AN ANALYSIS OF HUMAN MOVEMENT ACCELEROMETERY DATA FOR STROKE REHABILITATION ASSESSMENT

SHANE HALLORAN

Thesis submitted for the degree of

Doctor of Philosophy

School of Mathematics, Statistics & Physics

Newcastle University

Newcastle upon Tyne, United Kingdom

September 2019

Acknowledgements

Firstly, I would like to express my gratitude to my supervisory team and close collaborators who provided me with invaluable mentoring and were a pleasure to work closely with during my PhD: Jian Qing Shi, Yu Guan, Thomas Ploetz, Lin Tang and Nils Hammerla. Secondly, I would like to thank the *EPSRC Centre of Doctoral Training in Cloud Computing for Big Data* staff, who provided an excellent environment for me to research and study in over the past five years: Paul Watson, Darren Wilkinson, Matthew Forshaw, Sarah Heaps, Barry Hodgson, Oonagh McGee, Jennifer Wood and Andrew Turnbull.

I would also like to thank my family, especially my parents for the assistance which they provided me with, as always. My parents often remark on how pleasant each and every PhD student is when they visit. In particular I would like to warmly acknowledge Ossama Alshabrawy, Alex Brown, Mario Parreño Centeno, Xi Chen, Richard Cloete, Tom Cooper, Michael Dunne-Willows, Matt Edwards, Darren Fletcher, Naomi Hannaford, Jonny Law, Lauren Roberts, Ashleigh McLean, Jinzhao Liu, Dan Jamieson, Peťo Michalák, Tim Osadchiy and Pengcheng Zeng who have been a major source of support.

Finally, I would like to thank friends and people who I met along the way who also helped enrich my journey in various ways: Taiki Isami, Richard Brady and Ian MacKay, Nurses Jennifer, Lynn, Lynne, Katie, Alison and Debbie, Pol Bolívar, Mark Brunton, Michael Carey, Mary and Thomas Delaney, John Hinde, Mia Jiang, Kevin Jennings, Beth Gadsby, Fanting Hu, Yanling Jin, Mohammed Ali Kazi, Stephen Kinsella, William Laroche, Thomas Leahy, Bob Loughnane, Concepta McHugh, Alan Molloy, Daniel and Michael O'Connell, Lil O'Mahony, Erik and Christina Osborne, William Power, Priscilla, Niccolò Rivato, Jerome Sheehan, Evgeniya Shmeleva, Finbarr O'Sullivan, Jonathan Walsh, Binbin Wei, Xhaça and Cheda.

Abstract

Human Activity Recognition (HAR) is concerned with the automated inference of what a person is doing at any given time. Recently, small unobtrusive wrist-worn accelerometer sensors have become affordable. Since these sensors are worn by the user, data can be collected, and inference performed, no matter where the user may be. This makes for a more flexible activity recognition method compared to other modalities such as in-home video analysis, lab-based observation, etc. This thesis is concerned with both recognizing subjects activities as well as recovery levels from movement-related disorders such as stroke.

In order to perform activity recognition or to assess the degree to which a subject is affected by a movement-related disease (such as stroke), we need to create predictive models. These models output either the inferred activity (e.g. running or walking) in a classification model, or else the inferred disease recovery level using either classification or regression (e.g. inferred Chedoke Arm and Hand Activity Inventory Score for stroke rehabilitation assessment). These models use preprocessed data as inputs, a review of preprocessing methods for accelerometer data is given.

In this thesis, we provide a systematic exploration of deep learning models for HAR, testing the feasibility of recurrent neural network models for this task. We also discuss modelling recovery levels from stroke based on the number of occurrences of events (based on mixture model components) on each side of the body. We also apply a Multi-Instance Learning model to model stroke rehabilitation using accelerometer data, which has both visualization advantages and the potential to also be applicable to other diseases.

Contents

List of Symbols

$\#\text{Kern}_l$ The number of convolutional kernels on the l^{th} layer of a CNN.

δ The learning rate in the backpropagation algorithm for training a neural network.

$\delta_{aij,hk}$ Binary variable denoting whether $\boldsymbol{x}_{aij,h}$ belongs to the k^{th} cluster.

γ The parameter associated with momentum in the Adagrad learning rule

$\hat{L}_{aij,hk}$ The inferred probability that datapoint $\boldsymbol{x}_{aij,h}$ belongs to the k^{th} mixture component. $\hat{L}_{aij,hk} \in [0, 1]$

ι A hyperparameter which denotes which less discriminatory datapoints to exclude, as shown in Equation (4.11).

$\boldsymbol{X}_{*,h}$ The h^{th} (sliding-window) datapoint in our dataset, from all patients at all times of data collection, in a matrix of dimension w timestep samples $\times$ d sensor channels.

$\mathbf{x}_{*,i}$ Another version of $\boldsymbol{X}_{*,i}$, where the contents are flattened into a vector of length dw.

$\text{act}_{i,m}^{(l)}$ Activation value of the mth node on the lth layer for the ith datapoint

π_k, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, $0 < k < K$ Parameters of a Gaussian Mixture Model with K components.

$\rho_{h,m}^{i}$ The probability that subject i is undertaking activity class m during the hth sliding window

$\tilde{\ell}(\boldsymbol{\pi}, \boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{L})$ The estimated likelihood from each *M*-step of the EM algorithm.

$\varrho_{j \to i_{\tau+1}}^{(l)}$ Used to change the learning rate in the Adagrad learning rule

ϑ The decay of the learning rate on each iteration of the backpropagation algorithm for training a neural network.

$\boldsymbol{\theta}_k$ The parameters of the k^{th} component of a Gaussian Mixture Model, $\boldsymbol{\theta}_k = \{\boldsymbol{\mu_k}, \boldsymbol{\Sigma}_k\}$.

$\boldsymbol{Q}_i, \boldsymbol{\Psi}$ Used in principal component procedure

$\boldsymbol{s}_h$ Covariates generated from the h^{th} sliding window datapoint in a neural network model, used later for calculating prediction.

$\boldsymbol{x}$ Denotes collection of sliding window datapoints from all patients, all weeks from either limb, where each datapoint is from one limb only.

$\boldsymbol{x}_i$ Denotes the collection of all sliding window datapoints from patient i.

$\boldsymbol{x}_{aij,h}$ The hth sliding window from the ith patient on the jth week from locations on the body indexed by a.

$\boldsymbol{x}_{aij}$ Denotes the collection of all sliding window datapoints from patient i on week j for the limb indexed by a.

$\boldsymbol{x}_{ij}$ Denotes the collection of all sliding window datapoints from patient i on week j.

$\boldsymbol{L}_{aij}$ Vector representing all of the latent component-memberships in a GMM of sliding window datapoints from the i^{th} patient's a^{th} limb from the j^{th} week.

$\zeta'(\cdot)$ The inferred recovery score of a sliding window datapoint, but replaced with 0 if a criterion is not met, as shown in Equation (4.11).

$\zeta(\cdot)$ The inferred prediction of recovery score of a sliding window datapoint using a regression method, on a continuous scale.

a The index number referring to each limb, or the data from the accelerometer sensor placed on it. Where $a = 1$, the data is from the paretic side. Where $a = 2$, the data is from the non-paretic side.

$a_x(t), a_y(t), a_z(t)$ The values of the X,Y and Z sensor channels at time t.

$b_{0,m}^{(l)}$ Intercept term in neural network calculation of the activation of the m^{th} node on the l^{th} layer

$b_{j\to m}^{(l)}$ Parameter in neural network, denoting the linear effect of that node j on the previous layer has on node i of layer J.

d The number of sensor channels present in the data

H'_{ij} The number of elements of patient i's data for which $\zeta'(\boldsymbol{x}_{ij,h}) \neq 0$.

H_{ij} The number of sliding window datapoints from patient i on week j, where datapoints on each limb are treated separately.

L The number of layers in a neural network model.

$L_{aij,hk}$ Binary variable denoting if the latent variable $L_{aij,h}$ belongs to the k^{th} cluster.

$L_{aij,h}$ Latent variable denoting which component the h^{th} sliding window datapoint from the i^{th} patient from the j^{th} week from the a^{th} limb. $L_{aij,h} \in \{1, \ldots, K\}$.

L_{seq} The length of subsequence used to train an LSTM model.

M_l The number of nodes in the l^{th} layer of a neural network.

n The number of patients or other human subjects in the dataset.

n_i The number of weeks of data which were collected for patient i

p The number of supervised learning classes which can be associated with each datapoint, for Human Activity Recognition.

r_l Convolutional filter length on the l^{th} convolutional layer in CNNs.

$SVM(t)$ The Signal Vector Magnitude of the signal at time $t \in \mathbb{R}$.

t_k The time at the k^{th} index of data

$u(\cdot)$ The Rectified Linear Unit activation function, defined in Equation (2.6).

w The integer number of samples in each sliding window

z_{ijk} One of K covariates, denoting the difference in number of occurrences of mixture component k on each side of the body for patient i on week j.

$\mathbf{\Psi}$ A covariance matrix of a dataset used in Principal Components Analysis in Section 2.7.

$\boldsymbol{a}$(t) The continuous-time acceleration process at time t, usually multivariate

List of Figures

List of Tables

Chapter 1

Introduction

The overall aim of this project is to improve modelling and feature extraction approaches for accelerometer data, especially for healthcare data in automated stroke rehabilitation assessment. In this thesis, predictive models are created for human activity recognition (HAR) — i.e. classification of the human subjects' current activity into discrete classes such as walking or sitting — as well as the regression problem of inference of the Chedoke Arm and Hand Inventory (CAHAI) score for stroke rehabilitation assessment. All of the accelerometer data approaches make use of the preprocessing and feature extraction approaches in Chapter 2.

The activity classification for HAR is primarily discussed in Chapter 3, along with a systematic exploration of hyperparameters for deep learning models and of the feasibility of recurrent neural networks for this task.

In Chapter 5, we model recovery levels from stroke based on two sets of covariates: one based on the number of occurrences of events (mixture model components) on each side of the body, and another set of covariates based on a Multi-Instance Learning (MIL)-based model. In Chapter 4, we outline our methods for generating these sets of covariates. We also outline the visualisation advantages of both approaches and the potential of the MIL-based approach to also generalise to other diseases.

1.1 Background

As mentioned in the abstract of this thesis, accelerometer sensors and models based on data collected from them have the potential to improve applications in healthcare and in other areas. A typical application using accelerometer data uses a feature extraction approach, followed by modelling the desired quantity related to the human movement observed. An overview of feature extraction methods used for accelerometer data is given in Chapter 2, but mainly comprise of time domain, frequency domain, symbolic and neural network features (Figo et al., 2010). One possible aspect of human movement we may wish to model is in human activity that is recognising which of p a user is undertaking at any given time (e.g. running, walking, sitting, etc), where the set of classes is defined for each individual application. There has been extensive work in this field for many different applications. Non-statistical based methods typically involve using basic calculated criterion to determine occurrence of events of interest — for example to determine walking step counts or to detect when the human subject has suffered a fall (Bulling et al., 2014). Alternatively, more sophisticated models can be based on supervised learning models for time series data. In this thesis, we outline the background and explain our work on modelling stroke recovery using accelerometer data, as well as exploring the suitability of neural network-based methods for HAR in Chapter 3. In the rest of this section we discuss the background to stroke as a disease and briefly discuss the background of usage of accelerometer data for modelling stroke recovery. A more extensive background review of modelling stroke recovery using accelerometer data is given in Section 5.1.

Stroke, a leading cause of disability and death, occurs when a blood clot cuts off oxygen supply to a region of the brain. Often, hemiparesis, a reduction in the ability to perform activities using the opposite side of the body to where the blood clot occurred, results. Patients can recover some of their capabilities with intense therapeutic input.

Approaches to monitoring patient's recovery in the time after their stroke discussed

in the literature include brain imaging (Wintermark et al., 2005), questionnaire-based approaches, and lab-based clinical assessments. Advanced methods such as brain imaging are not commonly used in clinical routine (Santisteban et al., 2016), often due to cost, and moreover does not provide image of the brain during completion of naturalistic tasks in their living environment.

Questionnaire-based approaches enquire about physical or functional capabilities over the past several days, and can be divided into two types: patient-completed and caregiver-completed. Both are subject to recall bias, since the patient may not remember their activities, and also caregivers may not be with the patient the entire time to be able to observe the patient (Ferrari et al., 2007).

Lab-based task assessments exist for physical ability (capability of moving parts of each upper limb, such as the Fugl-Meyer Score and Wolf Motor Function Test) and functional capability (completing tasks in their preferred manner, including using the dominant upper limb to compensate for weakness in the non-dominant hand).

On the other hand, accelerometer sensors, which measure the acceleration force produced by movement of limbs they are affixed to, have the potential to provide a more objective measure of recovery levels of patients. In Chapter 4, we outline how we generate useful covariates from long-term accelerometer data for use in (automated and remote) prediction of rehabilitation levels. In Chapter 5, we outline a predictive model for the Chedoke Hand and Arm Inventory (CAHAI) score of patients, using the aforementioned covariates.

1.2 Contributions in this thesis

In this thesis the main contributions are threefold, we provide an exploration of deep learning for human activity recognition (through classification models for activities), as discussed in our published paper Hammerla et al. (2016) and Chapter 3 of this thesis. We

also develop new covariate extraction methods for stroke rehabilitation recovery assessment (from accelerometer data), as discussed in our two papers Halloran et al. (2019) and Tang et al. (2019) (the first of which has been published, and the latter of which is currently under review) and Chapter 4. These features are used in a non-linear mixed effects model (NLME) for longitudinal data analysis. While this is a mature statistical method, it is new to the field of healthcare accelerometery data analysis, and is discussed in our paper Tang et al. (2019) (currently under review) and in Chapter 5. The resulting predictive model uses either of these sets of covariates automatically and accurately assesses how well patients recover, without human intervention.

1.3 Structure of Thesis

The structure of the thesis is as follows, and is shown in Figure 1.1.

In Chapter 2, we give an overview of feature extraction methods used in the literature of analysis of accelerometer data, many of which are used throughout the rest of the thesis.

In Chapter 3, we discuss the usage of neural network models for Human Activity Recognition (HAR) for activity classes, and then we outline some of the findings from the experiments exploring the feasibility of advanced neural networking models for Human Activity Recognition in our published paper (Hammerla et al., 2016).

In Chapter 4, we outline our approaches to extraction of new covariates (for assessment of stroke patient recovery using accelerometer data) and associated visualisation of their workings, based on some of the methods described in Chapter 2. These are discussed in our published paper (Halloran et al., 2019), and in our paper currently under review (Tang et al., 2019).

In Chapter 5, we give an account of the non-linear mixed effects (NLME) predictive model for longitudinal data analysis, and predictive results from using the covariates obtained using the methods described in Chapter 4. This is discussed in our paper which is

currently under review (Tang et al., 2019).

In Chapter 6, we give a conclusion of our results from Chapters 3, 4 and 5.

1.A A note on notation in this thesis

We use bolded lowercase letters to denote vectors and functions which take vector values, for example $\boldsymbol{a}_1$ and $\boldsymbol{a}(t_1)$ respectively.

We use uppercase bolded letters to denote matrices or tensors, for example $\boldsymbol{C}$.

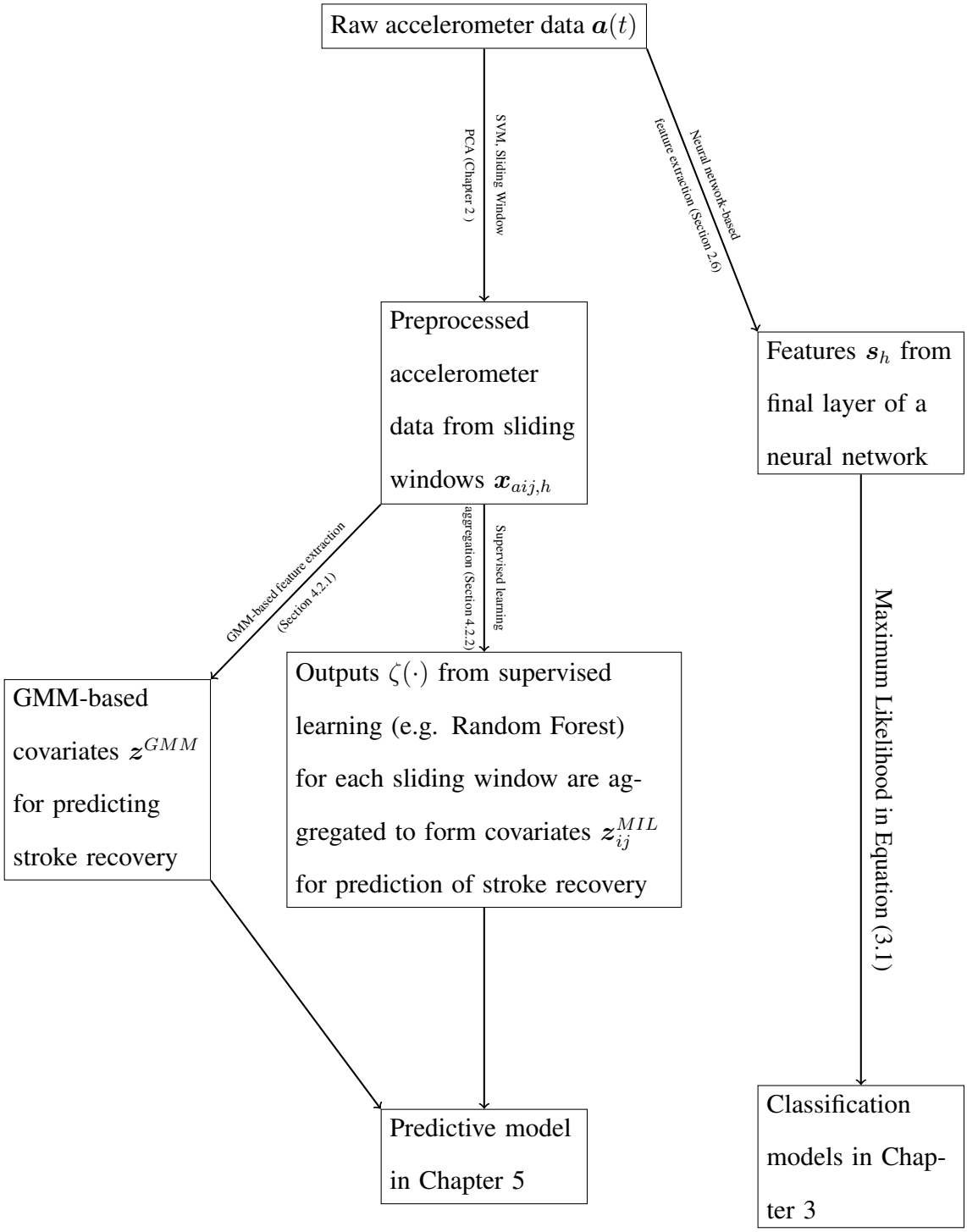


Figure 1.1: Structure of thesis and notation used

Chapter 2

Introduction to Feature Extraction for Accelerometer Data

In this chapter, we review a commonly discussed part of machine learning and data analysis pipelines, *feature extraction* (Figo et al., 2010; Hinton and Salakhutdinov, 2006), where relevant features, perhaps expressing the most useful characteristics of data samples, in a low dimension, are extracted. We provide an outline of the approaches used on human accelerometer data in the literature, including those based on time, frequency and symbolic domains, neural networks and dimensionality reduction methods and the usage of mixture models on sliding windows of the data to extract features. These methods will be used in the later chapters, in Chapter 3 for activity classification and in Chapters 4 and 5 in remote rehabilitation assessment of stroke patients.

2.1 Purpose of feature extraction

We may find that while data may be from a physical process which is of great interest to us, in its raw form the data is too heterogeneous or in too high a dimension to be useful due to the curse of dimensionality (Bellman, 1961), especially if we consider using the

sliding window procedure as shown in Section 2.3. A preprocessing function transfers the function into a space where the data is in a more useful form, where there may be better separation between classes of interest. In the rest of this chapter, we discuss sampling the accelerometer data as well as common feature extraction approaches.

2.2 Accelerometer movement process

In carrying out movements and activities of interest, acceleration forces are exerted on locations around the body of a human subject where sensors are situated. As part of our discussion of acceleration signals, we use the notational conventions for vectors (bolded lowercase) and matrices (bolded uppercase) mentioned in the Appendix of Chapter 1.

Let $\boldsymbol{a}(t) \in \mathbb{R}^d$ be some acceleration process in continuous time $t \in [0, T)$. Usually this process has $d = 3$ channels per location of interest on the body (where a sensor may be placed), denoting the anterior-posterior direction, medio-lateral direction, and vertical relative acceleration directions (often referred to as the 'X', 'Y' and 'Z' directions). In the case of one sensor location, then the vector is simply $\boldsymbol{a}(t) = (a_x(t), a_y(t), a_z(t))$, where $a_x(t)$ is the acceleration measure at time t on the 'X' direction, and likewise for $a_y(t)$ and $a_z(t)$. In this thesis we assume that there is no error in the sensed acceleration signal (unless otherwise stated). This is likely to be an unrealistic assumption.

Concurrently, $c(t) \in \{1, ..., p\}$, also in continuous time, denotes which of p possible activities the human subject is carrying out while the force is exerted. It is hypothesised that there is a relationship between the activity the subject undertakes and the forces exerted at the sensor locations.

In the next sections, we consider the observed realisations of this process, how we preprocess this data, approaches to modelling the data, as well as methods to calculate covariates for our models.

2.2.1 Data sampling and observed data

These processes are observed at N discrete timepoints , $t_1, ..., t_N$. The sampling rate, the reciprocal of the duration of time between t_i and t_{i+1}, can be chosen on a case-by-case basis for each application (for example taking into account between accuracy and battery life).

In Figure 2.21, we give a schematic example of continuous data being sampled at a rate of $R = 100Hz$ (100 times per second) at $N = 251$ discrete timepoints, i.e. $\boldsymbol{a}(t_1)$ is sampled at 0.00 seconds from the start of the data recording, $\boldsymbol{a}(t_2)$ is sampled at 0.01 seconds, and so forth, until $\boldsymbol{a}(t_{251})$ is sampled at 2.50 seconds from the start. In practical applications, of course, the duration of data recorded is likely to be much longer than in this illustrative example. We see that for each of the $N = 251$ discrete sampling points over $T = 2.50$ seconds of data, a vector of length $d = 3$ is sampled.

For notational simplicity, we denote the vector of sampled accelerometer data as $\boldsymbol{a}_i = \boldsymbol{a}(t_i)$ at the i^{th} sampling timepoint (and assume that there is no sensor error). The corresponding activity label is denoted as $c_i = c(t_i)$. $i \in \mathbb{N}, i \leq (TR) + 1$

In the rest of this chapter, we will discuss bow to extract features from this observed (sampled) data.

2.3 Time domain

The simplest feature extraction techniques involve basic statistical summaries of the amount of movement in a sliding window around a timepoint, and can be related to the average amount of force (and by proxy human energy level exerted) during that period of time. The sliding window duration can be chosen based on the typical movement durations in the application of interest.

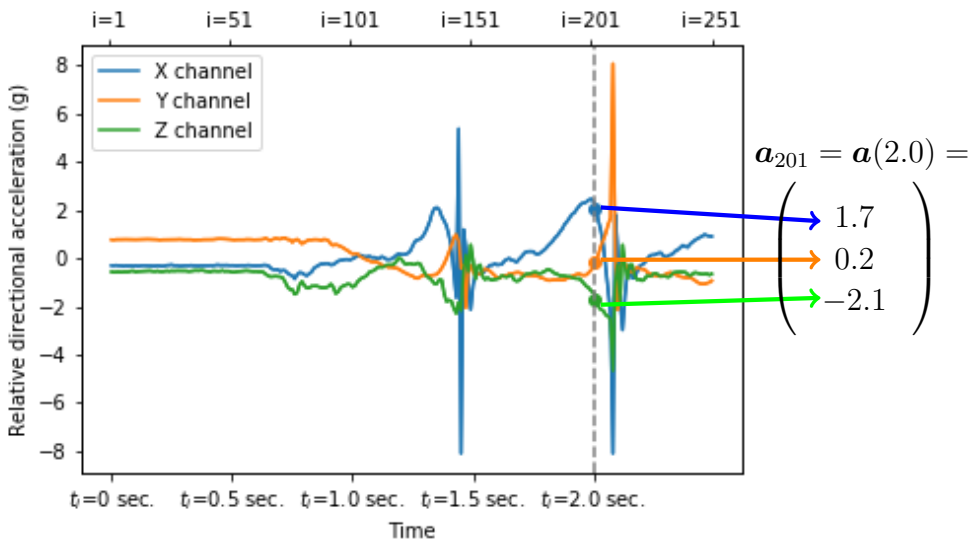


Figure 2.21: Schematic diagram of accelerometer data sampling.

Given a multivariate time series with N discrete timesteps $\boldsymbol{a}_1, ..., \boldsymbol{a}_N$ as input, and using the notation defined in Section 2.2.1, the h^{th} sliding window of duration w samples, with 50% overlap, is defined as (in matrix form):

$$\boldsymbol{X}_{*,h} = \{\boldsymbol{a}_k : k_{*,h}^{MID} - \frac{w}{2} \leq k < k_{*,h}^{MID} + \frac{w}{2}\} \tag{2.1}$$

where $k_{*,h}^{MID} = w \times \frac{h}{2}$ is the midpoint of the h^{th} sliding window, $k < N$, $0 \leq h \leq \frac{N}{w}$, $h, k \in \mathbb{N}$.

