generated/sklearn.linear_mdel.LogisticRegression.html

Biographical Information

Anil Kumar Mullapudi received his bachelor's degree in Information Technology from S.R.K.R. Engineering College, affiliated to Andhra University, Visakhapatnam, India in 2008. He worked in Intense Technologies, Hewlett-Packard, and Verizon as a software engineer for a period of five years.

In 2015, he started his masters in computer science at the University of Texas at Arlington. He worked as teaching assistant for Design and Analysis of Algorithms course and worked as research assistant in University of Texas at Arlington Research Institute(UTARI). His research interests are Robotics, Machine Learning, Neural Networks.