The matrix form of $\boldsymbol{X}_{*,h} \in \mathbb{R}^{d \times w}$ is used especially in Section 2.6.2, but we more commonly use the flattening of this matrix into a vector $\boldsymbol{x}_{*,h} \in \mathbb{R}^{dw}$.

The subscript on the left hand side of Equation 2.1 denotes the h^{th} datapoint from a dataset of all human subjects, without specifically referring to a particular patient, location on the body or episode of data collection. In later chapters, we will use $\boldsymbol{x}_{aij,h}$ to refer to the h^{th} sliding window from the i^{th} patient on the j^{th} week from locations around the body indexed by a. We will also use $N_* = \frac{N}{w}$ to denote the number of sliding windows generated.

In our sliding window models, we assume the activities associated with nearby frames are not temporally dependent. However, in reality this is not the case: overlapping frames will contain samples of data from the same activity, and nearby frames will likely coincide with the occurrence of correlated activities. When we partition our data, frames from our test set may be close by to those assigned to our training set, which they may be correlated with those assigned to our test set (Hammerla and Plötz, 2015).

For each sliding window, some measurements based on the time domain we can take are the mean, standard deviation minimum, maximum and range of measurements on each sensor channel or on all channels combined. These measurements can denote the energy of movement in each direction from a subject's body, as well as gestures when the user is still (due to the effect of gravity due to the direction that a person may be standing in, for example). Another statistic used from sliding windows is the number of zero crossings, i.e. the number of times each of the sensor channel signals crossed the zero axis. This usually indicates the number of repetitions of actions of a user, and for example may indicate the difference between walking (fewer zero crossings) and running (more zero crossings). The Signal Moving Average (SMA) is the sum of the accelerometer signals over time duration of the sliding window. Should the duration of the sliding window be long, it may correlate with the amount of energy a patient has expended over that duration. It is closely related to the Signal Vector Magnitude (SVM), which measures how different the radius of acceleration is to that of a resting position:

$$SVM(t) = \left| \sqrt{a_x(t)^2 + a_y(t)^2 + a_z(t)^2} - 1 \right| \tag{2.2}$$

These measurements tend to be useful for distinguishing different activities which require significantly different levels of human exertion (for example recognising when a patient is idle compared to when a patient is walking).

2.4 Frequency Domain

Since many human activities of interest could be based on repetitive motions which are the similar speed each time it is carried out, and different patients often carry out these actions within the same range of speeds, frequency domain features could be useful. These techniques decompose a sliding window into coefficients based on Fast Fourier Transform (FFT) or wavelet. FFT allows one to convert a signal from the time domain into the frequency domain, and analyse the frequencies of activity present. Some frequencies may be particularly interesting to certain applications, for example walking activity may lie in the 0.5Hz to 2.5Hz range (Sharma et al., 2008). Wavelet transforms involve computing the dot-product similarity between the sliding window and a mother wavelet which has been multiplied by various amplitudes and stretched by different frequencies (Daubechies, 1990).

2.5 Symbolic and mixture-based features

Symbolic domain features compress the signal into a string of discrete symbols, representing similar observations. For example, in Piecewise Aggregate Approximation (PAA), symbols strings can initially be formed from the construction of optimally-fitting piecewise-constant approximation to the signal, where the values of the piecewise constant parts are determined from percentiles of values of the signal. These methods are sometimes used in combination with Dynamic Time Warping, which enables the comparison of signal windows of different length when possibly two instances of activities have the same accelerometer signal representation apart from the speed at which they are carried out (Lin et al., 2007).

Another approach to symbolic string representations of accelerometer data is by using Gaussian Mixture Models (GMMs) over sliding windows. Chapter 5, we make exten-

sive use of GMMs to create symbolic representations of the data over sliding windows. A mixture model represents the presence of distributions of a subpopulation in an overall population, without requiring that any observed data explicitly indicates which subpopulation component it belongs to. GMMs are widely used in data mining, pattern recognition, machine learning and statistical analysis due to their capability of representing different subpopulations in an overall sampling distribution (Bishop, 2006). Therefore, for the preprocessed data set, we consider a GMM with fixed number of components to find a single set of clusters which exist within both the paretic and non-paretic data.

Suppose that for the data from all patients, the extracted data set is denoted as $\{\boldsymbol{x}_{*,1}, \ldots, \boldsymbol{x}_{*,N}\}$, $\boldsymbol{x}_{*,h} \in \mathbb{R}^{dw}$. Our goal is to partition the data set into some given number K of clusters. For this, data vectors are assumed to be generated from a mixture of Gaussian distributions. Let $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, $(k = 1, \ldots, K)$ denote the probability density function of the Gaussian distribution with mean vector $\boldsymbol{\mu}_k$ and covariance matrix $\boldsymbol{\Sigma}_k$ representing the clusters, then the distribution of $\boldsymbol{x}_{*,h}$ can be expressed with a mixture of K Gaussian distributions as

$$p(\boldsymbol{x}_{*,h}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_{*,h} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{2.3}$$

where π_k is the probability of the data belonging to the k-th component and typically $\{\pi_k : k = 1, \ldots, K\}$ satisfy

$$0 \leq \pi_k \leq 1, \text{ and } \sum_{k=1}^{K} \pi_k = 1. \tag{2.4}$$

Details of the expectation-maximization inferential procedures we have used for GMMs are given in Section 4.2.1.

Some problems which can occur with this model are (1) avoiding the possibility of degenerate components, where the variances along the diagonal of each $\boldsymbol{\Sigma}_k$ tends closer to 0 on each iteration and (2) we have no principled way to choose the number of components

in the model (apart from, for example, hyperparameter cross validation). A solution to both of these problems is to use a Dirichlet Process Gaussian Mixture Model (DPGMM). When computed using a variational inference approach, the model and inferential procedure for a DPGMM are very similar to that above, apart from the use of Maximum A Posteriori (MAP) estimation instead of maximum likelihood for the component distribution parameters, and the weights of each component are governed by a Dirichlet-Multinomial distribution, instead of being estimated as scalars $\pi_1, ..., \pi_k$ (Blei et al., 2006).

Using the EM algorithm for fitting a Gaussian Mixture model is similar to using a K-Means clustering algorithm, apart from providing soft cluster assignment probabilities which indicate the degree of certainty. Indeed, where the covariance matrices of each Gaussian are constrained to be diagonal and as the elements of the diagonal tend toward zero in the limit case, they are equivalent (Bishop, 2006; MacQueen et al., 1967).

Other approaches to estimating the parameters of the mixture model include Markov Chain Monte Carlo (MCMC)-based methods. An ergodic Markov Chain is created which has a stationary distribution equal to the posterior distribution of the parameters of the mixture model (McLachlan et al., 2019). Variational inference is considered to be less computationally intensive on larger datasets compared to MCMC-based methods (Blei et al., 2017).

Mixtures for time series have been developed where it can not be assumed that concurrent observations are independent (Nguyen et al., 2016). However, in applications such as our long-term accelerometer data which we discuss in Chapters 4 and 5, this may be a more realistic assumption as data dependent on an observation will amount to a very small fraction of our total duration of data.

Mixture models have also been used with non-Gaussian component distributions. For example, mixtures of (multivariate) Bernoullis have been used for clustering text documents, based on binary vectors encoding which words were contained in each document

(Juan and Vidal, 2002).

2.6 Neural Network-based Features

Many of the features discussed previously require quite a large degree of domain-specific knowledge in order to choose the appropriate ones for each application. Alternatively, in the case where we have access to response data for each datapoint (i.e. an activity label for each second), we can make use of neural network models in order to automatically create a prediction function from the initial raw data without the need for preprocessing. The neural network model achieves this firstly through usage of a high level of parametrization so that a large number of possible prediction functions can be approximated through the Universial Approximation Theorem. Secondly, the neural network makes use of regularization, so that the learning procedure doesnt result in a prediction function which is unlikely to be useful.

In the next subsections, we outline the structure of feedforward, convolutional and recurrent neural network models, the parameter estimation algorithms, as well as show some experiments on some lab-based accelerometer data.

2.6.1 Calculating covariates using feedforward neural network

A feedforward neural network acts as a function which outputs $\boldsymbol{s}_h$ for each flattened input window $\boldsymbol{x}_{*,h}$ using a highly non-linear function. This function is the composition of a series of hidden layers, each of which acts as a learnable non-linear function of its inputs, with learnable parameters and a non-linear activation function. Theoretical results (Cybenko, 1989; Hornik, 1993) show that for any underlying non-linear function and multi-layer perceptron architecture, there exists a set of parameters which approximate this function.

For the h^{th} input vector $\boldsymbol{x}_{*,h}$, the activation for the m^{th} node on the first layer is calculated as a linear combination with bias before a non-linearity function u (such as Equation (2.6) is applied:

$$\text{act}_{h,m}^{(1)} = u(b_{0,m}^{(1)} + \sum_{j=1}^{w} \boldsymbol{x}_{*,h_j} b_{j\to m}^{(1)}) \qquad b_{0,m}^{(1)}, b_{j\to m}^{(1)} \in \mathbb{R} \tag{2.5}$$

Where $\boldsymbol{x}_{*,h_j} \in \mathbb{R}$ is the j^{th} element of the input vector. (As we mention in Section 2.3, $\boldsymbol{x}_{*,h} \in \mathbb{R}^{wd}$. We denote the j^{th} element of this vector as $\boldsymbol{x}_{*,h_j} \in \mathbb{R}$). Each $b_{j\to m}^{(1)}$ is a scalar learnable parameter, which denotes the effect of the j^{th} element of the input on the m^{th} node on the first layer.

We employ the ReLU (Rectified Linear Unit) activation function.

$$u(\cdot) = \max(\cdot, 0) \tag{2.6}$$

The derivative of the ReLU activation function is:

$$u'(\cdot) = \begin{cases} 1, & \text{if} \quad \cdot > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2.7}$$

Since the derivative of the activation function for *active* units (those corresponding to the first case in Equation (2.7)) in our network is always 1, then errors do not vanish when we repeatedly multiply derivatives as we descend layers in the backpropagation algorithm. This makes it easier for us to train networks with many layers in comparison to the use of a sigmoid activation function, where the derivatives are never equal to 1. Using both a ReLU activation function and bias parameter allows us to set the activation of any node to zero when the weighted linear combination is below the value of the bias parameter.

Likewise, the activation value of the m^{th} node of the l^{th} layer for the h^{th} datapoint is

calculated as:

$$\text{act}_{h,m}^{(l)} = u(b_{0,m}^{(l)} + \sum_{j=1}^{M_{l-1}} \text{act}_{h,j}^{(l-1)} b_{j\to m}^{(l)}) \tag{2.8}$$

where $b_{0\to m}^{(l)}, ..., b_{M_{l-1}\to m}^{(l)} \in \mathbb{R}$ are learnable parameters and $\text{act}_{h,m}^{(l)} \in \mathbb{R}$.

There are L layers, and on the l^{th} layer, there are M_l nodes.

After successive non-linear transformations from several layers, we obtain the vector of features describing the h^{th} datapoint $\boldsymbol{s}_h$ from the activations of the last layer after inputting the frame sample $\boldsymbol{x}_{*,h}$:

$$\boldsymbol{s}_{h_m} \leftarrow \text{act}_{h,m}^{(L)} \qquad m \in 1, ..., M_L \tag{2.9}$$

2.6.2 Calculating covariates using convolutional neural network

Convolutional Neural Networks, or CNNs (LeCun et al., 1995) are specialised neural networks for processing data that has a grid-like topology. Application examples include time series data (with kernels convoluted along the time dimension) and image data (where 2D kernels are convoluted along the width and length of an image). CNNs consist of a series of layers, typically one or several convolutional layers followed by some feedforward layers (discussed in Section 2.6.1). In this section, we will give an account of each of three computation steps in convolutional layers for multidimensional time series data (see Figure 2.61 on steps in convolutional layer), and the motivation for each of them.

Convolution operator

To motivate the convolutional stage of a convolutional layer (as labelled 'A' in Figure 2.6.1), we first introduce the convolutional operation. The convolutional operator on a

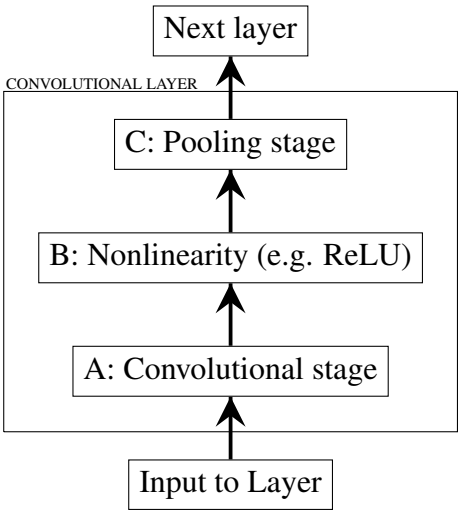


Figure 2.61: Overview of computation steps in convolutional layer of Convolutional Neural Network.

continuous time function Γ_{cts}, with a weighting function ω_{cts} is given by:

$$(\Gamma_{cts} * \omega_{cts})(t) = \int \Gamma_{cts}(\chi)^T \omega_{cts}(t - \chi) d\chi \tag{2.10}$$

where $t \in \mathbb{R}$, and $\Gamma_{cts}(t), \omega_{cts}(t) \in \mathbb{R}^d$. $\Gamma_{cts}(\chi)^T$ is the transpose of $\Gamma_{cts}(\chi)$.

The convolution operator's output at any point $t \in \mathbb{R}$ is the integral of Γ_{cts} times the kernel function ω_{cts} centred on the point t. In this way, the convolution's output at t gives us a measure of how well the function Γ_{cts} correlates with the kernel function around the timepoint t.

The analogous discrete operator is:

$$(\Gamma_{discrete} * \omega_{discrete})(t) = \sum_{\chi=-\infty}^{\infty} \Gamma_{discrete}(\chi)^T \omega_{discrete}(t - \chi) \tag{2.11}$$

In CNN terminology the first argument to the above equations (Γ_{cts} or $\Gamma_{discrete}$) would be termed the *input function* and the second (ω_{cts} or $\omega_{discrete}$) the *kernel function*. Where we access values outside the domain of $\Gamma_{discrete}$ where $\Gamma_{discrete}$ is undefined, we may assume that the value of the function is zero. This is called *zero padding*.

First step in convolutional layer: Convolution

For our application of CNNs to multivariate time series data, we use the notation defined in Section 2.1 for sliding window data with internal time ordering preserved in matrix form as $\boldsymbol{X}_{*,h}$.

Then, the calculation of the *convolutional feature map*, the convolutional stage of the first convolutional layer, for the k^{th} kernel is given by:

$$\mathrm{act}_{h,m}^{(1,k)} = u(b_{0,k}^{(1)} + (\boldsymbol{X}_{*,h} * \mathrm{Kern}_{l,k})(m)), \qquad \mathrm{Kern}_{l,k} \in \mathbb{R}^{d \times r_l}, \boldsymbol{X}_{*,h}(m) \in \mathbb{R}^d \quad (2.12)$$

$$1 \leq l \leq \#conv{-}layers, \quad 1 \leq k \leq \#Kern_l, \quad 1 \leq m \leq r_l, \quad 1 \leq h \leq N_*$$

where the number of convolutional layers is denoted as $\#conv{-}layers$, the number of kernels on the l^{th} convolutional layer is denoted $\#Kern_l$, the width (in number of time samples) of the kernels on the l^{th} layer is denoted r_l, and the number of sliding windows in the dataset considered is N_*.

A diagrammatic illustration of this is given in Figure 2.62, where $\mathrm{Kern}_{l,k}$ is the k^{th} kernel from the l^{th} convolutional layer, and is of width r_l along the time axis. Later, we denote the number of kernels used on the l^{th} convolutional layer as $\#\mathrm{Kern}_l$.

Motivations for convolutional stage

There are three motivations for using the convolutional stage in a neural network, in comparison to a fully connected feedforward layer (FCFFL) like described in Section 2.6.1 (Goodfellow et al., 2016). Firstly, convolutional layers benefit from *sparse interactions*, that is there are far less parameters to compute and store on convolutional layers compared

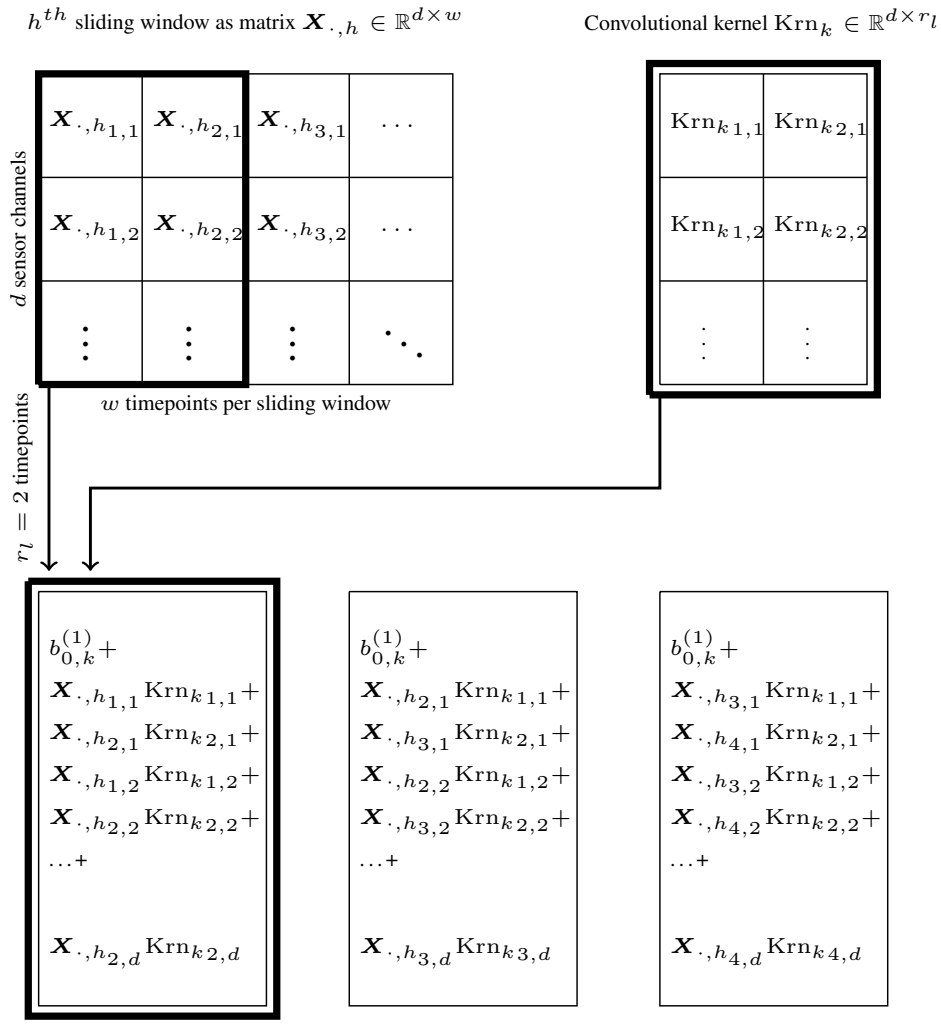


Figure 2.62: An overview of the convolution operation on the convolutional stage in convolutional layers. In this example, the kernel width r_l is 2.

to fully-connected layers, especially when the convolutional kernel width is much shorter than the length of input datapoints. Figure 2.63 gives a comparison between the number of connections (and therefore parameters) on convolutional compared to feedforward layers.

A second motivation is *parameter sharing*, because rather than each parameter being used in only one location in the input vectors in FCFFL, convolutional layers use each learned parameter at every single location (element) of the inputted object (i.e. sliding window $\boldsymbol{X}_{*,h}$). Therefore, parameter sharing is drastically more efficient in terms of storage requirements compared to FCFFL.

A third desirable feature is *equivalence to translation*, that is if some feature is seen later in an input object, then the resultant features will be the same , and simply appear in the corresponding part of the output of the convolutional layer.

Second step in convolutional layer: Nonlinearity

The second step in the convolutional layer (as shown as Part 'B' in Figure 2.61) is the nonlinearity (e.g. rectified linear unit). The motivation relating to the Universal Approximation Theorem is discussed in Section 2.6.1.

Third step in convolutional layer: Pooling

The third and final step is *pooling* (part 'C' in Figure 2.61), most commonly max pooling. This replaces the output of the latter at regular intervals (every Δ_l elements) with a summary statistic of nearly outputs from the previous step. This is shown diagrammatically in Figure 2.64, and is defined formulaically as:

$$\text{act}_{h,m}^{(l,k)\ NEW} \leftarrow \max_{\Delta_l \lfloor \frac{m}{\Delta_l} \rfloor \leq j < \Delta_l \lfloor \frac{m}{\Delta_l} \rfloor + m} \text{act}_{h,m}^{(l,k)} \tag{2.13}$$

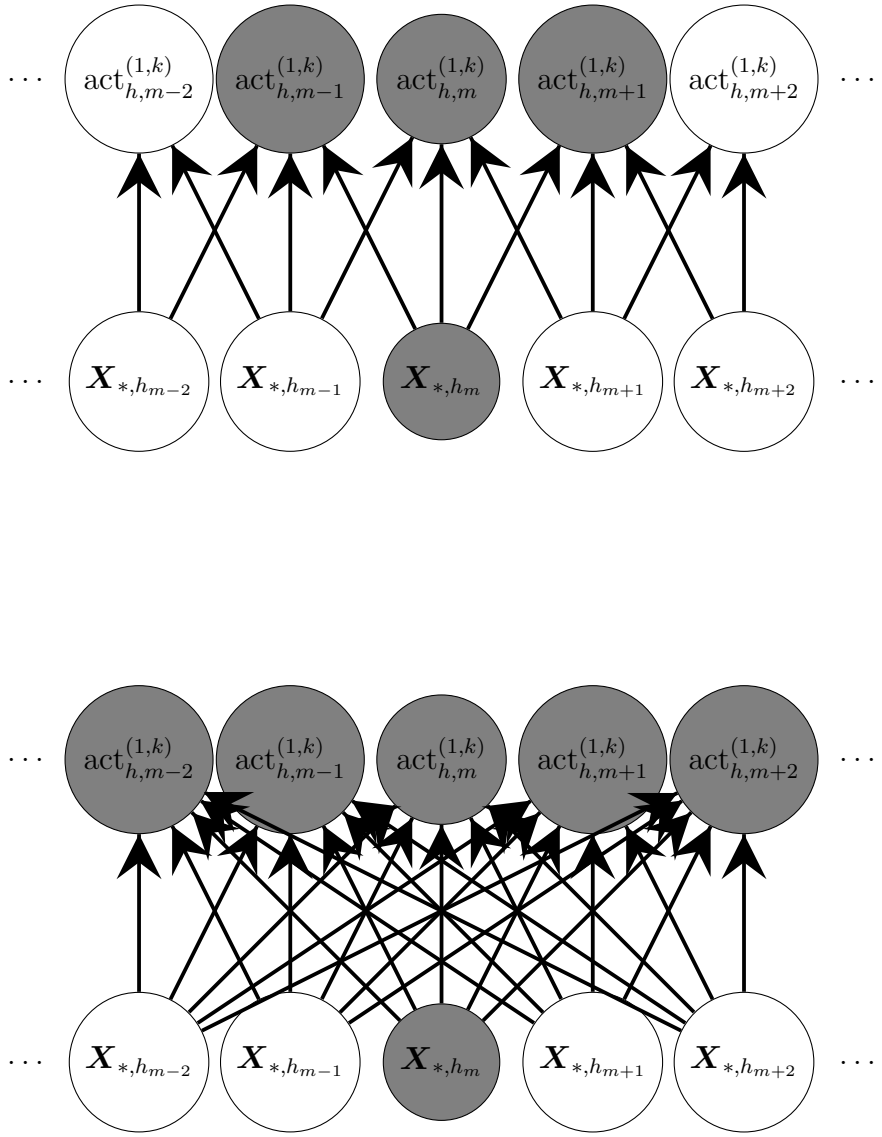


Figure 2.63: Illustration of sparsity of parameters in a convolutional layer with kernel width $r_l = 3$ (top) compared to feedforward layers (lower). In convolutional layers, the value of each element only affects a small number of output nodes, so therefore the layer requires less parameters.

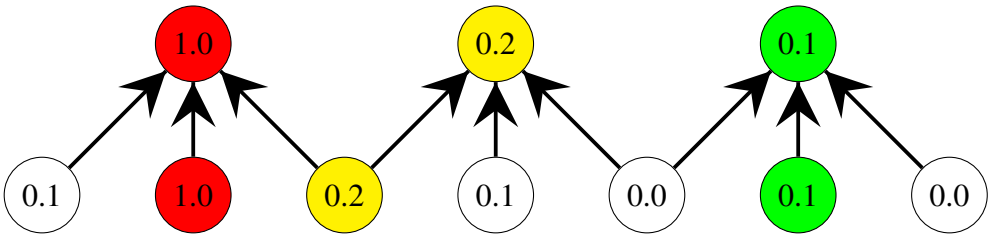


Figure 2.64: Illustration of max pooling with pooling region width $\Delta_l = 3$.

For the second and subsequent convolutional layers, then the output of the previous convolutional layers is used as input to the next, and the outputs from the next layers are computed in a similar manner.

Typically in a CNN model, there may be one or more convolutional layers, before some feedforward layers, to finally arrive at a feature vector $\boldsymbol{s}_h$, the usage of which we will discuss in Chapter 3.

2.6.3 Calculating covariates using recurrent neural network

Another approach to modelling time series data is to use state-space models (SSMs). In SSMs, a hidden state vector is calculated at each discrete timestep, based on hidden state values in the previous timestep, and input values (e.g. accelerometer data) in the current timestep (Goodfellow et al., 2016).

A useful advantage of these types of models with 'hidden states' for accelerometer data is that we may not need to use sliding windows at all (for example $\boldsymbol{x}_{*,h}$), as the models can operate on the discrete timesteps corresponding to raw high-frequency accelerometer data (for example $\boldsymbol{a}_i$).

A basic RNN (recurrent neural network) can be described as a state space model, with discrete timesteps, where the hidden state at every timestep is updated based on an affine transformation of the previous hidden state and some input vector containing acceleromter data, before a non-linear (elementwise) transformation is applied:

$$\mathbf{hidden}_i = u(\boldsymbol{W}_{hx}\boldsymbol{a}_i + \boldsymbol{W}_{hh}\mathbf{hidden}_{i-1} + \boldsymbol{b}_h), \quad i \leq (TR) + 1 \tag{2.14}$$

where $\boldsymbol{W}_{hx} \in \mathbb{R}^{d\times m}$, $\boldsymbol{W}_{hh} \in \mathbb{R}^{m\times m}$, $\boldsymbol{b}_h \in \mathbb{R}^m$ are learnable parameters, and u is a non-linear transformation such as defined in Equation (2.6).

Then, the predicted class probabilities of the current timestep can be computed using the multinomial model discussed in Section 3.A.

The model is trained by backpropagation, taking the derivative of the output with respect to each parameter and optimizing through a gradient descent procedure which we will discuss in Section 2.6.5. Additionally, Backpropagation Through Time (BPTT) is commonly used, where prediction errors are propagated back through the sequence of the input time series (Werbos et al., 1990; Graves, 2012).

This model, however, often results in hidden states which are not useful because of the multiplicative nature Equation (2.14). Since we are repeatedly multiplying the hidden vector on each timestep by the transition matrix $\boldsymbol{W}_{hh}$, then magnitudes of these values will tend to explode (tend toward infinity) or vanish (tend toward zero) over time. It is not possible for a hidden state to stay approximately constant for a long-term period, while also having the flexibility to change to differing values later when context changes.

Long Short-Term Memory (LSTM) network

It might be useful for our models to *forget* its hidden state (either completely or in part by some weight). Long Short-Term Memory (LSTM) neural networks allow us to do this. The hidden state is considered 'short-term memory' memory, but the LSTM network has the ability to retain this state for a dynamic, potentially long number of timesteps (hence the name Long Short-Term Memory) (Gers et al., 2000; Goodfellow et al., 2016).

The motivation for LSTM is apparent in modelling accelerometer data, as previous

accelerometer data and activities of a human subject are highly relevant in determining what activity (class) they are likely to be currently undertaking, even before taking into account current and recent accelerometer data. Additionally, context can stay constant for a long amount of time and then suddenly change, which we will discuss in this section.

The key principle of LSTM networks is to have a linear self-loop which is *gated* (controlled by another unit called the *forget gate*). Self-loop refers to each element of the current timestep only affecting the same element in the next timestep. By gating the self-loop (which determines whether and how much the network should carry over memory in each step), then the time scales over which internal state are held can be changed dynamically.

Instead of a unit that simply applies a non-linearity to a linear transformation of inputs and previous state (as in Equation (2.14)), LSTM networks have 'LSTM cells' that have an internal recurrence (a self-loop), in addition to an outer recurrence of the RNN (we see the outer recurrence in Equation 2.14, where each element of the hidden state vector is directly affected by all of the elements of the previous timestep). Each cell has the same inputs and outputs as an ordinary RNN, but has more parameters and a system of gating units that controls the flow of information. The most important is the state unit $\mathbf{memory}_i$, which has a linear self-loop. The self-loop weight is controlled by an elementwise *forget gate* unit $\mathbf{forget}_i$, which always has a value between 0 and 1 due to use of a sigmoid non-linearity:

$$\begin{gathered}\mathbf{forget}_i = \text{sigmoid}(\boldsymbol{W}_{fx}\boldsymbol{a}_i + \boldsymbol{W}_{fH}\mathbf{hidden}_{i-1} + \boldsymbol{b}_{forget}) \\ \mathbf{forget}_i, \boldsymbol{b}_{forget}, \mathbf{hidden}_{i-1} \in \mathbb{R}^m, \boldsymbol{W}_{fx} \in \mathbb{R}^{m\times d}, \boldsymbol{W}_{fH} \in \mathbb{R}^{m\times m}\end{gathered} \tag{2.15}$$

where the sigmoid function for any real value is defined as:

$$\text{sigmoid}(\cdot) = \frac{\exp(\cdot)}{1 + \exp(\cdot)} \tag{2.16}$$

and is applied elementwise for vector inputs.

We see where the forget gate is used in a self loop inside the first parenthesis of the following equation. We see that the previous LSTM memory cell state $\mathbf{memory}_{i-1}$ is carried over to the current timestep using the elemetwise multiplication operator $\circ$ with $\mathbf{forget}_i$:

$$\begin{aligned} \mathbf{memory}_i = (\mathbf{memory}_{i-1} \circ \mathbf{forget}_i) + (\mathbf{cell}_i \circ \mathbf{input}_i) \\ \mathbf{memory}_i, \mathbf{cell}_i, \mathbf{input}_i \in \mathbb{R}^m \end{aligned} \tag{2.17}$$

The term in the second parenthesis of Equation (2.17) refers to $\mathbf{cell}_i$. $\mathbf{cell}_i$ is the 'outer loop' of the LSTM cell, since the value of this vector on each timestep depends on all of the elements in the vector $\mathbf{hidden}_i$ in the previous timestep:

$$\begin{aligned} \mathbf{cell}_i = u(\boldsymbol{W}_{cx}\boldsymbol{a}_i + \boldsymbol{W}_{cH}\mathbf{hidden}_{i-1} + \boldsymbol{b}_{cell}) \\ \mathbf{cell}_i, \boldsymbol{b}_{cell} \in \mathbb{R}^m . \boldsymbol{W}_{cx} \in \mathbb{R}^{m \times d}, \boldsymbol{W}_{cH} \in \mathbb{R}^{m \times m} \end{aligned} \tag{2.18}$$

The value of $\mathbf{cell}_i$ is *gated* (elementwise multiplied by to create a weighting between zero and one) in the second parenthesis of Equation (2.17) by the *input gate* $\mathbf{input}_i$. The value of the input gate at each timestep is calculated the same way as the forget gate, but with its own parameters:

$$\begin{aligned} \mathbf{input}_i = u(\boldsymbol{W}_{Ix}\boldsymbol{a}_i + \boldsymbol{W}_{IH}\mathbf{hidden}_{i-1} + \boldsymbol{b}_{input}) \\ \mathbf{input}_i, \boldsymbol{b}_{input} \in \mathbb{R}^m . \boldsymbol{W}_{Ix} \in \mathbb{R}^{m \times d}, \boldsymbol{W}_{IH} \in \mathbb{R}^{m \times m} \end{aligned} \tag{2.19}$$

The external output of the LSTM network,$\mathbf{hidden}_i$, is based on gating the memory state of the network as calculated in Equation (2.17):

$$\mathbf{hidden}_i = \tanh(\mathbf{memory}_i) \circ \mathbf{output}_i \tag{2.20}$$

where the values of the *output gate* are calculated in the same way as the forget and input gates, but with different parameters:

$$\begin{gathered}\mathbf{output}_i = u(\boldsymbol{W}_{ox}\boldsymbol{a}_i + \boldsymbol{W}_{oH}\mathbf{hidden}_{i-1} + \boldsymbol{b}_{output}) \\ \mathbf{output}_i, \boldsymbol{b}_{output} \in \mathbb{R}^m, \boldsymbol{W}_{ox} \in \mathbb{R}^{m\times d}, \boldsymbol{W}_{oH} \in \mathbb{R}^{m\times m}\end{gathered} \tag{2.21}$$

A prediction of the class probability (in this case which activity a human subject is carrying out) can be made by using a logistic model (described in Appendix 3.A) with the vector $\mathbf{hidden}_i$ as the input.

Because the rates of forgetting of the hidden state are learnable functions based on the previous hidden state and current inputs, rather than simply based on static state transition matrices, the model is capable of holding state information for longer and varying amount of time. For example, an element of the hidden state vector could hold one value, until such time as the observation vector $\boldsymbol{a}_i$ is a certain value related to a critical event which changes all the predictions later in the time series. One situation when this may occur is if the transition matrix $\boldsymbol{W}_{cx}$ is close to an identity matrix, all elements of the input gate vector are close to 0, and all elements of the forget gate vector are close to 1, until $\boldsymbol{a}_i$ reaches a critical value, and then elements of the input gate become non-zero and the input data after the i^{th} timestep start to have an effect on the hidden state vector (Goodfellow et al., 2016).

2.6.4 Bidirectional recurrent neural network models

Where we want to carry out an inference at timestep i, based on what has come before *and after* timestep i, we can use bidirectional neural networks (b-LSTM-s) (Graves and Schmidhuber, 2005). In this model, the time series is split into sequences of length

N_{seq} which occur one after another, and an LSTM recurrent neural network (as described above in Section 2.6.3), and is fed the sequence of observations from the startpoint of the sequence which timestep i belongs to (for instance time $t_{SeqStart}$) until the end of the sequence timestep i belongs to $t_{SeqStart} + N_{seq}$, before being fed the sequence in reverse, that is ($\boldsymbol{a}_{t_{SeqStart}}, ..., \boldsymbol{a}_i, ..., \boldsymbol{a}_{t_{SeqStart}+N_{seq}}, ..., \boldsymbol{a}_i, ..., \boldsymbol{a}_{t_{SeqStart}}$). The corresponding data labels used in training are ($c_{t_{SeqStart}}, ..., c_i, ..., c_{t_{SeqStart}+N_{seq}}, ..., c_i, ..., c_{t_{SeqStart}}$). Then, the hidden states for timestep i for both the forward and reverse passes are used as covariates for prediction at timestep i.

2.6.5 Parameter estimation approaches for neural network models

In this section, we discuss some possible approaches to parameter estimation for the neural network model discussed in the previous sections of this chapter, as well as approaches we used in our deep learning exploration experiments in greater detail. We also discuss approaches to regularization for improving the generalisability of neural network models.

From a random initialization of model parameters — based on some random samples from a Gaussian distribution such as based on Xavier or Glorot initialization as discussed in Sutskever et al. (2013) — gradient descent attempts to maximize the likelihood of the model we are fitting (for example in supervised learning classification we may wish to maximize the likelihood of the multinomial distribution of all the classes for all the datapoints, given in Equation (3.1)). We calculate the gradient with respect to each parameter in the model, where the gradient of the parameter denoting the influence which the jth node on the $l-1$th layer has on the mth node on the lth layer is denoted as $\nabla\mathcal{L}(b^{(l)}_{j\to m})$ on the τth iteration of this gradient descent algorithm.

Then, we update the weights according to the Adagrad rule:

$$b^{(l)}_{j\to m_{\tau+1}} = b^{(l)}_{j\to m_\tau} - \frac{\delta\nabla\mathcal{L}(b^{(l)}_{j\to m_\tau})}{\sqrt{\varrho^{(l)}_{j\to m_{\tau+1}}} + \varepsilon} \tag{2.22}$$

where $\varepsilon > 0$ is a small numerical constant for numerical precision and

$$\varrho^{(l)}_{j \to m_{\tau+1}} = \varrho^{(l)}_{j \to m_{\tau}} + \nabla \mathcal{L}(b^{(l)}_{j \to m_{\tau}})^2 \tag{2.23}$$

Equation (2.22) is the standard gradient descent equation with learning rate $\delta \in (0, 1]$, where the learning rate is divided by a value based on Equation (2.23). This number increases the weight update for parameters which previously had small weight updates, thus making it easier to tune the weights of parameters in parts of the model which are infrequently activated, for example in the case of datapoints of a rare class. Some other similar learning rate rules are given in Adam (Kingma and Ba, 2014) and RMSProp (Ruder, 2016). Learning rate decay allows us to reduce the learning rate by a factor of $\vartheta \in (0, 1]$ on each successive iteration of our backpropagation algorithm through our dataset. This allows us to be less dependent on a particular choice of learning rate, as the effective learning rate $\delta\vartheta^{\tau}$ can range through different orders of magnitude as our learning procedure progresses. Then, our update equation becomes:

$$b^{(l)}_{j \to m_{\tau+1}} = b^{(l)}_{j \to m_{\tau}} - \frac{\delta \vartheta^{\tau} \nabla \mathcal{L}(b^{(l)}_{j \to m_{\tau}})}{\sqrt{\varrho^{(l)}_{j \to m_{\tau+1}}} + \varepsilon} \tag{2.24}$$

We use minibatch stochastic gradient descent (SGD) when training. This is where we carry out backpropagation using batches of 64 frames at a time. SGD calculates the gradient using subsets of the available training data called batches (Zhang, 2004). This is as opposed to batch gradient descent which calculates gradient descent for the entire dataset before calculating the new weights of the network using backpropagation. Usage of random subsets of the data introduces randomness, which helps avoid local minima in the loss function, since each minibatch will have a different effect on the model's parameters.

2.7 Principal Component Analysis

Once some features based on sliding windows are chosen, there may be too many dimensions in the data, hindering the ability to perform inference. Principal Components Analysis (PCA) is a method of finding a set of basis vectors which span an input space, and for which act as linear transformations into a reduced-dimensionality space, and where these basis vectors capture as much of the variability in the input data as possible (Jolliffe, 1986). Suppose we have a dataset in a q-dimensional space, with a covariance matrix $\mathbf{\Psi} \in \mathbb{R}^{q \times q}$. Then, the q-dimensional linear combination of the input variables which maximizes the variance of the transformed data is given by:

$$\underset{\boldsymbol{Q}_1 \in R^q}{max} \ \boldsymbol{Q}_1^T \mathbf{\Psi} \boldsymbol{Q}_1$$

The optimal vector $\boldsymbol{Q}_1 \in \mathbb{R}^q$ is given by the leading eigenvector of the covariance matrix $\mathbf{\Psi}$, where the leading eigenvector is defined as the one with the highest eigenvalue. Likewise, the next most important linear combination vectors which explain variability can be computed using the remaining eigenvectors, in descending order of their eigenvalues. The benefit of using this approach is that we are able to effectively reduce the dimensionality of our data, while being able to inspect the resulting principal components for their possible meaning.

2.7.1 Functional PCA

Functional PCA is similar to the approach we have discussed, but is more suitable for the case of where we observe functions with possible sensing measurement errors which are assumed to be independent and identically distributed at each time t_i seconds after the start of the signal, and have variance governed by a smooth (for example constant) function $\sigma^2(t)$, for example representing sensor measurement error.

We will discuss and example where sliding windows are treated as realisations of functions. Suppose we have N_* realisations $\boldsymbol{v}_{*,1}, ..., \boldsymbol{v}_{*,N_*}$ of a function $\boldsymbol{v}$ sampled on a regular grid at w points, which is composed of the one-dimensional Signal Vector Magnitude of movement (i.e. in this case $d = 1$):

$$\boldsymbol{v}_{*,h} = (SVM(\boldsymbol{a}(t_1^{(h)})), ..., SVM(\boldsymbol{a}(t_w^{(h)})))^T \tag{2.25}$$

where SVM is defined in Equation (2.2), and $t_i^{(h)} \in [0, T)$ is the real-valued timepoint associated with the i^{th} element of the h^{th} sliding window. Because the SVM function outputs real-valued numbers, then $\boldsymbol{v}_{*,h} \in \mathbb{R}^w$. If we assume that there is an underlying physical process with an associated underlying physical signal vector magnitude value $SVM^{PHYS}(t) \in \mathbb{R}$, and a sensor error of $\sigma^2(t) \in \mathbb{R}$ which has a distribution which is smooth or constant over time, then:

$$\boldsymbol{v}_{*,h} = (SVM^{PHYS}(\boldsymbol{a}(t_1^{(h)})) + \sigma^2(t_1^{(h)}), ..., SVM^{PHYS}(\boldsymbol{a}(t_w^{(h)})) + \sigma^2(t_w^{(h)}))^T \tag{2.26}$$

Before carrying out the PCA procedure as discussed before, we construct a covariance matrix $\Psi^{w \times w}$ (taking into account that in this case $d = 1$). This matrix contains the covariances of each point on the regular sensing grid (i.e. the correlations between the i^{th} and j^{th} elements of each realisation $\boldsymbol{v}_{*,h}$). A smoother is applied to the matrix Ψ to account for the effect of the error terms given by the function $\sigma^2(t)$, before the previously-discussed PCA steps are carried out (Wang et al., 2016; Ramsay and Silverman, 2007).

Functional PCA can visualise the important features of large colelctions of similar curves (Jones and Rice, 1992).

2.8 Conclusion

In this chapter, we discussed the various ways of extracting features from sliding windows of accelerometer data: statistical, frequency, symbolic and neural network features (including parameter estimation approaches in neural network models), as well as the usage of dimensionality reduction and mixture models on data. These methods are used extensively in the next chapters: the usage of neural network features for supervised learning is discussed in Chapter 3, while the usage of dimensionality reduction and mixture models for the automated assessment of recovery levels of patients with stroke is discussed in Chapter 5.

Chapter 3

Supervised Learning for Activity Recognition

In this chapter, we discuss Human Activity Recognition (HAR), supervised learning classification models used for HAR, as well as an exploration of deep neural network models for HAR applications. Later in this chapter, we also provide an exploration of the predictive accuracy of some neural network models including feedforward (referred to as deep neural networks or DNN, discussed in Section 2.6.1), convolutional (CNN outlined in Section 2.6.2) and more recent recurrent neural network models. The recurrent neural network models include sliding window (frame)-based LSTM (referred to as LSTM-F), a sample-wise LSTM where non-sliding window data is input (referred to as LSTM-s, as outlined in Section **??**) and bidirectional LSTM (referred to as b-LSTM-s, outlined in Section 2.6.4) on some benchmark HAR datasets. To the best of our knowledge, this is the first work which includes an extensive hyperparameter search for the construction of neural network models for accelerometer-based HAR, with insights of which types of hyperparameters are most important to pay attention to when constructing a neural network for a new HAR application. HAR, like the remote stroke rehabilitation discussed in later chapters, makes use of the feature extraction approaches discussed in Chapter 2.

In this chapter we focus our interests on inferring which human activity, from a set of p pre-selected classes (e.g. walking, sitting, etc), a user is carrying out while they are wearing the body-worn accelerometer device, at any given time. The neural network based covariates may also be used in prediction of stroke rehabilitation, see the discussion in Section 6.3.

This chapter is structured as follows. In Section 3.1, we give an account of the task of classification in supervised learning for accelerometer data, how the performance of models is evaluated for that task. In Section 3.2 we give an outline of neural network hyperparameter choices. In Section 3.3 we give an account of experiments we performed to investigate the effect of model and hyperparameter choice on model performance. In Section 3.4 we give provide the results of our exploratory experiments and in Section 3.5 we give a discussion and conclusion on the outcomes of our experiments.

3.1 Supervised Learning for Accelerometer Data

Given an unforeseen vector time series of accelerometer data $(\boldsymbol{a}_1, ..., \boldsymbol{a}_T)$, HAR aims to assign probabilities that a human subject (e.g. a patient) is carrying out each of p classes of activity for each timepoint (e.g. by using sliding windows). To achieve this, raw time series data could be preprocessed and features extracted using some of the approaches outlined in Chapter 2. Then, a learnable function could map the preprocessed sliding window datapoints to probabilities predictive of the subjects current activity class. Some possible supervised learning models suitable for this purpose, given training time series from patients, could include a multinomial logistic model using covariates from either the deep feedforward (referred to as DNN later in this chapter), convolutional (referred to as CNN) or recurrent neural networks (referred to as LSTM-F later in this chapter) discussed in Chapter 2. We note that also it is also possible to use the recurrent neural network models to infer directly an associated activity label for each discrete input timepoint (sample)

of raw accelerometer data, and we have explored this model configuration (referred to as LSTM-s) later in this chapter.

3.1.1 Multinomial model

To construct the prediction function using supervised learning, we consider a model for the activity class c_h^i the i^{th} subject is undertaking as being multinomially-distributed with one trial at each time index h. Let $\boldsymbol{\rho}_h^i = (\rho_{h,1}^i, ..., \rho_{h,p}^i)$ represent the inferred probabilities of each subject undertaking each class. The log-likelihood of a multinomial distribution over all of the data from n subjects, with N_i datapoints from the i^{th} subject, is:

$$\ell(c_1^1, ..., c_{N_*}^n | \boldsymbol{\rho}_1^1, ..., \boldsymbol{\rho}_{N_*}^n) = \sum_{i=1}^{n} \sum_{h=1}^{N_i} \sum_{m=1}^{p} I(c_h^i = m) \ln(\rho_{h,m}^i) \tag{3.1}$$

where N_* denotes the total number of datapoints from all patients, and $I(\cdot)$ is an indicator function which takes the value of 1 if the statement in its argument is true and 0 otherwise. The derivation of this log-likelihood, and its use with the feature vectors from Section 2.6 are given in Appendix 3.A at the end of this chapter.

To fit our supervised learning classification algorithm, we maximize the likelihood in Equation (3.1) through the gradient descent procedure discussed in Section 2.6.5. Approaches for model evaluation are discussed in Section 3.1.2, model hyperparameter choices are outlined in Section 3.2, and the effect of the model hyperparameter choices is explored in Section 3.3.

3.1.2 Model evaluation

To find the optimal parameters for our neural network models (discussed in Section 2.6), we can employ an iterative gradient descent procedure to maximize the likelihood of the model, which we will discuss in Section 2.6.5.

We will want to assess the accuracy of our model, both after the inference procedure to assess the quality of fit on an unseen test set, and during the inference procedure to assess when we have met the necessary conditions to halt the model fitting. A commonly used metric in HAR and other classification tasks with multiple classification categories is called the *mean F1 score* (Bulling et al., 2014). It takes into account the condition where the ability of the model to detect if the model is overestimating the probability of the dominant class, for example by classifying all datapoints as being part of the dominant class. In this metric, the F_1 score of each of p classes is calculated, and then they are averaged:

$$\text{Mean F}_1\text{ Score} = \frac{1}{p}\sum_{m=1}^{p} F_1(\text{Class } m) \tag{3.2}$$

where the F_1 score for the m^{th} class is calculated as:

$$F_1(\text{Class } m) = 2 \times \left(\frac{\text{Precision}(\text{Class } m) \times \text{Recall}(\text{Class } m)}{\text{Precision}(\text{Class } m) + \text{Recall}(\text{Class } m)}\right) \tag{3.3}$$

where the precision is the proportion of datapoints classified as belonging to class m that are actually part of class m:

$$\text{Precision}(\text{Class } m) = \frac{\#\text{True Positives for Class } m}{\#\text{True Positives for Class } m + \#\text{False Positives for Class } m} \tag{3.4}$$

and recall is the proportion of datapoints which are actually in class m that are correctly classified as being part of that class:

$$\text{Recall}(\text{Class } m) = \frac{\#\text{True Positives for Class } m}{\#\text{True Positives for Class } m + \#\text{False Negatives for Class } m} \tag{3.5}$$

Similarly, the *Weighted F_1 score* is an average of the F_1 scores over all the classes, weighted by how prevalent each class is:

$$\begin{aligned}\text{Weighted } \mathrm{F}_1 \text{ Score} = \\ \frac{1}{\sum_{m=1}^{p} \text{Instances of class } m} \\ \times \sum_{m=1}^{p} \text{Instances of class } m \times F_1(\text{Class } m)\end{aligned} \tag{3.6}$$

In Section 2.6.5, we discuss parameter estimation approaches, and later we use these performance metrics and parameter estimation approaches in experiments to determine which configuration of neural networks is best for various typical HAR tasks.

3.2 Neural network hyperparameters

When constructing a neural network model for HAR, it is necessary to decide both on the type of network (i.e. DNN, CNN, LSTM-s, LSTM-F or b-LSTM-s, all of which are outlined in Section 2.6) and other choices related to learning scheme (parameter estimation), regularization and architecture (e.g. model size). The values of these choices are termed *hyperparameters*. In this section, we outline the main hyperparameter choices under the headings *learning*, *regularization* and *architecture*. Then for the rest of this chapter, we give an account of an exploration of the effect of all these hyperparameter values on predictive performance on three popular benchmark HAR accelerometer datasets, where the ranges of hyperparameters explored are shown in Table 3.31.

3.2.1 Learning-related hyperparameters

The first hyperparameter choice shown in Table 3.31, and one which we found to be crucial in our experiments later in this chapter, is *learning rate*. It determines the extent to which the model's parameters should change in response to each training data sample or minibatch. If the learning rate is too high, then the gradient descent-based algorithms will not succeed in finding good optimal parameter solutions, as each successive change in parameters may overshoot local minima in the loss function. If the learning rate is too low, then the model will learn very slowly (require a large number of training iterations to attain a good performance). It is defined in Section 2.6.5 and is denoted in this thesis with δ. In most applications of neural networks, practitioners experiment with a range of learning rates which vary by orders of magnitude, so for our experiments we randomly selected the learning rate to use in each experiment from a log-uniform distribution.

A closely-related hyperparameter which is also defined in Section 2.6.5 is *learning rate decay*. It determines the rate at which we decay the learning rate on each training iteration through the dataset, allowing us to effectively use a range of learning rates. It is denoted as ϑ in this thesis, and is shown on the second column of Table 3.31. For similar reasons to the previously-discussed hyperparameter, we also randomly select the value of learning rate decay for each experiment from a log-uniform distribution.

For LSTM recurrent neural networks (i.e. LSTM-F, LSTM-s and b-LSTM-s), the *length of training sequences*, is also a learning-related hyperparameter. To train these models, the dataset must be split into non-overlapping subsequences which are assumed to be independent of one another. If the length of these subsequences L_{seq} is too short, then it is impossible in this application to capture and predict with crucial contextual aspects as the subsequence will not capture important things which happened in the recent past (for example if a fridge door has just been opened, it is likely to be closed again soon). If a training subsequence is too long, the model could suffer from overfitting and poor

learning. This model choice is denoted as L_{seq} and is shown in the third column of Table 3.31.

3.2.2 Regularisation-related hyperparameters

Regularisation techniques are required to learn simpler models, so that the models generalize well to unseen test sets as opposed to overfit to the training dataset used. We used the approaches such as momentum, max-in-norm, p-carry, and dropout.

Momentum (Sutskever et al., 2013) refers to replacing the gradient in Equation (2.22) with a weighted moving average of the current and past gradients over previous iterations, with hyperparameter $\gamma \in (0, 1]$. It encourages regularization by preventing the learned parameters from being changed too much based on any one minibatch:

$$b^{(l)}_{j\to m_{\tau+1}} = b^{(l)}_{j\to m_{\tau}} - \frac{\delta\vartheta^{\tau}\sum_{k=0}^{\tau}\gamma^{\tau-k}\nabla\mathcal{L}(b^{(l)}_{j\to m_{\tau}})}{\sqrt{\varrho^{(l)}_{j\to m_{\tau+1}}} + \varepsilon} \tag{3.7}$$

The ranges of momentum we tried in our experiments is shown in Table 3.31.

Dropout (Srivastava et al., 2014) is a form of model regularization for neural networks. It is designed to be an efficient way to implicitly emulate bagging (bootstrapped aggregation) in estimating model parameters. Bagging is a way of combining many high-variance, low bias expert models for parameter estimation through model averaging. In dropout, a proportion of neurons in the network are randomly zeroed out in the training phase. In the feedforward phase of training, for each pass, the network acts like a component of a mixture-of-experts system. Model parameters are updated through backpropagation based on the subset of parameters which are activated. In the test or application phases, we can multiply the parameters by the reciprocal of the dropout percentage to obtain the average of all the subnetworks in the training phase. In this way, we are implicitly model averaging over many different models. To limit the number of dimensions

in our hyperparameter search, we chose not to explore the effect of this hyperparameter on model performance.

Max-in-norm refers to scaling the incoming weights to each node in the neural network to have a maximum Euclidean length. It is suggested that this approach performs well when dropout is used (Srivastava et al., 2014). The range of values for this hyperparameter is shown in the fifth column of Table 3.31.

p_{carry} is the hyperparameter associated with a regularisation technique for recurrent (e.g. LSTM-based) neural network models which to the best of our knowledge has not been employed in other works before. In the recurrent neural network models (LSTM-F, LSTM-s and b-LSTM-s), we carry over the internal states of the model (given by the hidden state $\mathbf{hidden}_{t-1}$ and memory cell $\mathbf{memory}_{t-1}$) with probability p_{carry} between consecutive minibatch subsequences. The length of training subsequences denoted by L_{seq} is another closely-related hyperparameter which we explored and discussed in Section 3.2.1.

3.2.3 Architecture-related hyperparameters

Architecture-related hyperparameters govern the number of parameters in the neural network model. The more parameters a model has, the more likely that the model will overfit to the training dataset used and not generalize well to testing datasets, unless regularization is employed. However, if a model doesn't have a sufficient number of parameters, it may not be expressive enough to predict the human activity classes well.

For both DNN models (which are composed of hidden fully-connected feedforward layers) and CNN models (which are composed of some hidden fully-connected feedforward layers which process the outputs of some convolutional layers), the number of hidden feedforward layers (denoted as L in Section 2.6.1) has an obvious effect on the number of parameters in the model, as each hidden feedforward layer has M hidden

nodes, and for each of those nodes there is a separate parameter associated with the affect of each element of the input of the layer on the activation value of that node.

Likewise, for CNNs, the number of convolutional layers (which is shown as the ninth column of Table 3.31), as well as the kernel widths on each layer (r_1, r_2 and r_3, which are in the $10^{th} - 13^{th}$ column of Table 3.31) as well as the number of kernels on each of those layers (shown on the 14^{th} to 16^{th} columns) all affect the expressiveness of the model. As outlined in Section 2.6.2, the number of kernels used on a layer allows us to match more patterns, while having a wider kernel allows us to match patterns of a longer duration. Appropriate kernel widths may depend on the application in which the CNNs are being used in.

For the rest of this chapter, we discuss the methods for exploration and the effect of hyperparameter choice on predictive results on typical HAR tasks with accelerometer data.

3.3 Investigating hyperparameters effect on model performance

In order to estimate the impact of each hyperparameter on the performance observed across all experiments we apply the fANOVA analysis framework. fANOVA (Hutter et al., 2014) determines the extent to which each hyperparameter contributes to a network's performance. It builds a predictive model (Random Forest) of the model performance as a function of the model's hyperparameters. This non-linear model is then decomposed into marginal and joint interaction functions of the hyperparameters, from which the percentage contribution to overall variability of network performance is obtained. fANOVA has been used previously to explore the hyperparameters in recurrent networks (Greff et al., 2015).

For the practitioner it is important to know which aspect of the model is the most crucial for performance. We grouped the hyperparameters of each model into one of three categories (see Table 3.31): i) *learning*, parameters that control the learning process; ii) *regularisation*, parameters that limit the modelling capabilities of the model to avoid overfitting; and iii) *architecture*, parameters that affect the structure of the model. Based on the variability observed for each hyperparameter we estimate the variability that can be attributed to each parameter category, and to higher order interactions between categories.

3.3.1 Experiments for hyperparameter selection

The different hyper-parameters explored in this work are listed in Table 3.31. The last column indicates the number of parameter configurations sampled for each dataset, selected to represent an equal amount of computation time. We conduct experiments on three benchmark datasets representative of the problems tyical for HAR (described below). Experiments were run on a machine with three GPUs, where two model configurations are run on each GPU except for the largest networks.

After each epoch of training we evaluate the performance of the model on the validation set. Each model is trained for at least 30 epochs and for a maximum of 300 epochs. After 30 epochs, training stops if there is no increase in validation performance for 10 subsequent epochs. We select the epoch that showed the best validation-set performance and apply the corresponding model to the test-set.

3.3.2 Datasets

We select three datasets typical for HAR for the exploration in this work. Each dataset corresponds to an application of HAR. The first dataset, Opportunity, contains manipulative gestures like opening and closing doors, which are short in duration and non-repetitive. The second, PAMAP2, contains prolonged and repetitive physical activities typical for

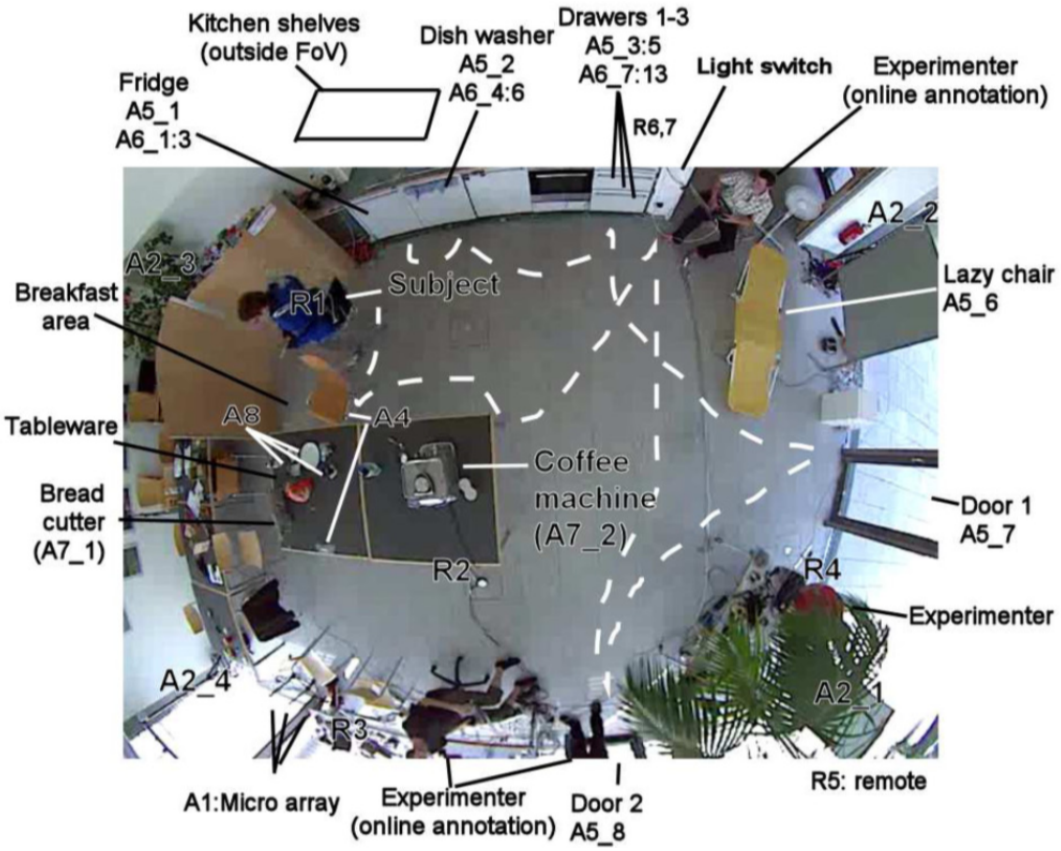
		Learning rate δ	LR decay ϑ	Sequence length L_{seq}	momentum γ	max-in norm	p_{carry}	#layers L	#units M	#conv-layers	Kernel width r_1	Kernel width r_2	Kernel width r_3	$\#\text{Kern}_1$	$\#\text{Kern}_2$	$\#\text{Kern}_3$	#experiments
	Category	Learning			Regularisation			Architecture									
	log-uniform?	y	y	-	-	-	-	-	-	-	-	-	-	-	-	-	
DNN	max	10^{-1}	10^{-3}	-	0.99	4.0	-	5	2048	-	-	-	-	-	-	-	1000
	min	10^{-4}	10^{-5}	-	0.0	0.5	-	1	64	-	-	-	-	-	-	-	
CNN	max	10^{-1}	10^{-3}	-	0.99	4.0	-	3	2048	3	9	5	3	128	128	128	256
	min	10^{-4}	10^{-5}	-	0.0	0.5	-	1	64	1	3	3	3	16	16	16	
LSTM-F	max	10^{-1}	-	64	-	4.0	1.0	3	384	-	-	-	-	-	-	-	128
	min	10^{-3}	-	8	-	0.5	0.0	1	64	-	-	-	-	-	-	-	
LSTM-s	max	10^{-1}	-	196	-	4.0	1.0	3	384	-	-	-	-	-	-	-	128
	min	10^{-3}	-	32	-	0.5	0.0	1	64	-	-	-	-	-	-	-	
b-LSTM-s	max	10^{-1}	-	196	-	4.0	1.0	1	384	-	-	-	-	-	-	-	128
	min	10^{-3}	-	32	-	0.5	0.0	1	64	-	-	-	-	-	-	-	

Table 3.31: Hyper-parameters of the models and the ranges of values explored in experiments.

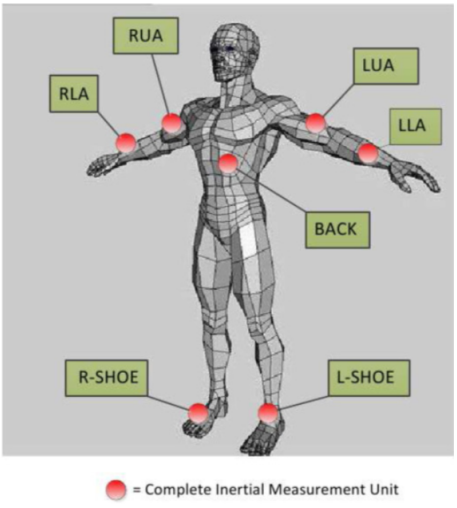
systems aiming to characterise energy expenditure. The last, Daphnet Gait, corresponds to a medical application where participants exhibit a typical motor complication in Parkinson's disease that is known to have a large inter-subject variability. Below we detail each dataset:

The **Opportunity dataset (Opp)** (Chavarriaga et al., 2013) consists of annotated recordings from on-body sensors from 4 participants instructed to carry out common kitchen activities. Data is recorded at a frequency of 30Hz from multiple locations on the body, and annotated with 18 mid-level gesture annotations (e.g. Open Door / Close Door). For each subject, data from 5 different runs is recorded. We used the subset of sensors that did not show any packet-loss, which included accelerometer recordings from the upper limbs, the back, and complete IMU data from both feet. The sensor locations on the body as well as the room where the data was recorded is illustrated in Figure 3.31. The resulting dataset had 79 dimensions. We use run 2 from subject 1 as our validation set, and replicate the most popular recognition challenge by using runs 4 and 5 from subject 2 and 3 in our test set. The remaining data is used for training. For frame-by-frame analysis (used in DNN, CNN and LSTM-F models), we created sliding windows of duration 1 second and 50% overlap. The resulting training-set contains approximately $650,000$

samples ($43,000$ frames).



A top view of the data recording room. The dashed line shows a typical user trajectory in the drill run.



The positions of on-body sensors.

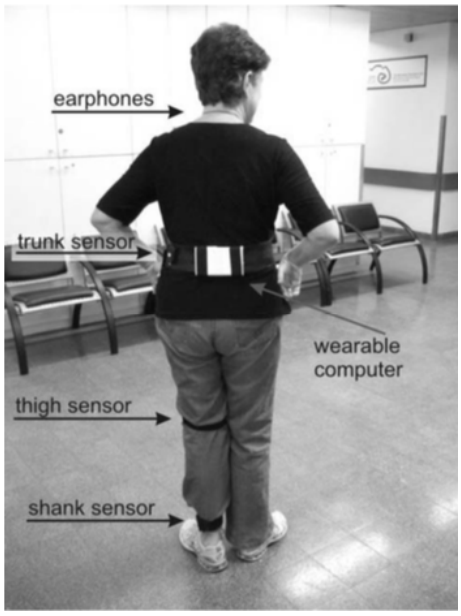
Figure 3.31: Setup for data collection in the Opportunity Dataset (Chavarriaga et al., 2013).

The **PAMAP2 dataset** (Reiss and Stricker, 2012) consists of recordings from 9 participants instructed to carry out 12 lifestyle activities, including household activities and a variety of exercise activities (Nordic walking, playing soccer, and more shown in Table 3.32). Accelerometer, gyroscope, magnetometer, temperature and heart rate data are recorded from inertial measurement units located on the hand, chest and ankle over 10 hours (in total). The resulting dataset has 52 dimensions. We used runs 1 and 2 for subject 5 in our validation set and runs 1 and 2 for subject 6 in our test set. The remaining data is used for training. In our analysis, we downsampled the accelerometer data to 33.3Hz in order to have a temporal resolution comparable to the Opportunity dataset. For frame-by-frame analysis, we replicate previous work with non-overlapping sliding windows of 5.12 seconds duration with one second stepping between adjacent windows (78% overlap) (Reiss and Stricker, 2012). The training-set contains approximately $473,000$ samples ($14,000$ frames).

The **Daphnet Gait dataset (DG)** (Bachlin et al., 2009) consists of recordings from 10 participants affected with Parkinson's Disease (PD), instructed to carry out activities which are likely to induce freezing of gait. Freezing is a common motor complication in PD, where affected individuals struggle to initiate movements such as walking. The objective is to detect these freezing incidents, with the goal to inform a future situated prompting system. This represents a two-class recognition problem. Accelerometer data was recorded from above the ankle, above the knee and on the trunk. An illustration of data collection is given in Figure 3.32. The resulting dataset has 9 dimensions. We used run 1 from subject 9 in our validation set, runs 1 and 2 from subject 2 in our test set, and used the rest for training. In our analysis, we downsampled the accelerometer data to 32Hz. For frame-by-frame analysis, we created sliding windows of 1 second duration and 50% overlap. The training-set contains approximately $470,000$ samples ($30,000$ frames).

Activity Name
Lying
Sitting
Standing
Walking
Running
Cycling
Nordic Walking
Computer work
Car driving
Ascending stairs
Descending stairs
Vacuum cleaning
Ironing
Watching TV
Folding laundry
House cleaning
Playing soccer
Rope jumping
Other (transient activities)

Table 3.32: List of activities performed by subjects in the PAMAP2 dataset (Reiss and Stricker, 2012).



Sensor locations in the Daphnet Gait dataset (Bachlin et al., 2009): sensors are attached to the shank (just above the ankle) and the thigh (just above the knee) using an elasticized strap and Velcro. A third sensor is attached to the belt where also the wearable computer (used for data storage and automatically prompting the patient) is attached to.

A snapshot of the study taking place . A Parkinson's Disease patient is depicted alongside a therapist (near the subject for safety reasons) and the research assistants (more remotely from the patient) who were documenting trials.

Figure 3.32: Setup for data collection in the Daphnet Gait dataset (Bachlin et al., 2009).

	PAMAP2	DG	OPP	
Performance	Mean F_1	F_1	Mean F_1	Weighted F_1
DNN	0.904	0.633	0.575	0.888
CNN	**0.937**	0.684	0.591	0.894
LSTM-F	0.929	0.673	0.672	0.908
LSTM-s	0.882	**0.760**	0.698	0.912
b-LSTM-s	0.868	0.741	**0.745**	**0.927**
CNN	Yang et al. (2015)		–	0.851
CNN	Ordóñez and Roggen (2016)		0.535	0.883
DeepConvLSTM	Ordóñez and Roggen (2016)		0.704	0.917
Delta from median	Δ(Mean F_1)	ΔF_1	Δ(Mean F_1)	Δ(Weighted F_1)
DNN	0.129	0.149	0.357	0.221
CNN	**0.071**	**0.122**	0.120	**0.104**
LSTM-F	0.10	0.281	0.085	0.156
LSTM-s	0.128	0.297	**0.079**	0.168
b-LSTM-s	0.087	0.221	0.205	0.172

Table 3.41: Best results obtained for each model and dataset, along with some baselines for comparison. Mean and weighted F_1 scores are defined in Equations (3.2) and (3.6) respectively. Delta from median (lower part of table) refers to the absolute difference between peak and median performance across all experiments.

3.4 Results

In this work we explored the performance of state-of-the-art deep learning approaches for Human Activity Recognition using wearable sensors. We described how to train recurrent approaches in this setting and introduced a novel regularisation approach. In thousands of experiments we evaluated the performance of the models with randomly sampled hyper-parameters. We found that bi-directional LSTMs (b-LSTM-s, introduced in Section 2.6.4) outperform the current state-of-the-art on Opportunity, a large benchmark dataset, by a considerable margin.

However, interesting from a practitioner's point of view is not the peak performance for each model, but the process of parameter exploration and insights into their suitability for different tasks in HAR. Recurrent networks (LSTM-s and to a lesser extent LSTM-F) outperform convolutional networks significantly on activities that are short in duration but

have a natural ordering, where a recurrent approach benefits from the ability to contextualise observations across long periods of time — for example on the Opportunity dataset. For bi-directional RNNs we found that the number of units per layer has the largest effect on performance across all datasets. For prolonged and repetitive activities like walking or running we recommend to use CNNs. Their average performance in this setting makes it more likely that the practitioner discovers a suitable configuration, even though we found some RNNs that work similarly well or even outperform CNNs in this setting. We further recommend to start exploring hyperparameters related to learning rates (those hyperparameters in the first group of columns in Table 3.31), before optimising the architecture of the network (those hyperparameters in the rightmost columns of Table 3.31), as the learning-parameters had the largest effect on performance in our experiments (see Figure 3.41(d)).

We found that models differ in the spread of recognition performance for different parameter settings. Regular DNNs, a model that is probably the most approachable for a practitioner, requires a significant investment in parameter exploration and shows a substantial spread between the peak and median performance. Practitioners should therefore not discard the model even if a preliminary exploration leads to poor recognition performance. More sophisticated approaches like CNNs or RNNs show a much smaller spread of performance, and it is more likely to find a configuration that works well with only a few iterations.

Results are illustrated in Figure 3.41. Graphs (a-c) show the cumulative distribution of the main performance metric on each dataset. Graph (d) illustrates the effect of each category of hyper-parameter estimated using fANOVA.

Overall we observe a large spread of peak performances between models on OPP and DG, with more than 15% mean F_1-score between the best performing approach (b-LSTM-s) and the worst (DNN) on OPP (12% on DG) (see Table 3.41). On PAMAP2

this difference is smaller, but still considerable at 7%. The best performing approach on OPP (b-LSTM-s) outperforms the current state-of-the-art by a considerable margin of 4% mean F_1-score (1% weighted F_1-score). The best CNN discovered in this work further outperforms previous results reported in the literature for this type of model by more than 5% mean F_1-score and weighted F_1-score (see Table 3.41). The good performance of recurrent approaches, which model movement at the sample level, holds the potential for novel (real-time) applications in HAR, as they alleviate the need for segmentation of the time-series data.

The distributions of performance scores differ between the models investigated in this work. CNNs show the most characteristic behaviour: a fraction of model configurations do not work at all (e.g. 20% on PAMAP2), while the remaining configurations show little variance in their performance. On PAMAP2, for example, the difference between the peak and median performance is only 7% mean F_1-score (see Table 3.41). The DNNs show the largest spread between peak and median performance of all approaches of up to 35.7% on OPP. Both forward RNNs (LSTM-F, LSTM-s) show similar behaviour across the different datasets. Practically all of their configurations explored on PAMAP2 and OPP have non-trivial recognition performance.

The effect of each category of hyperparameter on the recognition performance is illustrated in Figure 3.41(d). Interestingly, we observe the most consistent effect of the parameters in the CNN. In contrast to our expectation it is the parameters surrounding the learning process (see Table 3.31) that have the largest main effect on performance. We expected that for this model the rich choice of architectural variants should have a larger effect. For DNNs we do not observe a systematic effect of any category of hyperparameter. On PAMAP2, the correct learning parameters appear to the be the most crucial. On OPP it is the architecture of the model. Interestingly we observed that relatively shallow networks outperform deeper variants. There is a drop in performance for networks with

more than 3 hidden layers. This may be related to our choice to solely rely on supervised training, where a generative pre-training may improve the performance of deeper networks.

The performance of the frame-based RNN (LSTM-F) on OPP depends critically on the carry-over probability introduced in this work. Both always retaining the internal state (p_{carry} closer to 1) and always forgetting the internal state (p_{carry} closer to 0) lead to the lower performance. We found that p_{carry} of 0.5 works well for most settings. Our findings merit further investigation, for example into a carry-over *schedule*, which may further improve LSTM performance.

Results for sample-based forward LSTMs (LSTM-s) mostly confirm earlier findings for this type of model that found learning-rate to be the most crucial parameter Greff et al. (2015). However, for bi-directional LSTMs (b-LSTM-s) we observe that the number of units in each layer has a suprisingly large effect on performance, which should motivate practitioners to first focus on tuning this parameter.

3.5 Discussion and Conclusion

In this chapter we explored the performance of state-of-the-art deep learning approaches for Human Activity Recognition using wearable sensors. We described how to train recurrent approaches in this setting and introduced a novel regularisation approach. In thousands of experiments we evaluated the performance of the models with randomly sampled hyper-parameters. We found that bi-directional LSTMs (b-LSTM-s) outperform the current state-of-the-art on Opportunity, a large benchmark dataset, by a considerable margin. This approach may be where particularly useful where data for HAR is collected from a naturalistic environment (similar to the Opportunity dataset) and where time ordering of activities is relevant to the application.

However, interesting from a practitioner's point of view is not the peak performance

for each model, but the process of parameter exploration and insights into their suitability for different tasks in HAR. Recurrent networks outperform convolutional networks significantly on activities that are short in duration but have a natural ordering, where a recurrent approach benefits from the ability to contextualise observations across long periods of time. For bi-directional RNNs we found that the number of units per layer has the largest effect on performance across all datasets. For prolonged and repetitive activities like walking or running we recommend to use CNNs. Their average performance in this setting makes it more likely that the practitioner discovers a suitable configuration, even though we found some RNNs that work similarly well or even outperform CNNs in this setting. We further recommend to start exploring learning-rates, before optimising the architecture of the network, as the learning-parameters had the largest effect on performance in our experiments.

We found that models differ in the spread of recognition performance for different parameter settings. Regular DNNs, a model that is probably the most approachable for a practitioner, requires a significant investment in parameter exploration and shows a substantial spread between the peak and median performance. Practitioners should therefore not discard the model even if a preliminary exploration leads to poor recognition performance. More sophisticated approaches like CNNs or RNNs show a much smaller spread of performance, and it is more likely to find a configuration that works well with only a few iterations.

In future work, the approaches discussed in this chapter, approaches such as bi-directional LSTMs (b-LSTM-s) could be integrated with the approach described in Equation (4.12) to construct better covariates for prediction of stroke patient recovery.

3.A Multinomial model and derivation of its log-likelihood

To construct the prediction function using supervised learning, we consider a model for the activity class c_h^i the i^{th} subject is undertaking as being multinomially-distributed with one trial at each time index h (where the first argument of the multinomial distribution is the number of trials):

$$c_h^i \sim \text{Multinomial}(1, \rho_{h,1}^i, ..., \rho_{h,p}^i) \quad h \in \mathbb{N} \qquad h < N_* \qquad \sum_{m=1}^{p} \rho_{h,m}^i = 1 \tag{3.8}$$

where $\rho_{h,m}^i = P(c_h^i = m|\boldsymbol{x}_{ij,h})$ is the probability of activity m at discrete time index h for subject i, and $\boldsymbol{x}_{ij,h}$ is the hth preprocessed sliding window from the ith subject from the jth accelerometer data recording session.

Using this approach assumes that individual datapoints are independent, i.e. that $P(c_{h+1}^i|c_h^i) = P(c_{h+1}^i)$. However, this may not be the case, especially with datapoints which is from the same subject and temporally close to each other. Some solutions to this problem could include collecting more data such that there exists a sufficient number of datapoints far away from one another temporally to be more independent, as well as collecting data from different subjects.

We model the probabilities $\rho_{h,m}^i$ based on covariates extracted from a neural network describing the contents of the hth sliding window from the ith subject $\boldsymbol{s}_h^i \in \mathbb{R}$ (which is for the i^{th} subject, and is discussed in Section 2.9):

$$\rho_{h,m}^i = P(c_h^i = m|\boldsymbol{s}_h^i) = \frac{P(\boldsymbol{s}_h^i|c_h^i = m)P(c_h^i = m)}{\sum_{k=1}^{p} P(\boldsymbol{s}_h^i|c_h^i = k)P(c_h^i = k)} \tag{3.9}$$

Let $\eta_{h,m} = \ln P(\boldsymbol{s}_h^i|c_h^i = m)P(c_h^i = m) \quad \eta_{h,m} \in \mathbb{R} \quad m \in 1, ..., p$. Then, the proceeding equation reduces to a normalized exponential — often referred to in the neural network literature as *softmax* (Goodfellow et al., 2016):

$$\rho^i_{h,m} = P(c^i_h = m|\boldsymbol{s}^i_h) = \frac{\exp(\eta_{h,m})}{\sum_{k=1}^{p} \exp\left(\eta_{h,k}\right)} \tag{3.10}$$

We model $\eta_{h,m}$ as a linear combination of our covariates in the final layer of our neural network model:

$$\eta_{h,m} = \boldsymbol{\beta}^T_m \boldsymbol{s}^i_h + \beta_{m_o} \tag{3.11}$$

where $\beta_{m_o} \in \mathbb{R}$ and $\boldsymbol{\beta}_m \in \mathbb{R}^{M_L}$ are learnable parameters. Since the final layer of our network has M_L nodes, therefore the network outputs a vector of length M_L, so $\boldsymbol{s}^i_h$ and the associated parameters are in $\mathbb{R}^{M_L}$.

Let $\boldsymbol{\rho}^i_h = (\rho^i_{h,1}, ..., \rho^i_{h,p})$. The log-likelihood of a multinomial distribution over all of the data from n subjects, with N_i datapoints from the i^{th} subject, is:

$$\ell(c^1_1, ..., c^n_{N_*} | \boldsymbol{\rho}^1_1, ..., \boldsymbol{\rho}^n_{N_*}) = \sum_{i=1}^{n} \sum_{h=1}^{N_i} \sum_{m=1}^{p} I(c^i_h = m) \ln(\rho^i_{h,m}) \tag{3.12}$$

where $I(\cdot)$ is an indicator function which takes the value of 1 if the statement in its argument is true and 0 otherwise (Bishop, 2006).

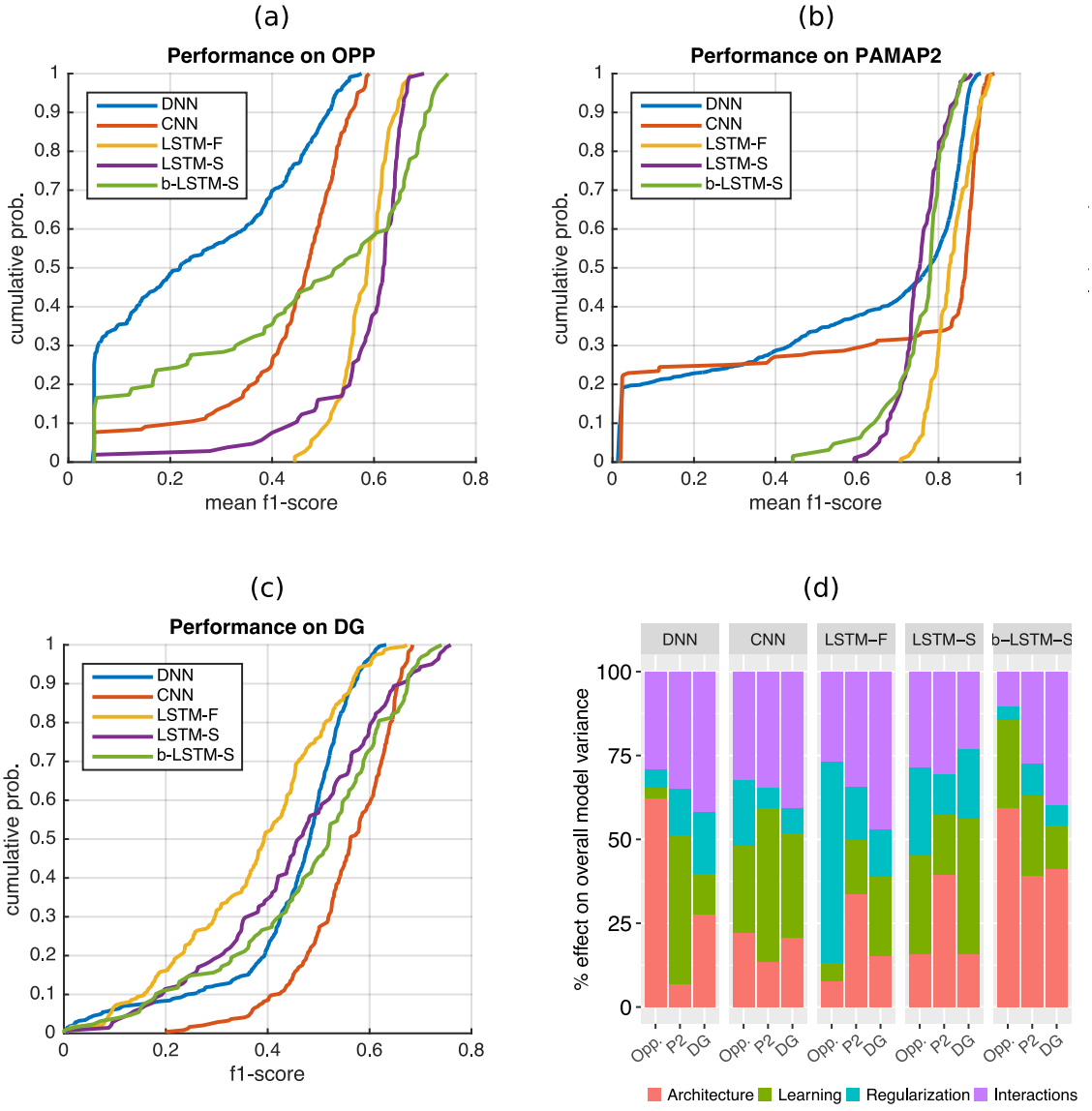


Figure 3.41: (a)-(c): Cumulative distribution of recognition performance for each dataset. (d): results from fANOVA analysis, illustrating impact of hyperparameter-categories on recognition performance (see table 3.31).

Chapter 4

New features for stroke patients' accelerometer data

In this chapter, we outline how we extract useful covariates from large scale accelerometer data collected from stroke patients. We construct new covariates describing aspects of activities of the stroke patients, and in doing so use some of the preprocessing techniques outlined in Chapter 2, namely Signal Vector Magnitude, sliding window extraction and Principal Components Analysis. These covariates are used in Chapter 5 to predict the recovery levels of stroke patients. Two separate approaches are employed to construct covariates. Firstly, a Gaussian Mixture Model-based method which allows us to visualise the clustering patterns used in our model. We analyse and plot the learned Gaussian cluster components which are most important in predicting stroke rehabilitation levels. Separately, we outline a Multi-Instance Learning (MIL)-based method based on aggregating the outputs of regression models from each sliding window in the dataset from each patient that week. That method allows us to visualise aspects of recovery through time.

This chapter is structured as follows. In Section 4.1 we give an outline of how data was collected from the patients and preprocessed. In Section 4.2 we give an account of

how we extracted new features or covariates describing the movements of the patients. In Section 4.3 we describe how visualizations can be made of the workings of our covariate generation methods. Finally, in Section 4.4 we provide a conclusion summarising our findings and contributions in this chapter.

4.1 Methodology

Data was collected from 59 patients who had a stroke in the past 2 years. Accelerometer data was collected at 100Hz from both wrists of each patient for 24 hours a day, for 3 days per week over the course of 8 weeks. Concurrently, patients' upper limb functionality was assessed using the CAHAI-9 score, a measure of how much upper limb function a patient has recovered so far, which is discussed in greater detail in Section 5.1 (Barreca et al., 2005).

One AX3 triaxial lightweight accelerometer[1] was placed on each wrist, attached with a wrist strap. We use data from both wrists due to the asymmetric nature of stroke. Prediction based on activity levels from just one side of the body likely wouldn't be effective (Wei et al., 2018), as a key feature of stroke is parallelization of one side of the body (hemiparesis), which can be best understood when we know the activity levels on both sides of the body. Then, a model based on time series from both sides of the body is likely to work best, such as those described in Sections 4.2.1 and 4.2.2.

4.1.1 Preprocessing steps used on accelerometer data

In the analysis of the accelerometer data, preprocessing is a necessary step used to remove the noise of the original signals, reduce the dimensionality and extract representative features from the raw accelerometer data. The preprocessing procedure can improve the

[1] https://axivity.com/product/ax3

quality of the raw data and such improvements is later measured by fitting the predictive model over the data. In order to eliminate the effects of the potentially irrelevant night-time data, we (1) only focus on the analysis of the data collected during the daytime hours from 9 a.m. to 9 p.m.. Then we (2) compute the Signal Vector Magnitude of the signal as discussed Section 2.3, take one second sliding windows with 50% overlap and (3) take the first 10 principal components of each sliding windows as described in Section 2.7 (the first 10 components account for approximately 75% of the variation in the data). See Figure 4.11 for a diagrammatic overview of the preprocessing steps we have used.

Let $\boldsymbol{x}_{1ij,h}$ and $\boldsymbol{x}_{2ij,h}$ be the vectors including the first 10 principal components of the hth sliding window for the paretic and non-paretic hand of the ith patient for the jth week, respectively. Both $\boldsymbol{x}_{1ij,h}$ and $\boldsymbol{x}_{2ij,h}$ are 10×1 vectors of sliding windows. Then, all the following analyses are based on the dataset $\{\boldsymbol{x}_{aij,h} : a = 1, 2; i = 1, ..., n; j = 1, ..., n_i, h = 1, ..., H_{ij}\}$

where H_{ij} is the number of sliding windows from the i^{th} patient over the j^{th} week. The first subscript, a, indexes over limbs (where a value of 1 corresponds with the paretic-side limb and 2 the non-paretic limb). The second subscript, i, indexes over patients. The third, j, indexes over weeks. The final subscript, h, indexes over sliding windows. n_i refers to the number of weeks of data collected from patient i.

4.1.2 Parallel Computing

To preprocess our data, we make use of Apache Spark version 2.3 (Meng et al., 2016; Shanahan and Dai, 2015; Zaharia et al., 2012; Armbrust et al., 2015; Kestelyn, 2013) to manage the splitting of the computational tasks across worker machines in a computing cluster. We create a Spark cluster on Microsoft Azure's HDInsight with 4 large-size worker nodes. We estimate the cost of running all of the machines to be approximately

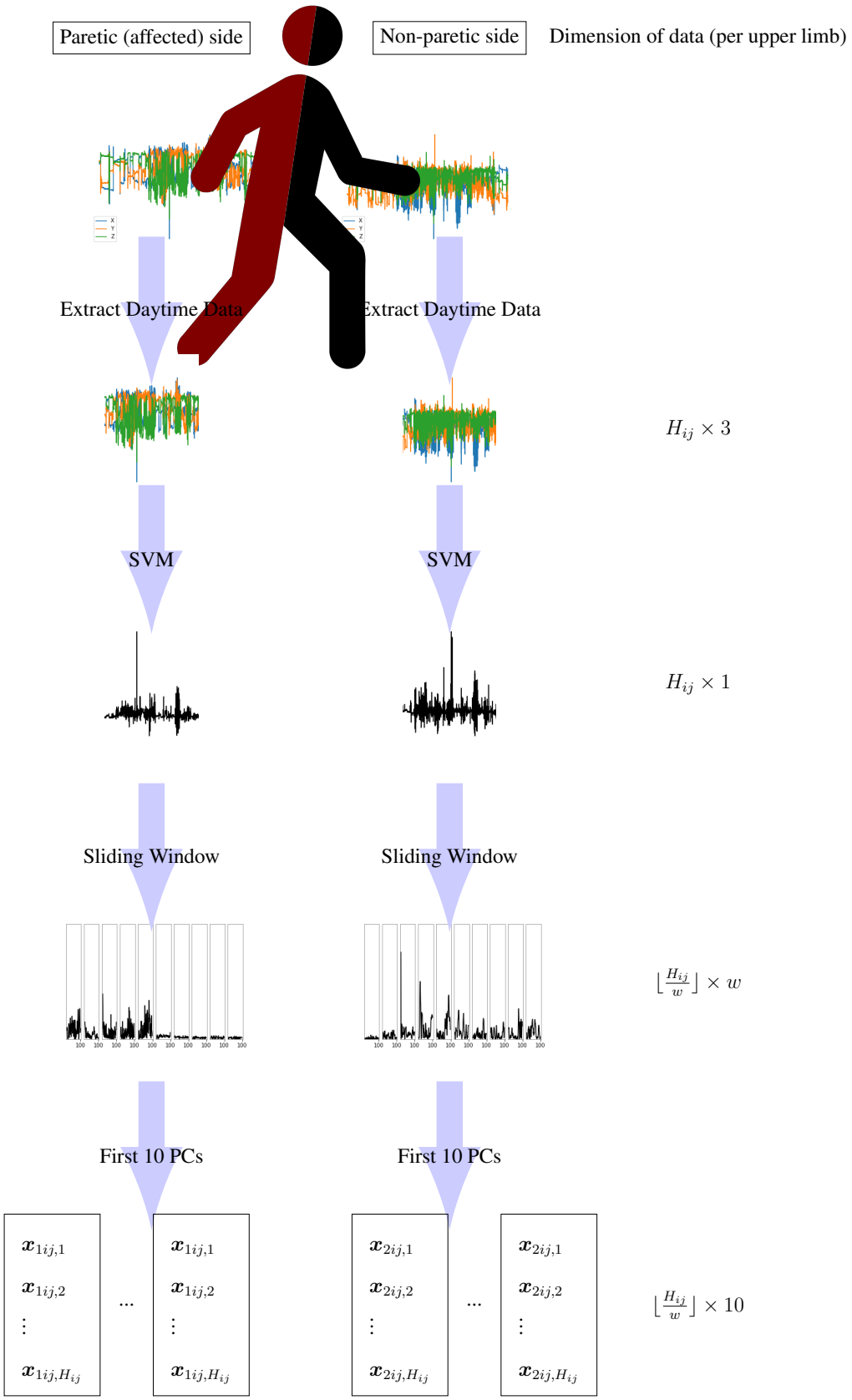


Figure 4.11: Preprocessing steps used

5 GBP per hour. Processing our 120GB of Comma Separated Values format data from 59 chronic and acute patients takes approximately 7 hours (approximately 8 minutes per patient).

4.2 Feature extraction approaches

In this section, we outline two feature extraction approaches – one based on mixture model features and another based on Multi Instance Learning – and contrast the qualities of both.

4.2.1 GMM-based features

We begin with the application of Gaussian Mixture Models (Bishop, 2006) as a tool for clustering the accelerometer data. GMMs have been widely used as a clustering method for both theoretical and computational considerations. With GMMs, the Activities of Daily Living (ADLs) measured by the accelerometer can be partitioned into homogeneous groups, which can capture the information contained in the raw accelerometer data and also will facilitate modelling the recovery level of the upper limb function. Assume that each $\boldsymbol{x}_{aij,h}$ comes from a mixture of K Gaussian distributions such that

$$\begin{aligned} &f(\boldsymbol{x}_{aij,h}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_{aij,h}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \\ &a = 1, 2;\ i = 1, \ldots, n;\ j = 1, \ldots, n_i;\ h = 1, \ldots, H_{ij}, \end{aligned} \tag{4.1}$$

where π_k is the mixing proportions with constraints $\sum_{k=1}^{K} \pi_k = 1$ and $0 \leq \pi_k \leq 1$ for all $k = 1, \ldots, K$. $\mathcal{N}(\cdot; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is the corresponding density of multivariate Gaussian distribution of $\boldsymbol{x}_{aij,h}$ parameterized by $\boldsymbol{\theta}_k = \{\boldsymbol{\mu_k}, \boldsymbol{\Sigma}_k\}$. Let $\boldsymbol{L}_{aij} = (L_{aij,1}, \ldots, L_{aij,H_{ij}})^{\mathrm{T}}$ be a latent variable with $L_{aij,h} \in \{1, \ldots, K\}$, where $L_{aij,h} = k$ indicates that $\boldsymbol{x}_{aij,h}$ belongs to the k-th cluster. For notational simplicity, denote the data from all patients as

$\boldsymbol{x} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, data from all weeks from patient i as $\boldsymbol{x}_i = \{\boldsymbol{x}_{i1}, \ldots, \boldsymbol{x}_{in_i}\}$, from patient i's jth week as $\boldsymbol{x}_{ij} = \{\boldsymbol{x}_{1ij}, \boldsymbol{x}_{2ij}\}$, where the sliding windows from that patient, from that week, on their ath upper limb is denoted as $\boldsymbol{x}_{aij} = \{\boldsymbol{x}_{aij,1}, \ldots, \boldsymbol{x}_{aij,H_{ij}}\}$. Similarly, we define $\boldsymbol{L} = \{\boldsymbol{L}_1, \ldots, \boldsymbol{L}_n\}$, $\boldsymbol{L}_i = \{\boldsymbol{L}_{i1}, \ldots, \boldsymbol{L}_{in_i}\}$ and $\boldsymbol{L}_{ij} = \{\boldsymbol{L}_{1ij}, \boldsymbol{L}_{2ij}\}$, where $\boldsymbol{L}_{aij} = (L_{aij,1}, \ldots, L_{aij,H_{ij}})^{\mathrm{T}}$. Then the joint distribution of the complete data set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{L}_1, \ldots, \boldsymbol{L}_n\}$ is

$$
\begin{aligned}
& f(\boldsymbol{x}, \boldsymbol{L}; \boldsymbol{\pi}, \boldsymbol{\theta}) \\
= & \prod_{i=1}^{n} \prod_{j=1}^{n_i} \prod_{a=1}^{2} \prod_{h=1}^{H_{ij}} f(L_{aij,h}) f(\boldsymbol{x}_{aij,h} | L_{aij,h}) \\
= & \prod_{i=1}^{n} \prod_{j=1}^{n_i} \prod_{a=1}^{2} \prod_{h=1}^{H_{ij}} \prod_{k=1}^{K} \{\pi_k \mathcal{N}(\boldsymbol{x}_{aij,h}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\}^{L_{aij,hk}},
\end{aligned}
$$

where

$$
L_{aij,hk} = \begin{cases} 1, & \text{if } L_{aij,h} = k \\ 0, & \text{if } L_{aij,h} \neq k \end{cases},
$$

$\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)^{\mathrm{T}}$ and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_k\}$. The complete data log-likelihood function is denoted as

$$
\ell(\boldsymbol{\pi}, \boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{L}) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{a=1}^{2} \sum_{h=1}^{H_{ij}} \sum_{k=1}^{K} L_{aij,hk} \ln[\pi_k \mathcal{N}(\boldsymbol{x}_{aij,h}; \boldsymbol{\theta}_k)]. \tag{4.2}
$$

The estimation of $(\boldsymbol{\pi}, \boldsymbol{\theta})$ can be obtained by maximizing Equation (4.2) using the EM algorithm. At the $(s+1)$-th iteration, the *E*-step involves calculating the posterior probability of $\boldsymbol{x}_{aij,h}$ belongs to the k-th cluster as

$$
P(L_{aij,hk} = 1 | \boldsymbol{x}_{aij,h}; \pi_k^{(s)}, \boldsymbol{\theta}_k^{(s)}) \triangleq \hat{L}_{aij,hk}^{(s)} = \frac{\pi_k^{(s)} \mathcal{N}(\boldsymbol{x}_{aij,h}; \boldsymbol{\theta}_k^{(s)})}{\sum_{l=1}^{K} \pi_l^{(s)} \mathcal{N}(\boldsymbol{x}_{aij,h}; \boldsymbol{\theta}_l^{(s)})}, \tag{4.3}
$$

where $\pi_k^{(s)}$ and $\boldsymbol{\theta}_k^{(s)} = (\boldsymbol{\mu}_k^{(s)}, \boldsymbol{\Sigma}_k^{(s)})$ are the values of π_k and $\boldsymbol{\theta}_k$ at the s-th iteration, re-

spectively. Given the above posterior probability $\hat{L}^{(s)}_{aij,hk} \in [0,1]$, the *M*-step involves maximizing the following equation

$$\tilde{\ell}(\boldsymbol{\pi}, \boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{L}) = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{a=1}^{2} \sum_{h=1}^{H_{ij}} \sum_{k=1}^{K} \hat{L}_{aij,hk} \ln[\pi_k \mathcal{N}(\boldsymbol{x}_{aij,h}; \boldsymbol{\theta}_k)]. \tag{4.4}$$

with constraint $\sum_{k=1}^{K} \pi_k = 1$. This can be achieved using a Lagrange multiplier and maximizing the following equation

$$\tilde{\ell}(\boldsymbol{\pi}, \boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{L}) + \lambda \bigg(\sum_{k=1}^{K} \pi_k - 1 \bigg),$$

which yields the updated equation for the mixing proportions as

$$\pi_k^{(s+1)} = \frac{1}{2 \times n \times n_i \times H_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{a=1}^{2} \sum_{h}^{H_{ij}} \hat{L}^{(s)}_{aij,hk}, \tag{4.5}$$

We can also obtain the updated equations for the mean and the covariance matrix as

$$\boldsymbol{\mu}_k^{(s+1)} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{a=1}^{2} \sum_{h=1}^{H_{ij}} \hat{L}^{(s)}_{aij,hk} \boldsymbol{x}_{aij,h}}{\sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{a=1}^{2} \sum_{h=1}^{H_{ij}} \hat{L}^{(s)}_{aij,hk}}, \tag{4.6}$$

and

$$\boldsymbol{\Sigma}_k^{(s+1)} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{a=1}^{2} \sum_{h=1}^{H_{ij}} \hat{L}^{(s)}_{aij,hk} (\boldsymbol{x}_{aij,h} - \boldsymbol{\mu}_k^{(s)})(\boldsymbol{x}_{aij,h} - \boldsymbol{\mu}_k^{(s)})^{\mathrm{T}}}{\sum_{i=1}^{n} \sum_{j=1}^{n_i} \sum_{a=1}^{2} \sum_{h=1}^{H_{ij}} \hat{L}^{(s)}_{aij,hk}}. \tag{4.7}$$

The above EM algorithm for clustering can be implemented by the following steps:

(1) Initialize $\hat{\boldsymbol{L}}^{(0)}$ and let $s = 1$.

(2) Calculating $\hat{\boldsymbol{\pi}}^{(s)}$ and $\hat{\boldsymbol{\mu}}^{(s)}$ with $\hat{\boldsymbol{L}}^{(s-1)}$ using equations (4.5) and (4.6).

(3) Calculating $\hat{\boldsymbol{\Sigma}}^{(s)}$ with $\hat{\boldsymbol{L}}^{(s-1)}$ and $\hat{\boldsymbol{\mu}}^{(s)}$ using equation (4.7).

(4) Update $\hat{\boldsymbol{L}}^{(s)}$ with $(\hat{\boldsymbol{\pi}}^{(s)}, \hat{\boldsymbol{\mu}}^{(s)}, \hat{\boldsymbol{\Sigma}}^{(s)})$ using equation (4.3).

(5) Repeat Step (2) - (4) until convergence.

New features

Hemiparesis often occurs in stroke survivors due to brain injury. For most stroke survivors, there is a dramatic reduction in the ability to use their paretic hand and they carry out ADLs mainly using their non-paretic hand instead. In evaluating the recovery level of the upper limb function by using the measurements of ADLs, it is of interest to find the most informative features from the clustering in order to build predictive models to estimate the recovery level of the upper limb function based on these features. By considering the asymmetry of the effect of the disease on upper limb movement, we propose to use the following new covariates for modelling the recovery level of the upper limb function of stroke survivors:

$$z_{ijk}^{GMM} = \sum_{h=1}^{H_{ij}} (\delta_{1ij,hk} - \delta_{2ij,h,k}),\ i = 1, \ldots, n;\ j = 1, \ldots, n_i;\ k = 1, ..., K, \tag{4.8}$$

where $\delta_{aij,hk}(a = 1, 2)$ is an indicator function such that

$$\delta_{aij,h,k} = \begin{cases} 1,\ \boldsymbol{x}_{aij,h} \text{ belongs to the } k\text{-th cluster} \\ 0,\ \text{otherwise.} \end{cases}$$

Actually, Equation (4.8) counts the difference between two upper limbs of the number of occurrences that the sliding windows $\{\boldsymbol{x}_{aij,h} : i = 1, \ldots, n;\ j = 1, \ldots, n_i;\ h = 1, \ldots, H_{ij};\ a = 1, 2\}$ are clustered into the k-th mixture component. In this way, the new features in Equation (4.8) can simultaneously capture the information contained in the raw

data and indicate the asymmetric feature of the upper limb actions, which are useful in assessing the recovery level of stroke survivors. More explanation about the new features will be given in Section 4.3.1.

4.2.2 Multi Instance Learning (MIL)-based covariates

According to Amores (2013) (which reviews approaches to Multi Instance Learning in a classification context), a bag is a set, where the elements are feature vectors called *instances* in Multi Instance Learning (MIL) terminology, and the number of instances can be different in each set. The objective of the MIL problem is to learn a model, at training time, that can be used to predict classification or regression problems for unseen bags.

This approach is applicable and useful to the case of modelling disease recovery scores using accelerometer data from each patient different weeks. For each patient for each week, we have a bag of H_{ij} of sliding windows $S_{ij} = \bigcup_{h=1}^{H_{ij}} \boldsymbol{x}_{ij,h}$, where $\boldsymbol{x}_{ij,h}$ are instance vectors.

Outline of Multi Instance Learning

Although Amores (2013) is mainly concerned with the classification problem, we see there are three main ways in which MIL methods can be grouped: *instance-space*, *bag-space* and *embedding space*. Instance-space methods involve extracting some useful feature from each instance vector and then performing an aggregation operation over all these features. Bag-space methods involve construction of features about bags in a global way, without aggregating over individual instances. Embedding-space methods seek to construct new vectors which describe the contents of entire bags, which can then be used in the classification or regression task. We are mainly concerned with instance-space methods, as they are amenable to computational parallelisation and their simplicity leads to interpretability advantages (as shown in Section 4.3.2).

Instance-based methods generally take the approach of carrying out a computation (e.g. a regression or classification prediction) on individual instances, and combining all of the outputs from the bag together using some aggregation operator:

$$\text{Prediction} = \frac{\zeta(\boldsymbol{x}_{ij,1}) \oplus ... \oplus \zeta(\boldsymbol{x}_{ij,H_{ij}})}{Z} \tag{4.9}$$

where $\oplus$ denotes some aggregation operator (popular choices include sum or max), and Z denotes some normalization constant (which could be the number of instances in each bag, or 1 if we choose not to use normalization). Some instance-based approaches assign instances different weights based on some inferred measure of importance, while others give the same weight to each instance. Modelling a time series problem with MIL would imply that we are treating the sliding windows as independent to one another, i.e. that $P(y_{ij}|x_{ij,h}) = P(y_{ij}|x_{ij,h}, x_{ij,k})\ \forall h \neq k$. This is somewhat unrealistic to assume, though in datasets of long duration pairs of points further away in time from one another may be less correlated.

Our approach to Multi Instance Learning

We use Random Forest regression to assign a recovery score to each instance, and use summation to aggregate. Finally, we employ *Isotonic regression* to calibrate our predictions (Best and Chakravarti, 1990; Chakravarti, 1989). We use isotonic regression to fit f_i to (the i^{th} week of) data. Then, our prediction can be used as a covariate in Chapter 5. The equation to obtain our new covariate becomes:

$$z_{ij0}^{*MIL} = f_i(\frac{1}{H_{ij}} \sum_{h=1}^{H_{ij}} \zeta(\boldsymbol{x}_{ij,h})) \tag{4.10}$$

where $\zeta(\boldsymbol{x}_{ij,h})$ is the instance-wise Random Forest Regression (Breiman, 2001; Pedregosa et al., 2011) prediction of CAHAI score of the hth sliding window of patient i

on week j, after concatenating the (preprocessed) data from the paretic and non-paretic upper limbs.

Isotonic regression fits a non-decreasing non-linear function to data, for example to predicted values produced by a model, in order to reduce the error of predictions further. This is useful in applications such as ours where there may be more bias in predicted values for some parts of the range of predicted values compared to others.

To fit function ζ, we use CAHAI scores associated with that week as the response, and assume it applies to all days in the week. Further work may involve sampling training accelerometer instances such that this assumption isn't necessary, potentially enhancing predictive performance. We discuss our rational for choosing to use Random Forest regression in Section 4.2.3, and describe the theoretical motivation for Random Forests in Appendix 4.A.

Excluding less relevant datapoints

To only consider datapoints exemplary of the most discriminative patterns, we may exclude datapoints where $\zeta(\boldsymbol{x}_{ij,h})$ is close to the median response. We exclude datapoints with $\zeta(\boldsymbol{x}_{ij,h})$ within a hyperparameter ι of the median response, $m = 37$. This is similar to the approach used in Gao et al. (2019).

Suppose we let

$$\zeta'(\boldsymbol{x}_{ij,h}) = \begin{cases} \zeta(\boldsymbol{x}_{ij,h}), & \text{if } |\zeta(\boldsymbol{x}_{ij,h}) - m| > \iota \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

and also if we let H'_{ij} be the number of elements of patient i's data for which $\zeta'(\boldsymbol{x}_{ij,h}) \neq 0$. Then, the covariates we construct become:

	Testing Partition	Training Partition
Fold 1	Patients 1,2,3,4,5	Rest
Fold 2	Patients 6,7,8,9,10	Rest
Fold 3	Patients 11,12,13,14,15	Rest
Fold 4	Patients 16,17,18,19,20	Rest
Fold 5	Patients 21,22,23,24	Rest

Figure 4.21: Patient-wise Training/testing partitioning scheme in evaluating function ζ in Equation (4.12).

$$z_{ijk}^{MIL} = f_i(\frac{1}{H'_{ij}} \sum_{h=1}^{H'_{ij}} \zeta'(\boldsymbol{x}_{ij,h})) \tag{4.12}$$

for various values of $\iota = \iota_1, ..., \iota_{K-1}$, the value for each of which is determined based on percentiles of the values of $\zeta(\boldsymbol{x}_{ij,1}), ..., \zeta(\boldsymbol{x}_{ij,H_{ij}})$. Subsequently, we model average over this hyperparameter, evaluating predictions with ι. This generates K covariates (when the output of Equation (4.10) is also included), which we use in Chapter 5 to model recovery level scores.

To ensure no ground truth leakage in our experiments, we group patients into 5 folds (shown in Figure 4.21), and use the function ζ' in Equation (4.12) for prediction on each fold by training it on data from all remaining folds.

4.2.3 Motivation for using Random Forest Regression

In this work, we chose to use the Random Forest (RF) algorithm (Breiman, 2001) for several reasons. Firstly, it is relatively straightforward to use and in many other applications gives close to state-of-the-art performance with little need for hyperparameter tweaking. Secondly, fitting an RF is relatively time efficient in comparison with other algorithms, especially for large datasets. Thirdly, excellent off-the-shelf implementations of the algorithm exist, for example in this work we use the implementation of Python's Sklearn 0.18.1 (Pedregosa et al., 2011). Fourthly, RFs have some of the advantages of non-parametric

classification algorithms, but have the benefit that the overall storage space required for the fitted models can be limited even when the models are trained on large datasets. This is in comparison to other classification and regression algorithms we could potentially have chosen, such as kNN (Cover and Hart, 1967), for which in naive implementations the trained model requires a storage space equal to the size of the training dataset (though more space-efficient formulations exist). If it is attempted to use kNN, then the size of the trained model will likely not fit in the RAM of a single computer, and this problem is further exacerbated when multiple models have to be stored when evaluating predictions using cross validation with several folds.

4.3 Visualisation results

In this section, we show some visualisation results associated with both of the covariate generation methods discussed in Sections 4.2.1 and 4.2.2. In Section 4.3.1, we show and interpret the learned clusters in our GMM-based method, and in Section 4.3.2 we analyse datapoint-wise trends in our MIL-based method.

4.3.1 Visualising learned cluster components

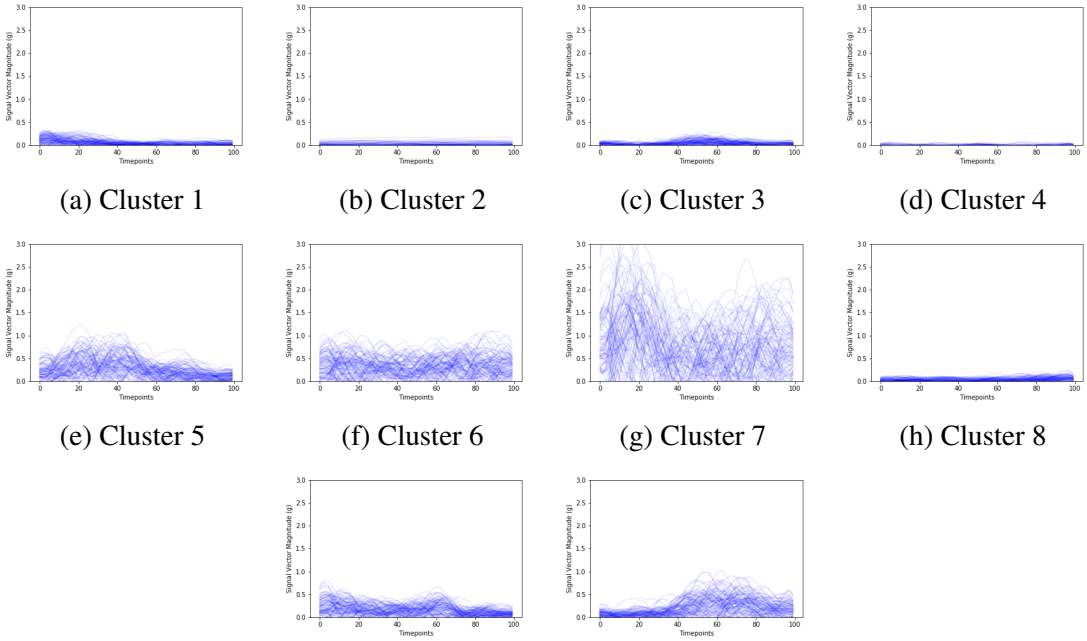
After preprocessing, the GMM-based method is used to cluster the data in each of acute group and chronic group. The number of the mixture component is set to be 20, which we assume that is large enough to cover all the patterns of ADLs during daytime. We randomly select 1% of all the sliding windows $\{\boldsymbol{x}_{aij,h} : i = 1\ldots, n; j = 1, \ldots, n_i;\ h = 1\ldots, H_{ij};\ a = 1, 2\}$ as the training data. Given the algorithm in Section 4.2.1, the unknown parameters $\{(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) : k = 1, \ldots, 20\}$ in the mixture components are trained and then the mixing proportions are predicted for all the sliding windows to decide their cluster membership. The clusters are presented in Figures 4.31 and 4.32.

We can see from those figures that the activities considered to be potentially from the same source are merged into a single cluster. In some clusters, such as Cluster 2 and Custer 4, the signals are near zero, perhaps correspond to those activities such as sleeping or sitting on the couch without doing anything. Most clusters correspond to the signals of having a relatively low amount of activity, see for example, Cluster 5, Cluster 6, Cluster 9, Cluster 11, Cluster 16 and Cluster 18. These signals might correspond to the situation that the patients practice to use their paretic hands to carry out activities. The less idleness on the paretic hand compared to the non-paretic hand, the better the patient is likely to be able to move his/her paretic hand. Cluster 7 corresponds to signals of doing an activity at a high intensity for a short of time, this might illustrate that the patient tried to carry out an activity but not being able to do it. If this patient is only able to sustain force for a short of time on their paretic hand, then they may not be recovering well. In clusters such as Cluster 10 and Cluster 15, the patient is idle for a period of time before carrying out activity at a relatively high intensity. This might indicate that if the patient uses the paretic hand more and more often than the non-paretic hand and thus they are recovering very well.

With these clusters, by taking the asymmetric feature of the upper limb actions after stroke into consideration, we calculate the new features based on Equation (4.8) for modelling the recovery level of the upper limb function.

4.3.2 Visualisation based on MIL-based method

Seasonal decomposition methods attribute variations in time series observations to overall trends, diurnal patterns, as well as random variation at individual timepoints. This allows us to plot long term inferred recovery score trends, without overwhelming the medical practitioner with information. To examine patterns in the data of patient j, we first define a shorthand notation $\zeta_h = \zeta(x_{ij,h})$. The time series ζ observed S times per day can be

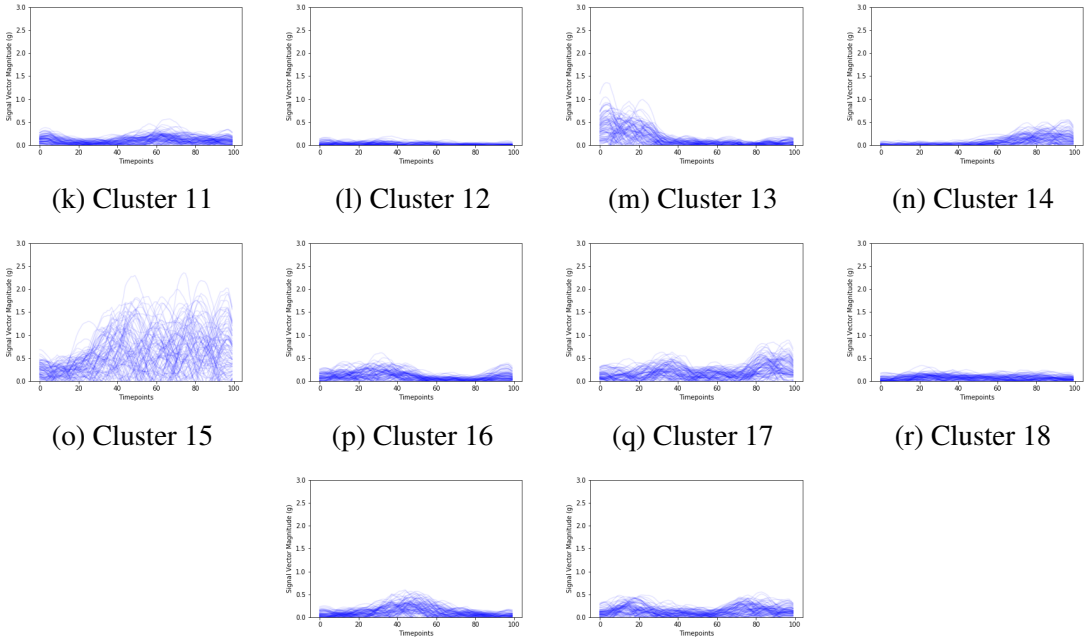


(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4

(e) Cluster 5 (f) Cluster 6 (g) Cluster 7 (h) Cluster 8

(i) Cluster 9 (j) Cluster 10

Figure 4.31: Clusters obtained by using 1% of the sliding windows.



(k) Cluster 11 (l) Cluster 12 (m) Cluster 13 (n) Cluster 14

(o) Cluster 15 (p) Cluster 16 (q) Cluster 17 (r) Cluster 18

(s) Cluster 19 (t) Cluster 20

Figure 4.32: Clusters obtained by using 1% of the sliding windows. (continued)

decomposed as (Cleveland et al., 1990; Seabold and Perktold, 2010):

$$\zeta_h = \zeta_h^{trend} + \zeta_{h \bmod S}^{season} + \varepsilon_h \qquad h \in \mathbb{N} \tag{4.13}$$

Extracted functions ζ^{trend}, ζ^{season} and ε on the right hand side of Equation (4.13) each provide understanding on patient recovery trend, diurnal pattern and outlier times, respectively. In Figure 4.33, we see the selected patient improves during the study, and see where fluctuations in this trend exist. In Figure 4.34, we see selected patient's estimated diurnal pattern in CAHAI scores: a slight daily pattern, where the patient can vary by about 1.5 units on the CAHAI scale over the course of the day. This may prompt a clinician to investigate the patient's activities during particular times in their daily schedule. Furthermore, Figure 4.35 indicates points of high residual ε, which perhaps could be flagged for attention to the clinician analysing the data.

4.4 Conclusion

In Chapter 5, we use the feature extraction methods discussed in this chapter to predict the recovery score of patients. The GMM-based approach has the advantage of allowing us to visualise clusters, allowing us to create more interpretable models. The MIL-based approach has the generalizability advantage: the model structure is not related to any aspect of stroke itself and could potentially be applied to prediction of affliction to other movement-related disorders, as long as periodic assessments could be completed alongside continuous accelerometer collection.

Furthermore, we show the possibility to extract useful daily patterns, trends as well as identify when unusual levels of capability are exhibited, which may be useful for guiding treatment plans for patients. Moreover, it would be interesting to understand and model how daily schedules of patients change as they rehabilitate, possibly even leading to im-

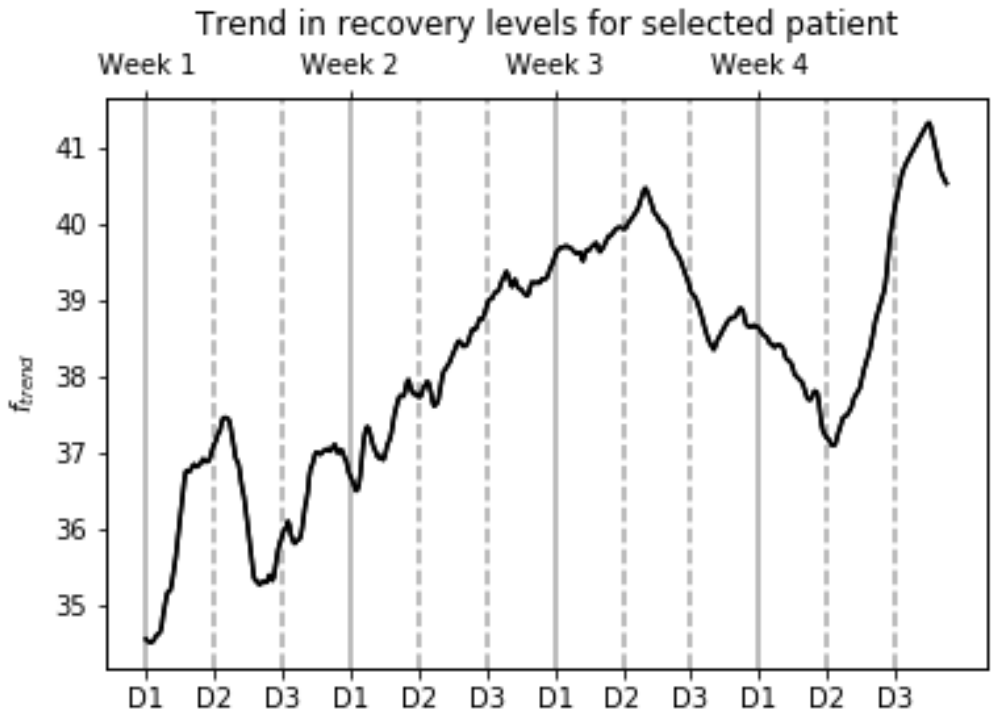


Figure 4.33: Extracted trend component ζ^{trend} in selected patient's recovery. Three days from the four weeks the patient was in the study are plotted, showing that capability deteriorated in day 2 of weeks 1 and 3.

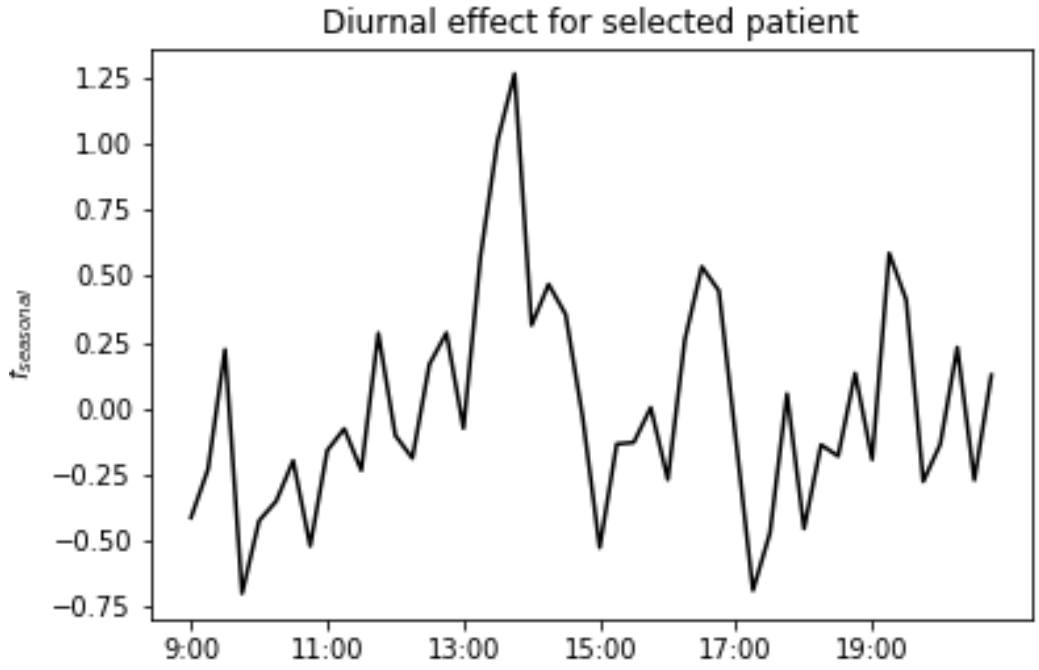


Figure 4.34: Selected patient's daily capabilities in ζ^{season} are shown for each 15-minute interval throughout their diurnal schedule. Perhaps the patient often undertakes a demanding activity after 1PM on many days.

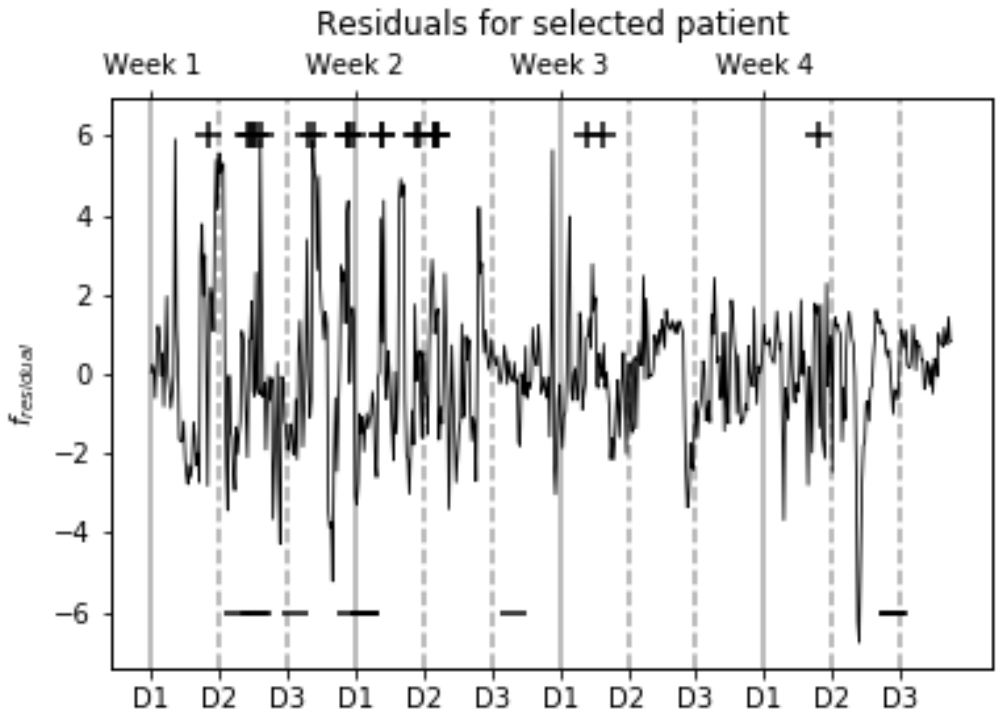


Figure 4.35: Residuals ε showing when selected patient exerts themselves more (less), given trend and diurnal context. Times with residual greater (less) than 2 standard deviations are denoted with a + (-) symbol.

proved stroke prompting systems. Also, it would be interesting to carry out a user study with daily behavioural diaries such as the Motor Activity Log (Uswatte et al., 2006b), to understand the effectiveness of our visualisation system on a daily basis.

4.A Theoretical Motivation for Random Forest

Whereas the Mean Square Error (MSE) of a model's predictions can be decomposed into terms related to the model's predictive bias and variance (Friedman et al., 2001), and many classification and regression models have hyperparameters which represent a tradeoff between flexible models (with lower predictive bias and higher variance) and less flexible models (with higher predictive bias though lower variance), *Ensemble Learning* models are composed of a large number of flexible models which are aggregated to reduce the overall predictive variance. In *bagging* (bootstrapped aggregation) for regression, each constituent model may be different due to different bootstrapped samples of the training data being taken, and the final predictions are arrived at by simply averaging the predictions of constituent models.

Deep CART trees (e.g. with little pruning) provide low bias, high variance predictors, which individually may not be so accurate. Breiman (2001) proposed RFs as an ensemble learning method over CART trees (Breiman et al., 1984), which take the bootstrapped aggregation approach previously mentioned, but also change how individual trees are constructed to make each have more variance in their predictions. In standard CART trees, the training dataset is partitioned recursively using a greedy algorithm, where each partition is made using the best covariate amongst all available covariates. However, in an RF, each node is split using the best of a subset of predictors randomly chosen at that node (Liaw and Wiener, 2002). This adds even more variance to the predictive performance of each constituent model, and the overall model turns out to perform well amongst other classifiers (Liaw and Wiener, 2002; Breiman, 2001).

Chapter 5

Prediction of stroke recovery score

In this chapter, we outline prediction methods for rehabilitation levels in stroke patients. A non-linear mixed effects (NLME) model is employed to model both fixed and individual effects. We use the covariates generated using the methods in Chapter 4, where GMM-based set of covariates are used separately to the MIL-based set of covariates in the same predictive models. Predictive performances are shown in Section 5.3. We note that the predictive performance of both covariate-generating methods from Chapter 4 are quite similar, though comparative *qualitative* advantages for each, interpretability for the GMM-based covariates and generalizability for the MIL-based covariates.

5.1 Background to stroke recovery prediction using accelerometer data

It has been proposed to use accelerometer sensors on both wrists in a patient's naturalistic environment to assess rehabiltation (see for example Noorkiv et al. (2014) for an excellent survey). Accelerometer-based systems are objective and have a cost saving advantage due to reduced requirement for trained assessment professionals.

Score	Description
1	0-24% of task completed.
2	25-49% of task completed.
3	50-74% of task completed.
4	75-99% of task completed.
5	100% of task completed, but with some cueing, coaxing or help with setting up objects.
6	Almost independent, apart from assistive devices, or more than reasonable time required for task completion, or there were safety concerns with how well the task is completed.
7	Task completed fully independently in a reasonable time, without modification or assistive devices.

Table 5.11: Scoring scale for tasks in CAHAI-9 assessment (Barreca et al., 2005).

In this study, we aim to predict patients' Chedoke Arm and Hand Inventory (CAHAI) recovery score. It measures recovered upper limb functionality in stroke patients. In this thesis, we refer to the CAHAI-9 variant, an assessment based on 9 tasks, for example, pouring a bottle of water, dialling 911, etc (Barreca et al., 2005). Tasks test the patient's capability to plan and complete simple tasks in a lab based environment. It is graded on a continuous scale from 9 to 63, the summation of scores involved in its composite tasks, each given a score between 1 and 7, as shown in Table 5.11. It is lab-based, thus may be affected by observer bias, such as when a patient puts in more effort completing tasks over the relatively short duration assessment duration (25 minutes) compared to what is possible to sustain while carrying out Activities of Daily Living (ADLs) in a naturalistic home environment (Lang et al., 2013; Barreca et al., 2005).

Related studies include computation of basic statistics describing a patient's behaviour (Joseph et al., 2018; Gebruers et al., 2014), or validating that various rehabilitation assessment scales – questionnaire-based (Uswatte et al., 2006a) and task-observation based (Thrane et al., 2011; Rand and Eng, 2012; Lang et al., 2007; Jing et al., 2011; Bailey et al., 2015; Gebruers et al., 2014) – correlate with real world activity levels. Those studies don't have a predictive focus in reporting a train-test split with their correlatory results. Predic-

tive studies include Kumar et al. (2013), which was based on data collected from patients in the semi-constrained inpatient hospital setting. To our knowledge, our work is the first study with a predictive focus (using a validatory testing set while attempting to predict some aspect of stroke recovery) which is based on naturalistically-collected free-living data from patients in normal community dwelling.

Other studies include assessment of upper limb functional rehabilitation using remote video and accelerometer-based systems (Patel et al., 2010; Yu et al., 2016; Salazar et al., 2014), ADL (activities of daily living) recognition (Roy et al., 2009; Anderson et al., 2018) and prompting exercise of affected limb using diurnal context (Moore et al., 2016). A survey of related works are given in Table 5.12.

5.2 Predictive model

We develop a predictive model to evaluate the recovery level of the upper limb function based on the features from Chapter 4 extracted from the raw accelerometer data and other variables. In this section we discuss the model, in Section 5.2.1 we discuss the estimation procedure, in Section 5.2.2 we discuss the predictive distribution of the random effects part, and in Section 5.2.3 we show the predictive distribution for the mixed effects model.

Let y_{ij} be the CAHAI response for the i-th patient measured at the j-th week ($i = 1, \ldots, n; j = 1, \ldots, n_i$). $\boldsymbol{z}_{ij} = (z_{ij1}, \ldots, z_{ijK}, y_{i,0})^{\mathrm{T}}$ is a $K' = (K+1)$-dimensional covariate vector including the features calculated from Equation (4.8) or (4.12), and where the last element refers to the initial CAHAI score at the start of the study. To address the problems of heterogeneity and nonlinearity, we consider the following nonlinear mixed effects model with the nonlinear patient-specific random effects modelled by a Gaussian process prior:

$$y_{ij} = \boldsymbol{z}_{ij}^{\mathrm{T}}\boldsymbol{\beta} + g(\boldsymbol{\phi}_{ij}) + \epsilon_{ij}, \;\; \epsilon_{ij} \sim \mathcal{N}(0, \sigma^2_{FIX}), \tag{5.1}$$

Citation	Predictive	Community-based free-living data	Outputs recovery score?	Num of participants	Accelerometer duration
Ours	**Yes**	**Yes**	**Yes**	24	3 days x 8 weeks
Gao et al. (2019)	**Yes**	No	**Yes**	4-10 minutes x 155 trials	
Kumar et al. (2013)	**Yes**	No*	**Yes**	15	5 hours
Patel et al. (2010)	**Yes**	No	**Yes**	24	During assessment only
Yu et al. (2016)	**Yes**	No	**Yes**	5	1 hour x 10 days
Roy et al. (2009)	**Yes**	No	No	10	During assessment only
Rau et al. (2013)	**Yes**	No	**Yes**	12	During assessment only
Ferrari et al. (2007)	No	**Yes**	No	7 days x 4 weeks	
Lang et al. (2007)	No	No*	**Yes**	34	24 hours
Gebruers et al. (2014)	No	No*	**Yes**	129	48 hours
Lee et al. (2018)	No	No*	No	24	3 days
Urbin et al. (2015)	No	Mixed	**Yes**	35	22 hours
Uswatte et al. (2006a)	No	**Yes**	**Yes**	222	3 days x waking hours
Thrane et al. (2011)	No	**Yes**	**Yes**	31	24 hours
van der Pas et al. (2011)	No	**Yes**	**Yes**	45	3 days
Jing et al. (2011)	No	**Yes**	**Yes**	51	-
Rand and Eng (2012)	No	**Yes**	**Yes**	60	6 days
Liu et al. (2018)	No	**Yes**	No	10	8 hours
Joseph et al. (2018)	No	**Yes**	No	23	18 days
Chen et al. (2018)	No	**Yes**	**Yes**	82	6 days x 4 weeks
Narai et al. (2016)	No	**Yes**	**Yes**	19	24 hours
Bailey et al. (2015)	No	**Yes**	**Yes**	126	25-26 hours
Wei et al. (2018)	No	**Yes**	**Yes**	24	3 hours x 7 days x 4 weeks
Salazar et al. (2014)	No	No	No	4	During assessment only

Table 5.12: Review of related user studies. Asterisk (*) denotes studies conducted in hospital inpatient settings.

where $\boldsymbol{z}_{ij}^{\mathrm{T}}\boldsymbol{\beta}$ is the fixed effects with regression coefficients $\boldsymbol{\beta}$ of dimension K', ϵ_{ij} are independent random errors and $g(\boldsymbol{\phi}_{ij})$ is an unknown nonlinear function of $\boldsymbol{\phi}_{ij}$. $\boldsymbol{\phi}_{ij}$ can be selected as subsets of $\boldsymbol{z}_{ij}$ with dimension K_g. For each patient, we include in our sets of covariates the initial CAHAI score y_{i0}, which is assessed at the start of the study. When we are using the GMM-based set of covariates from Chapter 4, then $\boldsymbol{z}_{ij} = ((\boldsymbol{z}_{ij}^{GMM})^T, y_{i0})^T$ from Equation (4.8), and similarly when we use the MIL-based set of covariates from Equation (4.12) then $\boldsymbol{z}_{ij} = ((\boldsymbol{z}_{ij}^{MIL})^T, y_{i0})^T$. As one of the nonparametric Bayesian regression approaches, the following zero-mean GP prior can be used to model the unknown function $g(\boldsymbol{\phi}_{ij})$ as

$$g(\boldsymbol{\phi}_{ij}) \sim GP(\mathbf{0}, \boldsymbol{\kappa}(\cdot, \cdot; \boldsymbol{\theta}_g)), \tag{5.2}$$

where $\boldsymbol{\kappa}(\cdot, \cdot; \boldsymbol{\theta}_g)$ is the kernel covariance function parameterized by $\boldsymbol{\theta}_g$. A popular choice of the covariance kernel is a squared exponential covariance kernel given by

$$\boldsymbol{\kappa}(\phi_{ij,l}, \phi'_{ij,l}; \boldsymbol{\theta}_g) = v_0 \exp\left\{ -\frac{1}{2}\sum_{l=1}^{K_g} w_l(\phi_{ij,l} - \phi'_{ij,l})^2 \right\}, \tag{5.3}$$

where $\phi_{ij,l}$ is the l-th element of $\boldsymbol{\phi}_{ij}$, $\boldsymbol{\theta}_g = (v_0, w_1, \ldots, w_{K_g}, \sigma^2_{RDM})^{\mathrm{T}}$ is the set of hyper-parameters. σ^2_{RDM} is the noise of observations from the Gaussian Process and is regarded as a hyperparameter. Other choices of the kernel covariance can be found in Shi and Choi (2011). The fixed effects in Equation (5.1) provides a clear physical explanation between the CAHAI response y_{ij} and the new features obtained in Section 4.2.1 (for the GMM-based set of features) or Section 4.2.2 (for the MIL-based set of features), while the unexplained part can be modelled by the nonlinear random effects $g(\boldsymbol{\phi}_{ij})$.

5.2.1 Estimation

The estimation of parameters in the fixed effects and the estimation of hyper-parameters in the nonlinear random effects can be obtained by using the following iterative procedure

(1) Initialize $\boldsymbol{\beta}^{(0)}$ by fitting the linear regression model $y_{ij} = \boldsymbol{z}_{ij}^T\boldsymbol{\beta}^{(0)} + r_{ij}^{(0)}$, where $r_{ij}^{(0)}$ is the residual error.

(2) Set $r_{ij}^{(\tau)} \leftarrow y_{ij} - \boldsymbol{z}_{ij}^T\boldsymbol{\beta}^{(\tau)}$, where $\leftarrow$ is an assignment operation. Then, fit the Gaussian Process to predict each residual r_{ij}, using the covariates in $\boldsymbol{\phi}_{ij}$, i.e. $r_{ij}^{(\tau)} = \hat{g}^{(\tau)}(\phi) + \gamma_{ij}^{(\tau)}$, where $\gamma_{ij}^{(\tau)}$ is an error term.

We can estimate the hyperparameters in the non-linear mixed effects $g(\boldsymbol{\phi}_{ij})$ by using the Empirical Bayesian method, which can be implemented using the *R* package *GPFDA* (Shi and Cheng, 2014). Once the estimate of the hyperparameter is obtained, the fitted value $\hat{g}(\boldsymbol{\phi}_{ij})$ can be calculated.

(3) Update $\hat{\beta}^{(\tau+1)}$ given $\hat{g}^{(\tau)}(\boldsymbol{\phi}_{ij})$. To do this, fit a linear regression to predict γ_{ij} using $\boldsymbol{z}_{ij}$ as covariates, i.e. $\gamma_{ij}^{(\tau)} = \boldsymbol{z}_{ij}^T\boldsymbol{\beta}^{(\tau+1)} + \epsilon_{ij}^{(\tau+1)}$ using a linear regression model.

As shown in Equation (5.1), $\epsilon_{ij}^{(\tau+1)}$ is assumed to be normally distributed with zero mean and heteroscedastic variance of σ_{FIX}^2. This variance is calculated based on the sum of squared deviations in linear regression.

(4) Repeat step (2) and (3) until convergence.

5.2.2 Prediction: random effects

It is of interest to predict the recovery level y_{ij}^* at a new set of input $(\boldsymbol{z}_{ij}^*, \boldsymbol{\phi}_{ij}^*)$ based on the training data $\mathcal{D}$, where the i^{th} patient is not included in the training data. It is straightforward to predict the fixed effects part by using the estimated coefficients from the training data. Therefore, we will first discuss calculating the prediction to the random effects part.

Suppose we let $\boldsymbol{C}$ be the covariance matrix of $g(\boldsymbol{\phi})$ with each element calculated from a covariance kernel with the estimated hyper-parameters. $\boldsymbol{C}$ is a square matrix, with each dimension of the matrix being the number of datapoints in the training dataset. Then following Shi and Choi (2011), the predictive distribution of the random effect is Gaussian distribution with mean and variance given by

$$\mathrm{E}(g(\boldsymbol{\phi}^*_{ij})|\mathcal{D}) = \boldsymbol{c}^{*\,T}_{ij}(\boldsymbol{C} + \sigma^2_{RDM}\boldsymbol{I})^{-1}\boldsymbol{r} \tag{5.4}$$

$$\mathrm{Var}(g(\boldsymbol{\phi}^*_{ij})|\mathcal{D}) = \kappa(\boldsymbol{\phi}^*_{ij}, \boldsymbol{\phi}^*_{ij}) - \boldsymbol{c}^{*\,T}_{ij}(\boldsymbol{C} + \sigma^2_{RDM}\boldsymbol{I})^{-1}\boldsymbol{c}^*_{ij} + \sigma^2_{RDM}, \tag{5.5}$$

where $\boldsymbol{r}$ is the vector of fixed effect residuals (similar to r_{ij}) associated with datapoints in the training set $\mathcal{D}$ (and has length equal to the number of training datapoints), $\kappa(\boldsymbol{\phi}^*_{ij}, \boldsymbol{\phi}^*_{ij})$ is the covariance corresponding to the new data point, $\boldsymbol{C}$ is the covariance matrix of $g(\boldsymbol{\phi})$ and $\boldsymbol{c}^*_{ij}$ is a vector the same length as the number of datapoints in the training dataset, and each element of it contains the covariance between the new point which has covariates $\boldsymbol{\phi}^*_{ij}$, and each of the datapoints in the training dataset. The transpose of $\boldsymbol{c}^*_{ij}$ is $\boldsymbol{c}^{*T}_{ij}$.

5.2.3 Prediction: mixed effects

The predictive distribution of the CAHAI score of the i^{th} patient for the j^{th} week, incorporating both fixed and random effects (from Section 5.2.2), is:

$$\mathrm{E}(g(\boldsymbol{\phi}^*_{ij})|\mathcal{D}) = \boldsymbol{z}^{*\,\mathrm{T}}_{ij}\boldsymbol{\beta} + \boldsymbol{c}^{*\,T}_{ij}(\boldsymbol{C} + \sigma^2_{RDM}\boldsymbol{I})^{-1}\boldsymbol{r} \tag{5.6}$$

$$\mathrm{Var}(g(\boldsymbol{\phi}^*_{ij})|\mathcal{D}) = \sigma^2_{FIX} + \kappa(\boldsymbol{\phi}^*_{ij}, \boldsymbol{\phi}^*_{ij}) - \boldsymbol{c}^{*\,T}_{ij}(\boldsymbol{C} + \sigma^2_{RDM}\boldsymbol{I})^{-1}\boldsymbol{c}^*_{ij} + \sigma^2_{RDM}, \tag{5.7}$$

where $\boldsymbol{z}^*_{ij}$ and $\boldsymbol{\beta}$ are the fixed effects covariates and parameters from Equation (5.1), and σ^2_{FIX} is the variance of the fixed effects shown in Equation (5.1). Other notations are

defined in Section 5.2.2.

5.3 Results

In this section we give an account of predictive results using GMM-based (Section 5.3.1) and MIL-based (Section 5.3.2) sets of covariates. The overall results are compared numerically in Table 5.31 and are discussed in Section 5.4.

5.3.1 Prediction using GMM-based covariates

Based on the clusters in Section 4.3.1, we can calculate the new feature vector $\boldsymbol{z}_{ij}^{GMM} = (z_{ij,1}^{GMM}, \ldots, z_{ij,K}^{GMM}, y_{i0})^{\mathrm{T}}$ by Equation (4.8), where $\boldsymbol{z}_{ij}$ can be thought as a $K' = (K+1)$-dimensional vector of variables used to model the CAHAI score. These features are referred to as *GMM-based covariates*.The least absolute shrinkage and selection operator (LASSO) (Tibshirani, 1996) method is used to select the significant clusters for modelling, as only a subset of the covariates are useful for modelling the CAHAI score. In this case with 20 potential covariates to choose from, the LASSO method is preferred for computational efficiency reasons to simpler methods like best subsets (which may consider as many as $2^{20} = 1048576$ models with different combinations of covariates), or forward or backward selection methods (which may consider as many as $20 + 19 + ... + 1 = 210$ models, though would likely stop before then when some model fit criterion is reached). Instead, the LASSO method merely requires the model to be evaluated with several different values for its shrinkage parameter.

The clusters corresponding to the selected covariates are shown in Table 5.32 and visualizations of the selected clusters can be found in Figures 4.31 and 4.32. We can see from Table 5.32 that for both groups, Cluster 2, Cluster 3 and Cluster 15 are selected as the significant clusters in predicting the CAHAI score.

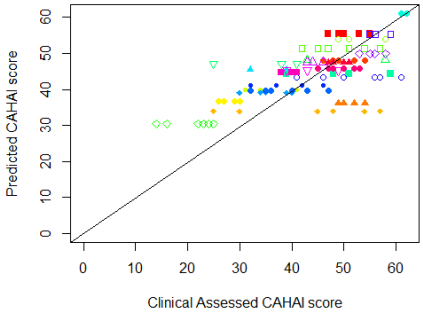
	Acute Patients	Chronic Patients
GMM-based covariates with GP	5.423231	3.633249
GMM-based covariates without GP	6.72677	3.528335
MIL-based covariates with GP	**5.289253**	**3.468572**
MIL-based covariates without GP	6.642244	3.541442

Table 5.31: Comparison of Mean Square Error of prediction both with and without using GP prior. Acute patients are those who suffered a stroke less than 6 months ago, and chronic patients are those who suffered a stroke more than 6 months ago. The best performing model/covariate combination for both acute and chronic sets of patients is bolded.

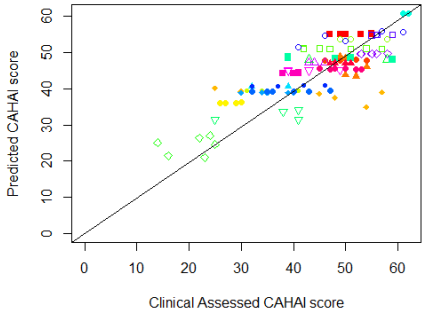
Table 5.32: Selected clusters for the predictive model.

	Selected clusters
Acute group	Cluster 2, Cluster 3, Cluster 5, Cluster 10 Cluster 12, Cluster 15, Cluster 18
Chronic group	Cluster 2, Cluster 3, Cluster 4, Cluster 8 Cluster 15, Cluster 20

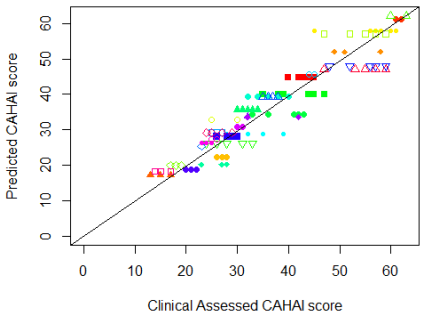
Based on these selected clusters, then we can model the CAHAI score for the acute patients and the chronic patients using the method in Section 5.2. We model the acute patients and the chronic patients separately due to different recovery levels in those two groups, see Figure 5.33 for example. A leave-one-patient-out cross-validation method is used to validate the model and calculate the root mean squared error (RMSE) of predictions. For both the acute patients and the chronic patients, the data of each patient is selected as the test data and the data of the other patients are used to train the model according to Section 5.2.1. The trained model is then used to predict the CAHAI score of the patient whose data is selected as test data using the algorithm given in Section 5.2.2. As a comparison, we also calculate the results by using the predictive model without the random effects. These results are shown in Figure 5.31.



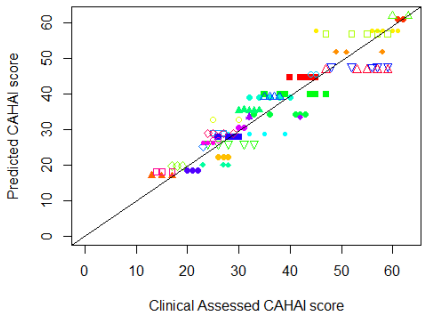
(a) Results for acute patients without using nonlinear random effects.



(b) Results for acute patients with nonlinear random effects.



(c) Results for chronic patients without using nonlinear random effects.



(d) Results for chronic patients with nonlinear random effects.

Figure 5.31: Clinically-assessed CAHAI score and predicted CAHAI using the accelerometer data, using GMM-based covariates described in Section 4.2.1.

5.3.2 Prediction using MIL-based covariates

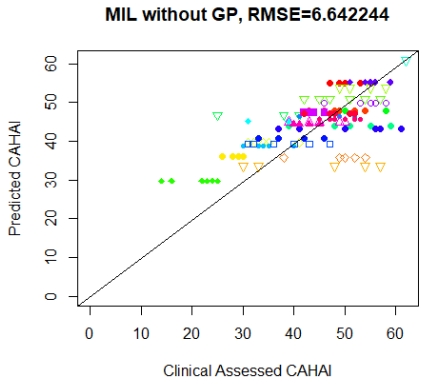
Likewise, we can use the MIL-based covariates from Section 4.2.2, using the predictive method discussed in Section 5.3.1, and include the initial CAHAI score y_{i0} as a covariate. As before, we model acute patients (those who had a stroke in the past six months) separately to chronic patients (those who had a stroke more than six months ago), due to the different recovery speeds of both groups of patients. As before, leave-one-patient-out cross validation predictive performance is evaluated for both groups of patients, each both with and without using the GP prior to model the nonlinear random effects.

In Figure 5.32, it is interesting that the RMSE of predictions, both for the chronic and acute groups of patients and both with and without use of nonlinear mixed effects, is slightly lower (better) when using these MIL-based covariates rather than when GMM-based covariates are used in Section 5.3.1. This could indicate that clustering leads to quanitzation error which slightly harms accuracy in this application, that is sliding windows relating to different recovery levels can be assigned to the same GMM cluster. Covariates similar to those discussed in Section 4.2.2 may not be as prone to this issue as they don't rely on clustering-based methods and therefore aren't prone to quantization error, as mentioned in Gao et al. (2019). Furthermore, since the MIL-based covariates are not dependent on specifics of stroke as a disease, they have the potential to generalize to any similar study for completely different movement disorders, as long as periodic rehabilitation assessments can be completed alongside the collection of free-living accelerometer data from the relevant locations on the human body.

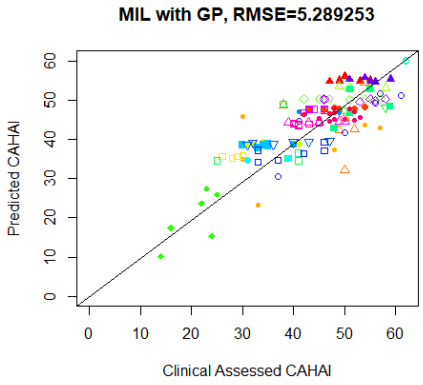
On the other hand, it may be useful in this application to be able to interpret how covariates are generated, so then it may be preferred to use the GMM-based model, even at the expense of slightly less accurate predictive performance.

5.4 Discussion

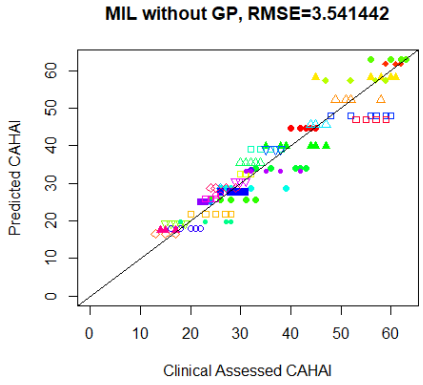
In this section, we discuss the results of running our models, comparing with and without using the GP prior in Equation (5.1) in Section 5.4.1, as well as compare the model's accuracy when using GMM-based covariates to the accuracy when using MIL-based covariates in Section 5.4.2.



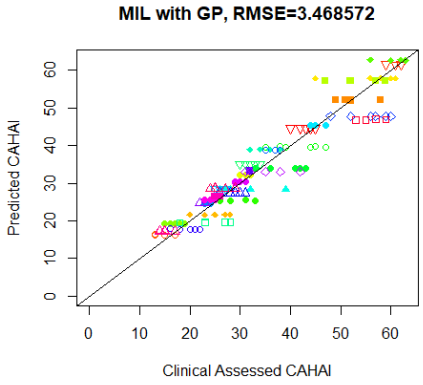
(a) Results for acute patients without using nonlinear random effects, using Multi-instance learning covariates



(b) Results for acute patients with nonlinear random effects, using Multi-instance learning covariates



(c) Results for chronic patients without using nonlinear random effects, using Multi-instance learning covariates

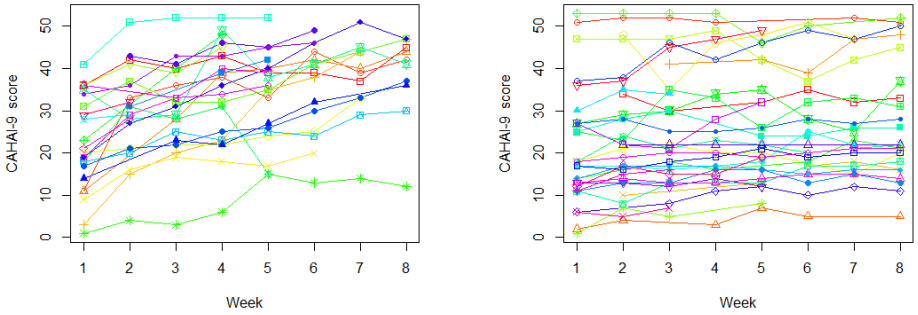


(d) Results for chronic patients with nonlinear random effects, using Multi-instance learning covariates

Figure 5.32: Clinical assessed CAHAI score and predicted CAHAI using the accelerometer data, using MIL-based covariates described in Section 4.2.2.

5.4.1 Performance with and without GP prior

We see in Table 5.31 that for both GMM-based and MIL-based sets of covariates and on both chronic and acute patient subgroups, the Root Mean Square Error (RMSE) is lower (better) when using the non-linear GP prior for modelling the random effects for each patient as opposed to simply using a fixed effects model. We see when comparing subplots (a) and (b) in Figure 5.31 that when using GMM-based covariates, the predictions are



(a) Acute patients (b) Chronic patients

Figure 5.33: The clinical assessment of the recovery level (CAHAI-9 score) for acute patients (left panel) and chronic patients (right panel). Each curve represents observations for one patient.

quite similar for both patients whether or not the GP prior is used, apart from predictions for a small number of patients for whom the accuracy improves when using the GP prior. This is particularly the case for that patient denoted with the green diamond and the one with the orange triangle. A similar pattern is observed when using MIL-based covariates in subplots (a) and (b) in Figure 5.32.

For acute patients (stroke occurred less than 6 months ago), it seems that patients generally improve over the course of the study, and those with worse scores in the beginning tend to improve at a slightly faster rate than others (see Figure 5.33), though recovery rate is highly patient dependent. Therefore, acute patients with lower initial CAHAI scores, or indeed with poorer movement shown in CAHAI scores, can be difficult to predict for over the course of the study. The patient-specific non-linear random effects $g(\phi)$ in the mixed effects model in Equation (5.1) help to account for this heterogeneity between patients.

However, for chronic patients (stroke occurred more than 6 months ago), the GP prior does not help much to improve model performance. This is because for most patients, the CAHAI score does not improve much from the initial CAHAI score (a covariate in our model), so improvement is slow or non-existent for all chronic patients, and the accuracy

of predictions is less affected by heterogeneity between patients. We see in Table 5.31 that the benefit of the GP prior is mainly present when predicting for acute patients, rather than chronic patients.

5.4.2 Comparison of performance between GMM and MIL-based covariates

Another noticeable aspect of performance in Table 5.31 is performance of one set of covariates over another. The set of MIL-based covariates tend to perform slightly better than the set of GMM-based covariates. We have identified two potential reasons for this. Firstly, the MIL-based covariates' predictions in Equation (4.12) are based on $\boldsymbol{x}_{ij,h}$, which is the concatenation of (preprocessed) sliding window datapoints from both the paretic and non-paretic upper limbs. $\boldsymbol{x}_{ij,h}$ represents the patient's upper limb movement on *both* sides of the body at the time that the h^{th} sliding window was sampled. This is because $\boldsymbol{x}_{ij,h}$ is the concatenation of vectors $\boldsymbol{x}_{1ij,h}$ and $\boldsymbol{x}_{2ij,h}$, which are the movement data from the paretic and non-paretic sides, respectively. Thus, $\zeta(\boldsymbol{x}_{ij,h})$ used in Equations (4.10) - (4.12) captures whether or not the patient was moving both arms *simultaneously* or if the paretic side had weaker movement compared to the non-paretic side. The capability of a patient to simultaneously move both upper limbs and with the same strength is crucial to understanding and monitoring recovery from hemiparesis in stroke.

Another reason the MIL-type approach detailed in Equation (4.12) can result in better performance compared to the GMM-based approaches is that the MIL-based one is not subject to *quantization error*, where cluster membership information does not accurately represent the underlying data. It is possible that when the cluster memberships are obtained, some of the (preprocessed sliding window) datapoints are quite dissimilar to other datapoints in the same cluster (Gao et al., 2019).

5.5 Contributions and Conclusion

We present a useful method for modelling recovery in human movement disorders using free-living accelerometer data. Overall, good predictive results are obtained in Table 5.31, which could make remote automated stroke rehabilitation assessment in a community setting, and the resulting cost savings this would bring, feasible. To our knowledge, this is the first user study of its kind where accelerometer data from a community dwelling setting is used to predict a measure of stroke rehabilitation (see Table 5.12 for a survey), and also the first study in this application are to make use of the NLME model (described in Section 5.2).

As discussed in Chapter 4, sets of covariates can both be introspected using visualization techniques to obtain some interesting and useful understanding on how the model's covariates were obtained. In future work, we may attempt to use both sets of covariates (GMM and MIL-based) together in the same model with variable selection techniques, to see if we would obtain a better predictive model.

Future work may also take into account use of censored regression techniques. The model for prediction of CAHAI score in Equation (5.1) doesn't take into account that the CAHAI score lies in a bounded interval $[9, 63]$. This could be solved, and perhaps the predictive performance improved, by using a censored regression model, such as a Tobit model (Tobin, 1958). Future work could also include a hyperparameter search (with patients split into training, validation and test groups) to find the optimal number of covariates for both GMM-based (i.e. number of components in the mixture model in Section 4.2.1) and MIL-based sets of covariates (from Section 4.2.2).

Chapter 6

Conclusion

6.1 Findings on evaluating upper limb function

In Chapter 5 , we detail the results of what is to our knowledge the first user study for prediction of stroke rehabilitation using free-living accelerometer data in a community setting (see Table 5.12 for a comparison with other studies). Furthermore, as shown in Section 5.3, the models are accurate, attaining an RMSE of 5.3 for chronic patients and 3.5 for acute patients using Multi-Instance Learning-based covariates. This level of accuracy would be useful for medical professionals to reduce the amount of costly home visits which need to be made, as recovery levels can be assessed automatically and objectively remotely by the prediction system. Recovery can automatically be assessed even for patients who are deemed to be a lower priority due to cost constraints.

Of the two covariate generation methods discussed in Chapter 4, the GMM-based one (Section 4.2.1) calculates based on the ratio of time which the paretic side upper limb is used in comparison to the non-paretic side upper limb. Having explanations of how the model works could aid clinicians' understanding. On the other hand, our Multi-Instance Learning-based model (Section 4.2.2) lends itself to interpretability through seasonal decomposition of inferred recovery levels. We can see for individual patients how their

recovery progressed (see Section 4.3.2).

6.2 Both Statistical and Machine Learning Models perform well

The non-linear mixed effects (NLME) model in Chapter 5 is very useful for predicting CAHAI scores given the inputted covariates. Although this is not the first work to use NLME for longitudinal data analysis, it is unique in the field of accelerometer data analysis for stroke patients' recovery.

Our current feature extraction methods in Chapter 4 are basic but perform quite well when input into the models in Chapter 5. In Chapter 3, we explore recurrent neural network based feature extraction for accelerometer data, which can perform supervised classification and regression tasks on time series, without dependence on a particular choice of sliding window duration, by taking into account the entire temporal context of events before or even after a particular timepoint on which we would like to perform a prediction on. Were a scheme to be developed to employ an RNN to calculate Equation 4.12, then it is reasonable to assume that we could obtain an even better accuracy in our model. One potential reason why this would lead to better model accuracy would be the potential for the recurrent model to take into account not just how often one arm is used in comparison to the other (as in our current models), but also aspects of temporal context such as the amount of time for which an arm is used for during each bout of activity, which may be much shorter on the paretic upper limb compared to the non-paretic side on patients who are severely affected by stroke hemiparesis due to them getting fatigued quicker when performing activities.

Also, the supervised learning models for Human Activity Recognition (HAR) discussed in Chapter 3 are very interesting by themselves, and show which neural net-

work model architecture, particularly LSTM-based ones, are useful for HAR in semi-naturalistic environments.

6.3 Future Work

Future work could include improving the covariates used. In Chapter 4, we mentioned two sets of covariates, each of which were generated in different ways: GMM-based (Section 4.2.1) and MIL-based (Section 4.2.2). A better performance could be obtained simply by using both sets of covariates together and employing a variable selection technique to pick the best subset of variables from both sets of covariates. Yet another improvement would be to design a covariate generation pipeline which combined the interpretability advantages of both of the aforementioned methods (the ability to plot the the mixture component distributions for the GMM-based method and the ability to decompose inferred recovery levels into trend, seasonal and residual components for the MIL-based method). This could perhaps be achieved either by using the mixture model component memberships of sliding window datapoints to calculate the inferred CAHAI for the datapoint in Equation 4.12.

Other future work on model visualization includes validating our inferences of daily recovery trends. When we decompose the recovery time series into trend, seasonal and residual components (see Section 4.3.2) for our MIL-based covariates, we have no validation that the inferred daily recovery trends are accurate about which days the patient is recovering or deteriorating. This is because we only have ground truth measurements once per week per patient, at most. Perhaps if additionally, a time and cost-efficient self-administered questionnaire-based assessment of stroke recovery, such as the Motor Activity Log (Uswatte et al., 2006b) were used in validation in data collection in a new user study, then we could validate the inferred recovery levels at a daily frequency.

Finally, for our MIL-based covariate generation, it would be good to perform a com-

parison between different (supervised learning) regression models for Equation 4.12, such as neural networks, random forest, etc. For our GMM-based model, we may experiment with using different models to represent our data as counts of observations of symbols and calculate features based on ratios of these counts between both hands. For example, recently on accelerometer data, approaches discussed in Lin et al. (2007) and Ciliberto and Roggen (2019) represent the data in a simpler way to extract features from raw accelerometer data and can be quite computationally efficient.

Bibliography

Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81–105.

Anderson, J. L., Green, A. J., Yoward, L. S., and Hall, H. K. (2018). Validity and reliability of accelerometry in identification of lying, sitting, standing or purposeful activity in adult hospital inpatients recovering from acute or critical illness: a systematic review. *Clinical Rehabilitation*, 32(2):233–242. PMID: 28805075.

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., et al. (2015). Spark SQL: Relational data processing in Spark. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1383–1394. ACM.

Bachlin, M., Roggen, D., Troster, G., Plotnik, M., Inbar, N., Meidan, I., Herman, T., Brozgol, M., Shaviv, E., Giladi, N., and others (2009). Potentials of Enhanced Context Awareness in Wearable Assistants for Parkinson's Disease Patients with the Freezing of Gait Syndrome. In *Proceedings of the 2009 ACM International Symposium on Wearable Computers* .

Bailey, R. R., Klaesner, J. W., and Lang, C. E. (2015). Quantifying real-world upper-limb activity in nondisabled adults and adults with chronic stroke. *Neurorehabilitation and Neural Repair*, 29(10):969–978.

Barreca, S. R., Stratford, P. W., Lambert, C. L., Masters, L. M., and Streiner, D. L. (2005). Test-retest reliability, validity, and sensitivity of the Chedoke arm and hand activity inventory: a new measure of upper-limb function for survivors of stroke. *Archives of Physical Medicine and Rehabilitation*, 86(8):1616–1622.

Bellman, R. (1961). Curse of dimensionality. *Adaptive Control Processes: A Guided Tour. Princeton, NJ*.

Best, M. J. and Chakravarti, N. (1990). Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1-3):425–439.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York.

Blei, D. M., Jordan, M. I., et al. (2006). Variational inference for Dirichlet Process mixtures. *Bayesian analysis*, 1(1):121–143.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Bulling, A., Blanke, U., and Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys (CSUR)*, 46(3):33.

Chakravarti, N. (1989). Isotonic median regression: a linear programming approach. *Mathematics of Operations Research*, 14(2):303–308.

Chavarriaga, R., Sagha, H., Calatroni, A., Digumarti, S. T., Tröster, G., Millán, J. d. R., and Roggen, D. (2013). The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042.

Chen, H., Lin, K., Hsieh, Y., Wu, C., Liing, R., and Chen, C. (2018). A study of predictive validity, responsiveness, and minimal clinically important difference of arm accelerometer in real-world activity of patients with chronic stroke. *Clinical Rehabilitation*, 32(1):75–83.

Ciliberto, M. and Roggen, D. (2019). Wlcsscuda: a cuda accelerated template matching method for gesture recognition. In *Proceedings of the 2019 International Symposium on Wearable Computers*. Association for Computing Machinery.

Cleveland, R. B., Cleveland, W. S., McRae, J. E., and Terpenning, I. (1990). STL: A seasonal-trend decomposition. *Journal Official Statistics*, 6(1):3–73.

Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314.

Daubechies, I. (1990). The wavelet transform, time-frequency localization and signal analysis. *IEEE Transactions on Information Theory*, 36(5):961–1005.

Ferrari, P., Friedenreich, C., and Matthews, C. E. (2007). The role of measurement error in estimating levels of physical activity. *American Journal of Epidemiology*, 166(7):832–840.

Figo, D., Diniz, P. C., Ferreira, D. R., and Cardoso, J. M. (2010). Preprocessing techniques for context recognition from accelerometer data. *Personal and Ubiquitous Computing*, 14(7):645–662.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The Elements of Statistical Learning*, volume 1. Springer series in Statistics, Springer, Berlin.

Gao, Y., Long, Y., Guan, Y., Basu, A., Baggaley, J., and Ploetz, T. (2019). Towards reliable, automated general movement assessment for perinatal stroke screening in infants using wearable accelerometers. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 3(1):1–22.

Gebruers, N., Truijen, S., Engelborghs, S., and De Deyn, P. P. (2014). Prediction of upper limb recovery, general disability, and rehabilitation status by activity measurements assessed by accelerometers or the Fugl-Meyer score in acute stroke. *American Journal of Physical Medicine & Rehabilitation*, 93(3):245–252.

Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Graves, A. (2012). Supervised sequence labelling. In *Supervised Sequence Labelling with Recurrent Neural Networks*, pages 5–13. Springer.

Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2015). LSTM: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

Halloran, S., Tang, L., Guan, Y., Shi, J. Q., and Eyre, J. (2019). Remote monitoring of stroke patients' rehabilitation using wearable accelerometers. In *Proceedings of the 2019 ACM International Symposium on Wearable Computers*. ACM.

Hammerla, N. Y., Halloran, S., and Plötz, T. (2016). Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1533–1540. AAAI Press.

Hammerla, N. Y. and Plötz, T. (2015). Let's (not) stick together: pairwise similarity biases cross-validation in activity recognition. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2015*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hornik, K. (1993). Some new results on neural network approximation. *Neural Networks*, 6(8):1069–1072.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *International Conference on Machine Learning*, pages 754–762.

Jing, L., Zhou, Y., Cheng, Z., and Wang, J. (2011). A recognition method for one-stroke finger gestures using a MEMS 3D accelerometer. *The Institute of Electronics Information and Communication Engineers (IEICE) Transactions on Information and Systems*, 94(5):1062–1072.

Jolliffe, I. T. (1986). Principal Component Analysis and Factor Analysis. In *Principal Component Analysis*, pages 115–128. Springer.

Jones, M. and Rice, J. A. (1992). Displaying the important features of large collections of similar curves. *The American Statistician*, 46(2):140–145.

Joseph, C., Conradsson, D., Hagströmer, M., Lawal, I., and Rhoda, A. (2018). Objectively assessed physical activity and associated factors of sedentary behavior among

survivors of stroke living in Cape Town, South Africa. *Disability and rehabilitation*, 40(21):2509–2515.

Juan, A. and Vidal, E. (2002). On the use of Bernoulli mixture models for text classification. *Pattern Recognition*, 35(12):2705–2710.

Kestelyn, J. (2013). Introducing Parquet: Efficient columnar storage for Apache Hadoop. *Cloudera Blog*, 3.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kumar, D., Gubbi, J., Yan, B., and Palaniswami, M. (2013). Motor recovery monitoring in post acute stroke patients using wireless accelerometer and cross-correlation. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6703–6706. IEEE.

Lang, C. E., Bland, M. D., Bailey, R. R., Schaefer, S. Y., and Birkenmeier, R. L. (2013). Assessment of upper extremity impairment, function, and activity after stroke: foundations for clinical decision making. *Journal of Hand Therapy*, 26(2):104–115.

Lang, C. E., Wagner, J. M., Edwards, D. F., and Dromerick, A. W. (2007). Upper extremity use in people with hemiparesis in the first few weeks after stroke. *Journal of Neurologic Physical Therapy*, 31(2):56–63.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Lee, I.-M., Shiroma, E. J., Evenson, K. R., Kamada, M., LaCroix, A. Z., and Buring, J. E. (2018). Accelerometer-measured physical activity and sedentary behavior in relation to all-cause mortality: the womens health study. *Circulation*, 137(2):203–205.

Liaw, A. and Wiener, M. (2002). Classification and regression by RandomForest. R News 2 (3): 18–22. *URL: http://CRAN. R-project. org/doc/Rnews*.

Lin, J., Keogh, E., Wei, L., and Lonardi, S. (2007). Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144.

Liu, X., Rajan, S., Ramasarma, N., Bonato, P., and Lee, S. I. (2018). Finger-worn sensors for accurate functional assessment of the upper limbs in real-world settings. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4440–4443. IEEE.

MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. Oakland, CA, USA.

McLachlan, G. J., Lee, S. X., and Rathnayake, S. I. (2019). Finite mixture models. *Annual Review of Statistics and its Application*, 6:355–378.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Tsai, D., Amde, M., Owen, S., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241.

Moore, S. A., Da Silva, R., Balaam, M., Brkic, L., Jackson, D., Jamieson, D., Ploetz, T., Rodgers, H., Shaw, L., van Wijck, F., and others (2016). Wristband Accelerometers to motiVate arm Exercise after Stroke (WAVES): study protocol for a pilot randomized controlled trial. *Trials*, 17(1):508.

Narai, E., Hagino, H., Komatsu, T., and Togo, F. (2016). Accelerometer-based monitoring of upper limb movement in older adults with acute and subacute stroke. *Journal of Geriatric Physical Therapy*, 39(4):171–177.

Nguyen, H. D., McLachlan, G. J., Ullmann, J. F., and Janke, A. L. (2016). Spatial clustering of time series via mixture of autoregressions models and Markov random fields. *Statistica Neerlandica*, 70(4):414–439.

Noorkiv, M., Rodgers, H., and Price, C. I. (2014). Accelerometer measurement of upper extremity movement after stroke: a systematic review of clinical studies. *Journal of Neuroengineering and Rehabilitation*, 11(1):144.

Ordóñez, F. J. and Roggen, D. (2016). Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1):115.

Patel, S., Hughes, R., Hester, T., Stein, J., Akay, M., Dy, J., and Bonato, P. (2010). Tracking motor recovery in stroke survivors undergoing rehabilitation using wearable technology. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pages 6858–6861. IEEE.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ramsay, J. O. and Silverman, B. W. (2007). *Applied Functional Data Analysis: Methods and Case Studies*. Springer.

Rand, D. and Eng, J. J. (2012). Disparity between functional recovery and daily use of the upper and lower extremities during subacute stroke rehabilitation. *Neurorehabilitation and Neural Repair*, 26(1):76–84.

Rau, C.-L., Chen, Y.-P., Lai, J.-S., Chen, S.-C., Kuo, T.-S., Jaw, F.-S., and Luh, J.-J. (2013). Low-cost tele-assessment system for home-based evaluation of reaching ability following stroke. *TELEMEDICINE and e-HEALTH*, 19(12):973–978.

Reiss, A. and Stricker, D. (2012). Introducing a new benchmarked dataset for activity monitoring. In *Proceedings of the 2012 16th Annual International Symposium on Wearable Computers (ISWC)*.

Roy, S. H., Cheng, M. S., Chang, S.-S., Moore, J., De Luca, G., Nawab, S. H., and De Luca, C. J. (2009). A combined sEMG and accelerometer system for monitoring functional activity in stroke. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 17(6):585–594.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Salazar, A. J., Silva, A. S., Silva, C., Borges, C. M., Correia, M. V., Santos, R. S., and Vilas-Boas, J. P. (2014). Low-cost wearable data acquisition for stroke rehabilitation: a proof-of-concept study on accelerometry for functional task assessment. *Topics in Stroke Rehabilitation*, 21(1):12–22.

Santisteban, L., Térémetz, M., Bleton, J.-P., Baron, J.-C., Maier, M. A., and Lindberg, P. G. (2016). Upper limb outcome measures used in stroke rehabilitation studies: a systematic literature review. *PloS one*, 11(5):e0154792.

Seabold, S. and Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference*, volume 57, page 61. Scipy.

Shanahan, J. G. and Dai, L. (2015). Large scale distributed data science using Apache Spark. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2323–2324. ACM.

Sharma, A., Purwar, A., Lee, Y.-D., Lee, Y.-S., and Chung, W.-Y. (2008). Frequency based classification of activities using accelerometer data. In *2008 IEEE International*

conference on Multisensor Fusion and Integration for Intelligent Systems, pages 150–153. IEEE.

Shi, J. and Cheng, Y. (2014). GPFDA: apply gaussian process in functional data analysis. *R package*, https://CRAN.R-project.org/package=GPFDA.

Shi, J. Q. and Choi, T. (2011). *Gaussian Process Regression Analysis for Functional Data*. New York: Chapman and Hall/CRC.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sutskever, I., Martens, J., Dahl, G. E., and Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. *International Conference on Machine Learning*, 28(1139-1147):5.

Tang, L., Halloran, S., Shi, J. Q., Guan, Y., Cao, C., and Eyre, J. (2019). Evaluating upper limb function after stroke using the free-living accelerometer data (under review). *Statistical Methods in Medical Research*.

Thrane, G., Emaus, N., Askim, T., and Anke, A. (2011). Arm use in patients with subacute stroke monitored by accelerometry: association with motor impairment and influence on self-dependence. *Journal of Rehabilitation Medicine*, 43(4):299–304.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tobin, J. (1958). Estimation of relationships for limited dependent variables. *Econometrica: journal of the Econometric Society*, pages 24–36.

Urbin, M., Bailey, R. R., and Lang, C. E. (2015). Validity of body-worn sensor acceleration metrics to index upper extremity function in hemiparetic stroke. *Journal of Neurologic Physical Therapy: JNPT*, 39(2):111.

Uswatte, G., Giuliani, C., Winstein, C., Zeringue, A., Hobbs, L., and Wolf, S. L. (2006a). Validity of accelerometry for monitoring real-world arm activity in patients with subacute stroke: evidence from the extremity constraint-induced therapy evaluation trial. *Archives of Physical Medicine and Rehabilitation*, 87(10):1340–1345.

Uswatte, G., Taub, E., Morris, D., Light, K., and Thompson, P. A. (2006b). The motor activity log-28. *Neurology*, 67(7):1189–1194.

van der Pas, S. C., Verbunt, J. A., Breukelaar, D. E., van Woerden, R., and Seelen, H. A. (2011). Assessment of arm activity using triaxial accelerometry in patients with a stroke. *Archives of Physical Medicine and Rehabilitation*, 92(9):1437–1442.

Wang, J.-L., Chiou, J.-M., and Müller, H.-G. (2016). Functional data analysis. *Annual Review of Statistics and Its Application*, 3:257–295.

Wei, W. X., Fong, K. N., Chung, R. C., Myint, J. M., Cheung, H. K., and Chow, E. S. (2018). Utility of a unilateral accelerometer for monitoring upper extremity use in subacute stroke patients after discharge from hospital. *Assistive Technology*, pages 1–6.

Werbos, P. J. et al. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Wintermark, M., Sesay, M., Barbier, E., Borbély, K., Dillon, W. P., Eastwood, J. D., Glenn, T. C., Grandin, C. B., Pedraza, S., Soustiel, J.-F., et al. (2005). Comparative overview of brain perfusion imaging techniques. *Stroke*, 36(9):e83–e99.

Yang, J. B., Nguyen, M. N., San, P. P., Li, X. L., and Krishnaswamy, S. (2015). Deep convolutional neural networks on multichannel time series for human activity recognition. In *International Joint Conference on Artificial Intelligence*.

Yu, L., Xiong, D., Guo, L., and Wang, J. (2016). A remote quantitative Fugl-Meyer assessment framework for stroke patients based on wearable sensor networks. *Computer Methods and Programs in Biomedicine*, 128:100–110.

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient Distributed Datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*. USENIX Association.

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *International Conference on Machine Learning*, page 116.